Gauging Quality of Software Products Using Metrics

S. Farid[1], M.Alam[2], A. Akbar[3], M. M. Iqbal[4], F. A. Siddiqui[5]

[1]Computer Science Department, Bahauddin Zakariya University, Multan, Pakistan.
[2]Informatics Complex (ICCC), H-8/1, Islamabad, Pakistan.
[3,4]Statistics Department, Bahauddin Zakariya University, Multan, Pakistan.
[5]Engineering Science Department, University of Oxford, Parks Road Oxford, United Kingdom.
[1]shahidfarid@bzu.edu.pk

Abstract-In the current era of information technology, everyone has to deal with software products. It is crucial to gauge the quality of software products from its various aspects. Always there is need of standard and specific methods to measure any parameter in scientific manner. As quality of software is concerned, using quality metrics is a standard method which can be preferred over its (simpler) counterparts. This method enables to estimate various characteristics of a software system like cost, complexity, volume, size, function points, quality etc. these characteristics determines the quality. The aim of this study is to enhance the quality of the software developed in localized environment by getting these metrics implemented on different software products developed by the students' final year projects. Students has been divided into two groups. Conventional and controlled groups. The controlled group is bounded to follow the quality metrics during each phase of Software Development Life Cycle (SDLC) with expected quality enhancement. Results obtained by experimentation show that implementation of software quality metrics improves the quality of software product. Therefore, it is highly recommended to use the appropriate software quality metrics during all phases of SDLC.

Keywords-Software Engineering, Software Quality, Software Quality Assurance, Software Metrics, Experimentation.

I. INTRODUCTION

The development of software products in a systematic manner is the core of Software Engineering (SE). It provide the controlled umbrella activity which leads to the development of undisciplined and complex scratch to well-defined, easier and significant quality product. Software systems are becoming pervasive in modern civilization and the quality of software products are considered as one of the crucial elements for the success of overall software system. The users of computers (individuals and organizations) are increasing exponentially round the globe due to availability of latest machines and cost effectiveness in the market. This has elevated the need of quality products and hence as a result, software systems are propagating rapidly almost all businesses have deployed software to perform various functions like invoicing, billing, payments, inventory control, purchase, reordering etc. [i]. Due to this reason, there is an essential call for not only bug free but also quality software applications for individuals and business organizations [ii].Software systems have recently propagated greatly and become a pervasive occurrence both in the life of individuals and in culture at large. Accompanying the expansion growth of software use, it is crucial to ensure the high quality of software.

In this age of information technology, everyone has to deal with software products. The competition in software market is increasing day by day, but no organization can capture this market unless it does not produce quality systems and services [iii]. The quality of software products is not only accepted universally but now considered to be a vital element in business success too [iv]. Whereas software industry in not capable to provide customers with high quality products within predefined schedule and budget constraints. Major computer system projects were sometimes years late, and the resulting software was unreliable, hard to maintain and performed poorly [v]. This situation has often been referred to as software crises [vi]. As poor quality of software product may not only lead to financial loss or failure of mission but also loss of human life [vii].The awareness of users about software is growing with the use of computers and they need more powerful and sophisticated products. This phenomenon arises the challenge of developing better and cost effective products within allocated time ensuring the quality [viii].

It is relatively easy to discuss quality and quality assurance, but it is quite difficult to measure the various characteristics of quality in the different phases of the software development. Since 1970s, researchers and practitioners have been looking for ways to characterize software quality. They have found that software artifact can be broken down to constructs or

quality characteristics that can be assured and measured. This enables evaluation of quality through the evaluation of more detailed characteristics [xi]. These quality characteristics collectively reflect the overall quality of the system [x]. However, quality means conformance to predefined specifications and meet the customers' needs [vii]. In other words, quality reflects the user satisfaction. Whereas, it is urged by [xi] quality of software system cannot be measured with a single factor like usability or user satisfaction. Hence it is required to consider other quality factors like availability, complexity, size, flexibility, accessibility, reliability, maintainability etc.

Software customers do not know what is going on behind the screen as they work with the interface of the product.This has elevated the responsibility of the software engineers and developers to develop quality product that must be reliable and meet the user needs. Software quality metrics can be utilized to assure the quality of products. There were no availability of mature metrics couple of decade ago. At that time, product quality were evaluated when product was ready to deliver. At this stage product finishing can be enhanced but errors occurred during different phases of SDLC are difficult to rectify. Evaluation of quality at this stage may result to software failure or nonconformance to the user requirements. Due to this reason, instead of delivering efficient software product, quality procedures become the mean of avoiding blame [xii].Many users are willing to pay higher price of software product provided that quality is excellent as well as demanding [xiii]. Hence, a competitive environment has created for most of the modern software companies due to tight schedule and budget constraints [xiv]. Therefore all companies are intended to assess and improve the process of software development by detecting the fault at earlier stages. It will not only enhance the implementation process but quality as well. The result is enhancement in the capability of a company to meet the demand of software market.

Measuring quality of a software product is crucial in order to get valuable results in software systems that are efficient, reliable, understandable and acceptable for their stakeholders [xv]. Similarly, it is also necessary to develop and utilize rigorous assessment models and mechanisms in order to facilitate and ensure the continuous quality of software application systems [xvi].Thus quality of software should be addressed continuously and resolve the related issues on regular basis. The solution to obtain this goal is use of effective software management with the collaboration of software metrics. Still development of software is a complex task. Currently there are limited numbers of well-defined and reliable measuring techniques for evaluation process of software development [xvii]. The software quality metrics are no more exception now a days. Therefore, one of the major objectives of this research is to demonstrate the effect of software development process using quality metrics on the produced software having quality. Its special aspect is to achieve customer satisfaction. Furthermore, this study contributes in a fashion by providing a roadmap to the software project managers to develop software products within allocated time and cost without compromising on quality. In order to cope with the above mentioned factors, a systematic procedure has been followed in this work according to the need of quality metrics and ultimate result is the improved quality of the developed software products.

This paper is organized as follows. Section 2 encapsulates the review of literature briefly explaining the software quality and software quality metrics; section 3 describes our research design. Analysis of the developed products applying various quality metrics are elaborated in section 4. In section 5a conclusion is drawn on the basis of the results obtained from experimentation.

II. Literature Review

It is difficult to find a single and comprehensive definition of software quality from the existing literature. According to Pressman [xviii]*"a product's quality is a function of how much it changes the world for the better."* Moreover, IEEE Standard Glossary of Software Engineering terminology defines the quality as, *"The degree to which a system, component, or process meets specified requirements"*, IEEE further explain quality as *"The degree to which a system, component, or process meets customer or user needs or expectations"*. It is urged by [xix] that quality refers to the extent or degree to which a customer's requirement is met. However, it is advocated by [xx] quality of a software product can be measured as internal quality or external quality. Internal quality is gauged by software professionals whereas external quality can be achieved by the users' satisfaction. It is clear from these definitions that overall quality of software refers to the measurement of various properties of software ranging from requirement gathering to implementation. Whereas it is insisted by [xiii] that actual quality can be achieved by implementing related software quality metrics.

Like Physics, software metrics have roots in ancient discipline of measurement established by scientists. Their purpose is to define basic rules to measure the various parameter accurately. On this basis, software engineers have adopted the principles of measurement in order to measure various software activities [xxi] as quality metric provides a numerical value that can be scaled to measure a quality factor [xx]. It is urged by [xxii] that metrics can be defined as a ratio of, or relationship between two measures that finally reaches to the real information status. However, according to [xvii] "software metrics deals with the

measurement of the software product and process by which it is developed". Some of the common metrics include Lines of Code (LOC), Volume (VOL) of the product, functions implemented in terms of Function Points (FP), complexity in terms of Cyclomatic Complexity (V(G)) of software, maintainability, cost, testability, customer's satisfaction, defect counts, crude effort (in person months), correctness and accuracy [xxiii]. Generally simple metrics are accepted while the complex metrics are rejected as it is comparatively easier to deploy the simple ones. Additional detail regarding working and computation of the various metrics like FP, V(G), VOL and etc. can be obtained from [xvii], [xxi], [xxiv] and [xxv].

Software metrics can be termed as a set of measurement activities in software engineering. These activities are performed during procedure of software development to characterize the properties of software code. Moreover scope of these activities cover the monitoring and recording the defects and bugs at any stage of development. Then result of characterization and monitoring is presented in meaningful information (combination of data and dimension) like LOC or FP or Critical Errors (CE). This information enables decision making easier for management and hence leads to better quality. Another consequence of using metrics is the proper schedule of software development that can be achieved on the basis concluding information.

Above discussion helps to conclude that various characteristics and aspects of a software product can measured by utilizing the software metrics. The cost estimation, size, complexity, maintainability, quality and so on are main attributes. Beside this, role of software metrics to complete the product within scheduled time, allocated budget and available resources cannot be ignored.

III. Research Design

There are three stages in the implementation of this research work in which software metrics are applied on the specific group of student projects. In first stage, groups of the sampled population are formed. Students are divided into two groups. In the second stage, similarity conditions are defined for both groups in order to fair comparison of results. In last stage, finding and outcome gathered, against deployment of software metrics, during different phases of SDLC are observed for conclusion.

A. Sample

The sample of this study is consisted of seventeen students from one of the public sector universities of Pakistan. Selected population is from the domain of computer science having similar academic background and all are working on their final year software development projects. These students have been granted one year time to complete their projects.These students are selected using Stratified Random Sampling technique to form two groups. The name given to first group is conventional group that is comprises of nine students. Rest of eight students are the members of controlled group Controlled group is kept under observation in a controlled environment providing proper guidance during whole SDLC process of their software development projects. The controlled group students are properly guided about quality metrics, deployment procedure of metrics and related measurement by providing valuable lectures and research articles. So that they may understand the significance of the task for which they are selected. On the other hand, second group was allowed to work as per routine without providing any additional guidance and/or knowledge about metrics and quality assurance. This group has been named as conventional group.

B. Similarity Conditions

Following similarity conditions have been formulated in order to compare the outcome of the study;

a) All the targeted respondents were from same level of education.
b) All the respondents were working on their final year software development projects.
c) All of the respondents in controlled group were restricted to follow same software process model.
d) All were instructed to develop their products using same development tool i.e. VB.Net
e) Both groups were provided same programming environment, lab facilities and other required resources.
f) Both groups were instructed to develop database systems in order to perceive the consistency.

C. Statistical Tool

Two independent samples t-test have been applied on the data collected from experimentation in order to verify the difference between various quality measures between conventional and controlled groups. The two-sample t-test is used to determine whether two population means are equalor not. Further detail regarding implication of two-sample-t-test can be obtained from [xxvi] and [xxvii].

IV. Analysis of the Software Products

The software metrics such as Complexity (V(G)), LOC, Volume (VOL) and FP have been deployed in the controlled group in order to improve the quality of the developed software product. Results obtained are elaborated in Table I for conventional group and Table II for controlled group respectively. However, comparison of different characteristics such as size, V(G), VOL is discussed in Section 5.

TABLE I
VARIOUS MEASURES OF CONVENTIONAL GROUP

No.	LOC	V(G)	FP	Volume	LOC/FP (Aprox.)
1.	5889	4	986	15453	6
2.	2051	4	348	13834	6
3.	8676	5	1084	11564	8
4.	20333	5	2947	34219	7
5.	3503	5	902	14453	4
6.	4700	5	1055	13625	4
7.	3922	3	975	8379	4
8.	4950	4	780	14845	6

V. RESULTS AND DISCUSSION

There are two major measures complexity and size of software product which impact on the quality. These are concentrated and discussed here. The size of the software product is explored in terms of LOC and VOL. The hierarchy has been shown in Fig. 1.

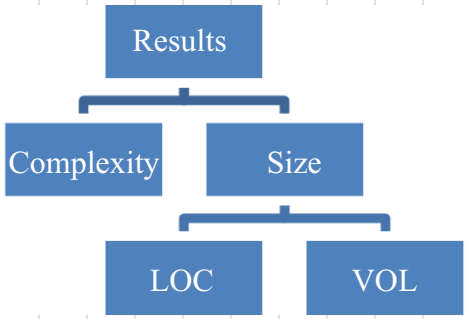

Fig. 1. Dimensions of results

Different hypotheses (H_0 and H_1) have been formulated for each measure. The formulated hypotheses and computed values using two-sample *t– test* are illustrated in Table III. It is pertinent to mention that the significant value used is p-value for all the cases. Significant means the computed results are indicating that implication of quality metrics on the products of controlled group is better than conventional group. Computed results are indicating that implication of quality metrics on the products of controlled group is better than conventional group.

TABLE III
FORMULATED HYPOTHESES AND COMPUTED VALUES USING TWO SAMPLE T-TEST FOR BOTH GROUPS.

	H_0	H_1	t-value	p-value
V(G)	$\mu_1 \leq \mu_2$	$\mu_1 > \mu_2$	6.23	0.000
LOC	$\mu_1 \geq \mu_2$	$\mu_1 < \mu_2$	0.93	0.184
FP	$\mu_1 = \mu_2$	$\mu_1 \neq \mu_2$	-0.12	0.548
VOL	$\mu_1 \leq \mu_2$	$\mu_1 > \mu_2$	1.84	0.043
LOC/FP	$\mu_1 \leq \mu_2$	$\mu_1 > \mu_2$	2.00	0.032

* μ_1 = Average of the **conventional** group and μ_2= Average of the **controlled** group

A. *Complexity*

Complexity is one of the crucial factors in the development of any product. However, its importance increases more in the field of software development due to complex nature of software itself. The complexity V(G) of projects is computed for both conventional and controlled groups and result is shown in Table I and Table II respectively. The two-sample *t–test* is applied on results for statistical analysis.

TABLE II
VARIOUS MEASURES OF CONTROLLED GROUP

No.	LOC	V(G)	FP	Volume	LOC/FP (Aprox.)
9.	1600	2	396	4732	4
10.	4100	3	1075	10678	4
11.	2700	2	537	9887	5
12.	6920	3	2419	12344	3
13.	8100	2	2100	15244	4
14.	4900	2	782	11745	6
15.	3827	2	1009	7296	4
16.	6558	3	1338	13456	5
17.	4805	3	942	10043	5

According to Table III, the p-value is less than 0.05, so reject H_0concluding that the products developed by conventional group are more complex than controlled group. Hence, the value of complexity is decreased by deploying software quality metrics. Comparison of complexity of both groups is illustrated in Fig. 2.It shows that complexity for controlled group is lower comparatively that is a positive impact on software product. A complex software product needs larger size, greater development time, and more cost for maintainability, troubleshooting and debugging.

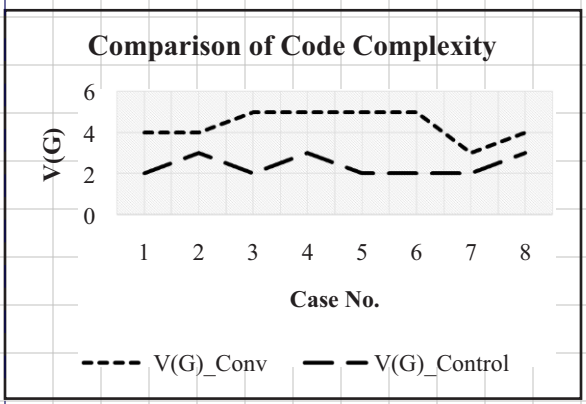

Fig. 2. Comparison of code complexity of both groups

1) Impact of Complexity

Software complexity affects the maintenance and modification of the projects require more time, increase in cost, and result in more errors [xxviii]. Hence, complexity is directly proportional to various quality factors of a software product. These factors include

maintainability, development time, debugging, size, cost etc.

a) Maintainability

Maintainability describes the procedure and required efforts to maintain a product. Software maintenance is one of the most vital phases of SDLC because maintenance cost of a software product is around 40-80% of its total development cost [xxix]. Complex nature of software products needs 60% of maintenance expenses to enhance its existing functionality. It means that maintainability directly depends on complexity of software products. Therefore, it is extremely important to formulate appropriate software artifacts during each phase of SDLC in order to reduce the maintenance cost.

b) Debugging

Debugging is an expensive, time consuming and continuous procedure of the software engineering. Moreover, it is a complex activity to model in real-world projects [xxx]. The software having complex nature are more difficult to understand, debug and troubleshoot. It is to be noted that debugging is twofold harder than that of writing the code for the first time [xxxi]. Finding and fixing bugs become tougher as the software becomes more complex. Reducing the complexity of the software product consequently results in reducing the debugging efforts.

c) Cost

Cost is one of the most important factors for any product. As cost of software product is concerned, it is measured in terms of developing time, maintenance, debugging and so on. It is directly related to the complexity of the software product. According to [xxxii] complexity is broadly esteemed an important determinant of software maintenance costs. It means that a software product becomes expensive as complexity increases. Thus use of deploying metrics makes the developing procedure systematic that reduces the all kinds of cost.

B. Size

Software projects grow continuously not only during SDLC but also after implementation or delivery. Therefore, the size of a software product cannot be a constant. The overall size of the software products in terms of LOC, VOL and FP for both groups has already been computed as mentioned in Table I and Table II respectively.

1) LOC

In order to assess the size of the developed products by both groups, formulated hypothesis is already mentioned above in Table III. According to Table III, the p-value > 0.05 against LOC, so accept H_0 concluding that overall lines of codeof the products developed by conventional group are greater than counterpart. Hence, the value of LOC has been reduced (as shown in Fig. 3) by applying appropriate quality metrics during different phases of SDLC. Decrease in the size of the product results in reducing the efforts and time to debug the software product.

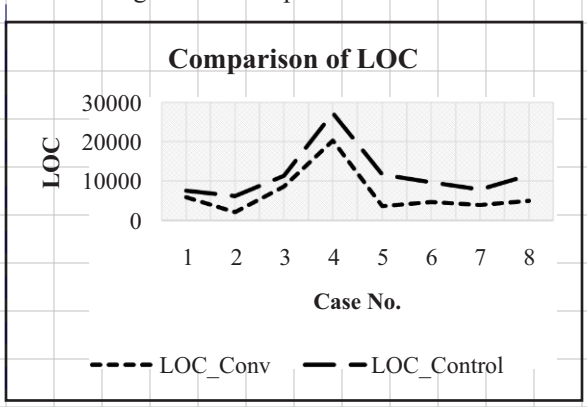

Fig. 3. Comparison of LOC of two groups

2) Volume

Volume contributes to the size of software products. The computed volume of students' project has been plotted in Fig. 4. Result shows that volume against controlled group has reduced value. According to Table III, the p-value < 0.05 for VOL, so accept H_1 concluding that overall VOL of the products developed by conventional group are greater than controlled group. Hence, the value of VOL has been reduced (as shown in Fig. 4) by applying appropriate quality metrics during different phases of SDLC. Decrease in the VOL of the product consequently reduces the efforts and time to debug the software product.

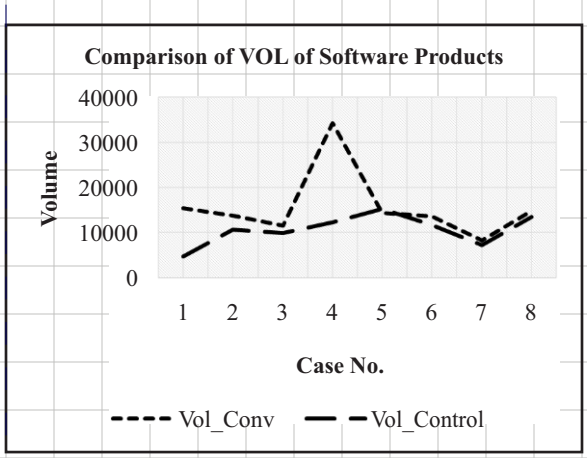

Fig. 4. Comparison of VOL of both groups

3) Function Points

Function point metrics are one of the most accurate and effective metrics developed to gauge the size of a software product. Moreover, cost, software productivity, quality, costs and risks can also be measured using FP. Function points implemented by both groups can be observed from Table I and Table II.

Similarly, according to Table III, the p-value > 0.05, hence accept H_0 concluding that function points implemented by both of the groups are equal. One of the possible cause can be that all products are small in size as large products can have above 10,000 FPs. The comparison of function points implemented by both groups is shown in Fig. 5.

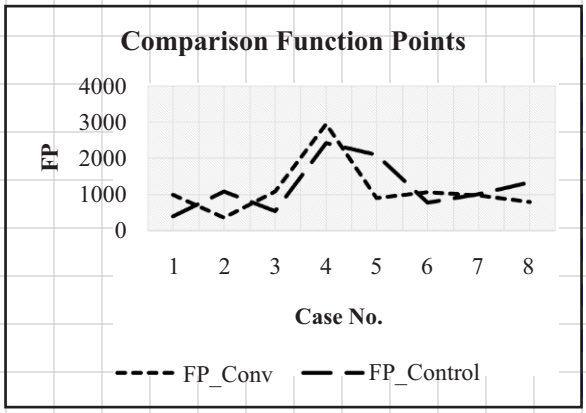

Fig. 5. Comparison of FP of both groups

VI. Limitations

There are certain limitations regarding deployment of software quality metrics on the targeted products. This research focused only on conventional software metrics. One of the major reasons behind adopting the conventional metrics is that the software products have been developed utilizing traditional/conventional software engineering process model(s). Therefore, further investigations are recommended to consider by employing object oriented quality metrics on the software products. Furthermore, this study limits the participants from one of the HEI of Pakistan. Nonetheless, this work can be enhanced by conducting an experiment on the software products developed by the small and medium software developing organizations of the town.

VIII. Conclusions

Quality is difficult to measure due to its variety of dimensions but it is crucial for a software product to be viable. To achieve a quality product, it is vital for software engineers to utilize the appropriate software quality metrics during each phase of SDLC. Unfortunately user satisfaction is the only mark of quality in the practice of software engineers and organizations which are developing software products in localized environment of Pakistan. However, quality of a software product cannot be measured on the basis of customers' satisfaction only. The efficacy of software quality metrics has been examined on different software products in variety of dimensions like complexity, maintainability, function points and size.

Seventeen students are selected for this research and are divided into two groups. One group is called controlled and other has been named as conventional as discussed in Section 3. Different quality metrics have been applied on the software products developed by controlled group during different phases of SDLC. On the other hand, conventional group has developed software without deploying software metrics. Observing Table I and II show findings of this work which are presented graphically in Fig. 2 to 5. It is crystal clear that controlled group has produced better quality of software products as compare to conventional group. The results of the experimentation have highlighted that complexity of the software is one of the most crucial factor for quality of software which can be reduced by deploying appropriate quality metrics during various phases of SDLC. There is a no hesitation to conclude that utilization of software metrics has at least two major achievements. One is the enhancement in the quality of software products. Secondly it helps software executives to complete the software projects within specified time, allocated budget and available resources.

References

[i] W. Jr, A. William and V. Buvaneswari,"Some observations on software quality", Proceedings of the 37th annual Southeast regional conference (CD-ROM). 1999. ACM.

[ii] N. A. Maleh, C. S. Lee, C. K. Ho and H. R. Chong, "A conceptual framework for enhancing the instructional design process",Malaysian Online Journal of Instructional Technology (MOJIT)2004.

[iii] S. N. Bhatti, "Why quality?: ISO 9126 software quality metrics (Functionality) support by UML suite", ACM SIGSOFT Software Engineering Notes, 2005. **30**(2): p. 1-5.

[iv] D. Jamwal, "Analysis of Software Quality Models for Organizations. International Journal of Latest Trends" in Computing (E-ISSN: 2045-5364), 2010. **19**.

[v] M. H. N. Nasir, and S. Sahibuddin, "Critical success factors for software projects: A comparative study", Scientific research and essays, 2011. **6**(10): p. 2174-2186.

[vi] S. R. Schach, "Object-oriented and classical software engineering". Vol. 6. 2007.

[vii] M. W. Sumanand M. Rohtak, "A Comparative Study of Software Quality Models", International Journal of Computer Science and Information Technologies, 2014. **5**(4): p. 5634-5638.

[viii] B. Wong, S. Chulani, J. Verner and B. Boehm, "Second Workshop on Software Quality", in Proceedings of the 26th International

Conference on Software Engineering. 2004. IEEE Computer Society.
[ix] D. Nabil, A. Mosad, and H.A. Hefny,"Web-Based Applications quality factors: A survey and a proposed conceptual model", Egyptian Informatics Journal, 2011. **12**(3): p. 211-217.
[x] R. E. A. Qutaish, "Quality models in software engineering literature: an analytical and comparative study"Journal of American Science, 2010. **6**(3): p. 166-175.
[xi] A. Hassanzadeh, F. Kanaani, and S. Elahi, "A model for measuring e-learning systems success in universities", Expert Systems with Applications, 2012. **39**(12): p. 10959-10966.
[xii] B. Kitchenham, "The Failure of Quality. in Proceedings of the Second Workshop on Software Quality", ICSE 2004. 2004.
[xiii] B. Boehm, S. Chulani, J. Verner and B. Wong,"Fifth Workshop on Software Quality", in Companion to the proceedings of the 29th International Conference on Software Engineering. 2007. IEEE Computer Society.
[xiv] M. Sulayman, C. Urquhart, E. Mendes and S. Seidel, "Software process improvement success factors for small and medium Web companies: A qualitative study", Information and Software Technology, 2012. **54**(5): p. 479-500.
[xv] I. ISO, "IEC 9126-1: Software Engineering-Product Quality-Part 1: Quality Model", Geneva, Switzerland: International Organization for Standardization, 2001.
[xvi] S. Mavromoustakos and A. S. Andreou, "WAQE: a web application quality evaluation model", International Journal of web engineering and technology, 2007. **3**(1): p. 96-120.
[xvii] E. E. Mills and K. H. Shingler, "Software Metrics-SEI Curriculum Module", SEI-CM-12-1.1. 1988.
[xviii] R. S. Pressman, "Software engineering: a practitioner's approach", McGraw Hill International Edition, 2005: p. 466-472.
[xix] B. Aziri, "Managing Quality: With Special Emphasizes on SME's in The Pollog Region", Publications in International Scientific Publications: Economy & Business Journal, 2015. **9**(1): p. 337-342.
[xx] R. S. Jamwal, D. Jamwal and D. Padha, "Comparative Analysis of Different Software Quality Models", in 3rd National Conference. 2009.
[xxi] S. Alexandre, "Software Metrics: An Overview (Version 1.0)", CETIC asbl—University of Namur, Software Quality Lab, Belgium, 2002.
[xxii] M. H. Halstead, "Elements of software science", Vol. 7. 1977: Elsevier New York.
[xxiii] M. L. Cook, "Software metrics: an introduction and annotated bibliography", ACM SIGSOFT Software Engineering Notes, 1982. **7**(2): p. 41-60.
[xxiv] S. R. Schach, "Object-oriented and classical software engineering (Vol. 8)", McGraw-Hill New York, 2011.
[xxv] S. H. Kan, "Metrics and models in software quality engineering", Addison-Wesley Longman Publishing Co., Inc. 2002.
[xxvi] J. H. Zar, "Biostatistical analysis", 1999: Pearson Education India.
[xxvii]J. Romano, "Testing statistical hypotheses", 2005, Springer Berlin.
[xxviii]R. D. Banker, S.M. Datar and D. Zweig, "Software complexity and maintainability Age", 1989. **11**(5.6): p. 3.
[xxix] A. M. F. Sáez, M. Genero, D. Caivano and M. R. V. Chaudron, "Does the level of detail of UML diagrams affect the maintainability of source code?: a family of experiments", Empirical Software Engineering, 2016. **21**(1): p. 212-259.
[xxx] M. Cinque, "On the impact of debugging on software reliability growth analysis: a case study", in Computational Science and Its Applications–ICCSA 2014. 2014, Springer. p. 461-475.
[xxxi] U. Software, "Increasing software development productivity with reversible debugging", 2014: Cambridge, United Kingdom.
[xxxii]B. W. Boehm, "Software engineering economics", Vol. 197. 1981: Prentice-hall Englewood Cliffs (NJ).