Review of an Efficient Software Metric Parameters for Component-Based Software Engineering

Iyyappan. M[1], Sultan Ahmad*[2, 3], Jabeen Nazeer[2], A.E.M. Eljialy[4], and Shoney Sebastian[5]

[1]Department of Information and Communication Technology, Adani University, Ahmedabad, Gujarat, India
[2*]Department of Computer Science, College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia
[3]University Center for Research and Development (UCRD), Department of Computer Science and Engineering, Chandigarh University, Gharuan, Mohali 140413, Punjab, India
[4]Department of Information Systems, College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University, Al-Kharj, 11942, Saudi Arabia
[5]Department of Computer Science, Christ (Deemed to be University), Bangalore-29, India
*Corresponding Author: Sultan Ahmad (*s.alisher@psau.edu.sa*)

Summary
Large software systems are being produced with a noticeably higher level of quality with component-based software engineering (CBSE), which places a strong emphasis on breaking down engineered systems into logical or functional components with clearly defined interfaces for inter-component communication. The component-based software engineering is applicable for the commercial products of open-source software. Software metrics play a major role in application development which improves the quantitative measurement of analyzing, scheduling, and reiterating the software module. This methodology will provide an improved result in the process, of better quality and higher usage of software development. The major concern is about the software complexity which is focused on the development and deployment of software. Software metrics will provide an accurate result of software quality, risk, reliability, functionality, and reusability of the component. The proposed metrics are used to assess many aspects of the process, including efficiency, reusability, product interaction, and process complexity. The details description of the various software quality metrics that may be found in the literature on software engineering. In this study, it is explored the advantages and disadvantages of the various software metrics. The topic of component-based software engineering is discussed in this paper along with metrics for software quality, object-oriented metrics, and improved performance.

Keywords:
Complexity metrics; object-oriented metrics; size metrics; software metrics; software quality; software metrics; Component Based Software Engineering(CBSE).

1. INTRODUCTION

A method known as CBSE is used to expand the planning process and incorporate more components into computer-based systems [1]. It offers a method for building massive software systems. The development phase primarily uses internal and commercially available components. The software systems of today are exceedingly complicated, voluminous, and unwieldy. As a result, there is decreased productivity, increased risk management, and poor software quality. Software metrics, which track numerous complexity facets, are crucial for evaluating and enhancing software quality [2]. Metrics offer important information for external quality elements including reliability, reusability, and maintainability [3]. Metrics assist in giving the system data and improving the system's quality. Software metrics will reduce the subjectivity of faults during the assessment of Software quality[20]. Metrics are the numerical value of software and it is used to predict the fault [21][27]. When it comes to business systems that need to retrieve a lot of data, it is really helpful. The software's quality is contained in the object-oriented design. The management of both large- and small-scale projects was handled via system objects associated to specific attributes or qualities [4]. The classification of an object's design includes its reusability, dependability, decomposition, and adaptability [5]. Metrics are described as "a quantifiable measure of the extent to which a component, system, or a given attribute" [6] in accordance with IEEE standards. The qualities of quality metrics include: [7] correctness, dependability, formality, implementability, minimal value, and orthogonality. System requires better quality, greater performance, and safety. The usage of software metrics could be used to accomplish this. By examining the elements on a component-based system, these metrics are primarily used to manage risk and improve quality. The measurements are used in software development and deployment methods. The primary metrics process in the CBSE identifies the likely risks and necessary corrective measures. Component-based metrics' primary purposes are

Manuscript received November 5, 2022
Manuscript revised November 20, 2022
https://doi.org/10.22937/IJCSNS.2022.22.11.108

to provide reusability, cut costs, and speed up development.

2. BACKGROUND STUDY

Many works related to CBSE quality metrics have been published. The related papers that are relevant to our paper are discussed as follows selectively.

Zhiqiao Wu.et.al Proposed the methodology for reduce the cost and increase the reliability of reuse models [8]. Iyyappan.et.al discussed about the coupling and cohesion metrics component and procedure followed for these metrics [9]. Miguel Goulao.et al. [10] provided a method for metric formalisation based on the usage of formal specification languages and ontologies.

Jianguo Chen .et al. [11] suggested a formal direct and indirect component coupling metric for both individual component and assembly components. P.K. Suri .et al. [12] provided measures for assessing the component's independence for reusability. For evaluation, the chi-square test has been used.

V.Lakshmi Narasimhanet .et.al. [13] detailed a methodical comparison of three sets of data, allowing the user to select the one that is most appropriate for their needs. P.Edith Linda .et al. [14] performed comparisons between different algorithms based on how well they perform and how much memory they use.

Abhikriti Narwal .et.al. [15] outlined the complexity metric for software parts based on interface techniques. Sidhu Pravneet .et.al. [16] proposed a method for measuring a software component's quality objectively using values and ratios. To determine the precise quality of the component in the metrics, a back propagation approach based on artificial intelligence is frequently utilised.

Hesham Abandahet .et. al. [17] presented the effectiveness and power of call graph based metrics by evaluating the many categories of bugs. Taranjeet Kauret .et.al. [18] made comparison of various lack of cohesion metrics to increase the fault prediction power and to decrease the complexity.

Divya Chaudharyet .et. al. [19] defined various management metrics, requirement metrics, and complexity metrics focusing on various attributes such as cost, quality and productivity. Ermiyas Birihanu Belachew et al[20] has identified Software quality(Correctness, Product quality, Scalibiity, Completeness and absence of bugs) is a means of meansuring how software is designed and how well the software confirms the design.

Ming-Chang Lee[25] has defined a few software metrics in the design factors and addressed a number of software quality assurance models as well as the use of some quality factors measurement methods in the quality life cycle. There are one or more QA measure metrics for each activity in the software life cycle that are focused on guaranteeing the quality of the process and finished product[26].

3. METRICS TAXONOMY

The taxonomy includes a set of qualitative behavior and quantitative evaluations, for the scale of the project depend on the Line of code (LOC) [28][29]. The quantitative evaluation enforces the desired comparability of proposals. The taxonomy's characteristics are as follows:

• ***Scope* -** This refers to the level of granularity and type of artifacts that are the objectives of the metrics- based on the evaluations. Some components are evaluated in white box and rest in the black box. So coarse and fine-grained component are totally differentiate with one another characteristics.

• **Intent -**Various approach and their functionality used to achieve those objectives in the domain.

• **Technique -** It's used for the explanation and verification of the metrics process. It follows the formal metrics definition on this technique.

• **Critique -** A qualitative assessment of the most important features of the proposal, including its most motivating aspects, as well as its main limitations also provided.
• **Maturity –** It follow the comparison framework on the basis characteristic. It follows four different viewpoints: quality replica, mapping among metrics and quality model, description about metrics procedure [2]. how users and providers' security authentication is performed).

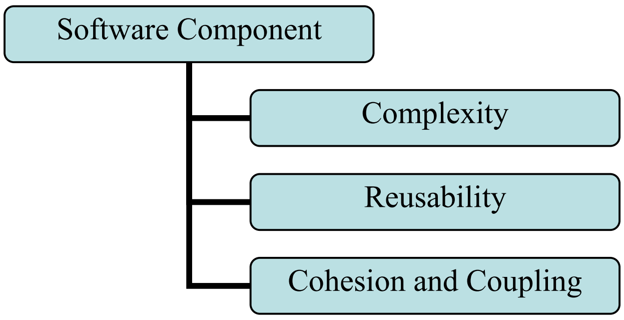

Fig 1: Software Components Classification

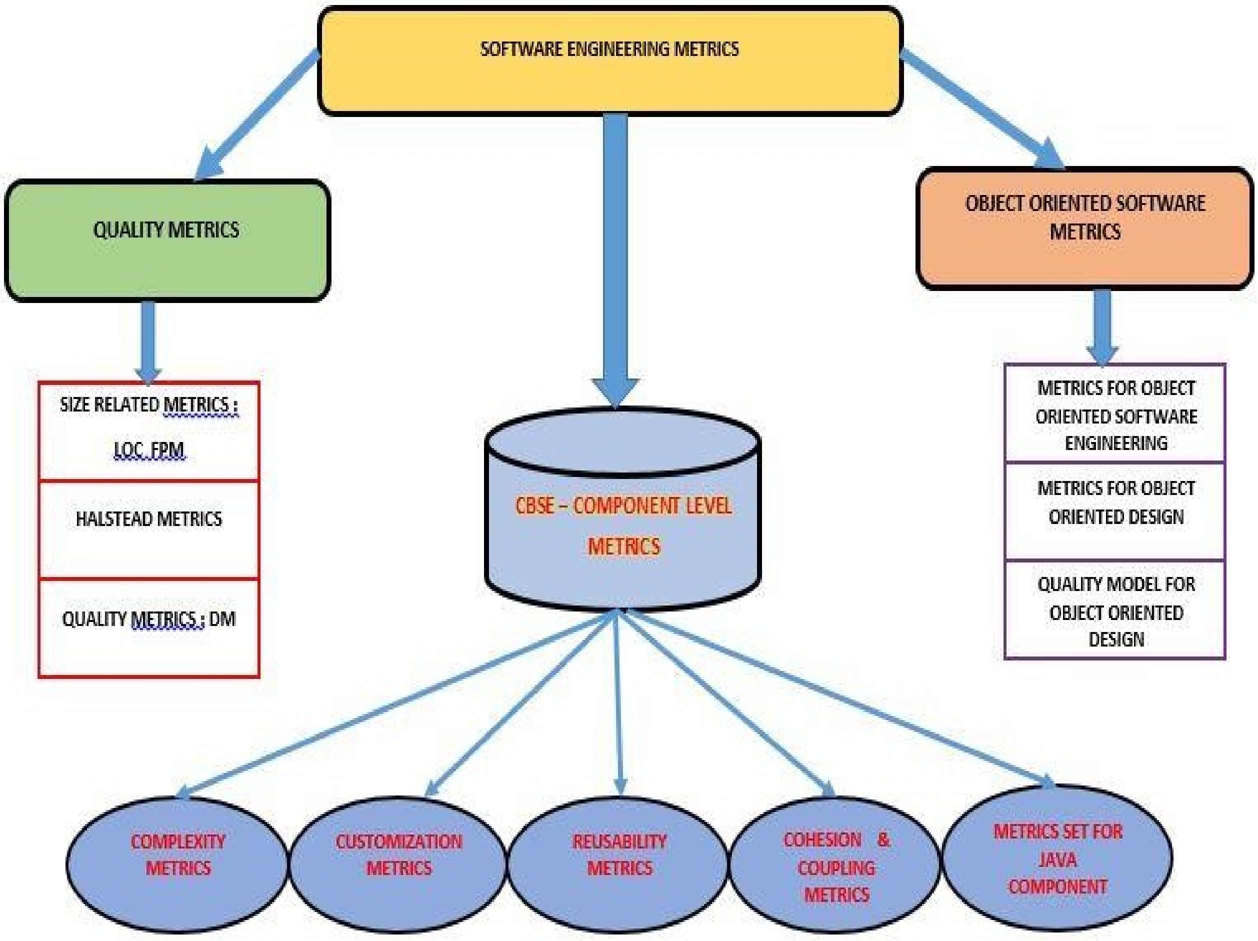
SOFTWARE ENGINEERING METRICS
QUALITY METRICS
OBJECT ORIENTED SOFTWARE METRICS
SIZE RELATED METRICS : LOC, FPM
HALSTEAD METRICS
QUALITY METRICS : DM
CBSE – COMPONENT LEVEL METRICS
METRICS FOR OBJECT ORIENTED SOFTWARE ENGINEERING
METRICS FOR OBJECT ORIENTED DESIGN
QUALITY MODEL FOR OBJECT ORIENTED DESIGN
COMPLEXITY METRICS
CUSTOMIZATION METRICS
REUSABILITY METRICS
COHESION & COUPLING METRICS
METRICS SET FOR JAVA COMPONENT

Fig 2: Classification CBSE metrics, Quality Metrics, Object Oriented Metrics

4. COMPONENT METRICS

Component level metrics are used for the measurement of qualities in the terms of their complexity, customizability, and reusability.

A. Component Complexity Metric:

Component complexity metrics can be segregated into four different methods a Plain, Static, Dynamic and Cyclomatic. The plain metric contains the various elements such as classes, abstract classes, and interface. The static metric determines the weighted sum of various types of relationships in a component. Complexity of message passing through also present in internal component is called dynamic. The Cyclomatic metric present after the implementation in the design stage.

B. Component Customization Metric:

On the interface level it follows the various methods. This metric component will extent of methods for customization in the interface.

C. Component Reusability Metric

In the development process we can measure how much reusable component are used in the design phase. It can be calculated using the ratio of the interface component between sums of the interface component. The metrics also provide the adaptability, compose-ability and flexibility of software.

D. JAVA Components using metrics

It's used to measure the black box component. The five metrics for measuring the reusability of a software component: existence of meta-information, component observability, customizability of component, and external dependency.

Available Primary Data - It is a binary metric value also collect the information from the existing Meta-data. It enhances the understandability of a component.

Component Observability – It analyses the number of fields available on the class implementation on the java component. The rate component observability contains readable properties, if class without fields it follows value should be zero. It's easy to understand the component from the external viewpoint.

Component Customizability - This method contains the writable properties to the java component to check their number of fields.

Component's without Return - It has been observed the percentage of business methods without any return value from all business methods implemented within a component. High value of the metric indicates a low level of external dependency of the component which results in ease of portability.

Component parameter - It is the percentage of business methods without any parameters from all business methods. External dependency of a component are used measure in the Self-completeness of parameter. The metrics are combined by the concept of a reusability model which consider that reusability also focuses on understand ability, interoperability, adaptability.

E. Component Cohesion and Coupling Metrics

This metrics mention about the component of high cohesion and low coupling. The cohesion metric takes in to the consideration of structural relationships is called as the classes of an object oriented component. Coupling between the classes Cm, Cn, it also mentions about the weighted sum of different types of method and modules of classes. It considers sum of coupling among all pairs of classes.

F. Contextual Reusability Metric

Adding the internal attributes to the component reusability process. Observance metric methods are used in this process: Architecture and Componentry consumer are provided by the cloud service provider.

5. SOFTWARE QUALITY METRICS

These are some of the software quality metrics that are used in the software development process:

A. Size related metrics

It used to measure size of the software. Either it should be small scale or large scale project.

Line of code (LOC): This metrics method mainly used to evaluate the module size in the software quality. It is related to the source code evaluations [4].

Function point metrics: Function point metrics is the type of metrics which is used to evaluate the line of code only when the accessibility of code is present and so that it cannot be used in early stage. There is a method to resolve the evaluation of software size early in the enlargement of life cycle which was proposed by Albrecht. This mainly depends on inquiries, input of the user, output of the user and the values expected to measure the value in evaluating the size of the program and thus intention which is needed for the development.

B. Halstead metrics:

It was proposed by Halstead fort the purpose software quality assessment. Halstead's measure of module conciseness is calculated using the basic premise that a well-structured program is merely a function of its specific operators and operands. In this method to totally find the software production effort which consists of some length (N), volume (V) and vocabulary (n).

These software Characterization formulas were developed by Halstead[22] .

The measure of vocabulary: $n = n1 + n2$

Program length: $N = N1 + N2$

Program volume: $V = N\log n$

Program level: $L = V^* / V$

Where

n1 = the number of unique operators

n2 = the number of unique operand

N1= the total number of operators

N2 = the total number of operands

C. McCabe's Cyclomatic complexity metrics:

McCabe [23] has proposed a complexity metric on mathematical graph theory.

The maximum number of "linearly independent" paths through a program serves as a proxy for the complexity of that program as determined by its control structure. This measurement can be used by software developers to identify whether program modules are excessively complex and require re-coding. The detail can be seen in Table. I.

TABLE 1 THE FORMULAS OF MCCABE'S CYCLOMATIC COMPLEXITY METRICS [25]

Software Metrics	Software quality factors	Formulas
Cyclomatic complexity V(G)	Complexity Maintainability Number of bugs Modularity Simplicity Reliability Testability Understandability	V(G)=e-n+2p
Essential complexity EV(G)	Complexity Conciseness Efficiency Simplicity	EV(G)=V(G)-m

D. Henry and Kafura's Information Metrics

Information flow complexity (IFC) [24] describes the amount of information which flows into and out of a procedure. This metrics use the flow between procedures to dhow the data flow complexity of a program. The Formula is:

IFC = Length´ (fan - in * fan - out)2

Where

Fan-in: The number of local flows into a procedure plus the number of global data structures from which a procedure retrieves information.

Fan-out: The number of local flows into a procedure plus the number of global data structures from which a procedure updates.

Length: It is the number of lines of source code in the procedure. In implementing this count, embedded comments are also counted, but not comments preceding the beginning of the executable code.

E. Quality metrics

In this quality metrics can be identified with the help of fault or failure metrics. The code inspections and the program test number are used to identify the error in this metrics.

6. OBJECT ORIENTED SOFTWARE METRICS

A. Metrics for object-oriented software engineering (MOOSE)

Similar metrics which inaugurated expressive amount of interest. Presently it suite for the evaluation of object-oriented software proposed by Chidamber and Kemerer (CK).

B. Depth of inheritance tree (DIT)

Here a tree exists with root node and leaf nodes. This metric used to measure the longest path from the tree. The class behaviour and the design complexity of potential reuse and class. Inheritance layers are very difficult to understand. Thus the tree with deeper hierarchy defines the reuse of inherited methods.

(a). Number of children (NOC): It's in the form of class hierarchy also hold the sub class and super class. The number of subclass increase the comparison of number of super class.

(b). Response for class (RFC): It defines the response set also evaluates set of methods are available in this metrics. Two different methods are followed executed in response and messages receive from the object. The value which is large also complicates debugging of the object and even testing where it needs the particular tester to be have the functionality knowledge. If the RFC value is very large it will be taking complex class which is a worst case scenario, and the RFC value will be helped in assuming the time required for the assessment testing.

C. Metrics for object-oriented design (MOOD)

It holds the functionality of inheritance, message passing, polymorphism and encapsulation. Each of the object oriented design metrics were expressed to measure. The actual use any feature of a particular design is defined by the numerator. Attributes and methods are the two main features of the MOOD metrics. The object in the system is like an attributes also methods are used to modify or maintain the objects [9]. Metrics of object-oriented design are defined as follows:

(a). Attribute hiding: The ratio of sum of all attributes are invisible in all classes. Along with entire classes and attribute are under this factor consideration.

(b). Method inheritance: Inheritance of class methods is to calculate the ratio of total number of the object are inherited [2].

(c). Attribute method: This method of inheritance factors are described about the total sum of attributes inherited in the classes, also number of attributes available on the system [9].

(d). Polymorphism factor (PF): It defines the number of available different polymorphic situation. In the method of attribute inheritance and method inheritance are used to evaluate the class inheritance and to give the property of similarity in to the classes.

D. Quality model for object-oriented design (QMOOD)

This type of the quality model which is comprehensive and inaugurates a precisely defined and imperiously a validated model. To determine the quality attributes of design such as reusable method and understandable method. Some mathematical formulas with structural Object oriented design characteristics like coupling technique and encapsulation method. The quality model contains of six equations which originate a relationship between the six quality attributes of the object-oriented design and the properties are: Functionality, effectiveness, extendibility, reusability, understand ability, flexibility and eleven design type characteristics.

7. CONCLUSION

With the rapid development in software industry, the measurement of the software product becomes more complex thus increasing the necessity of better software metrics are required on the time. CBSE is the widely used concept in the software industry development and innovative research phase. Metrics are used for separating the characteristics of the component. Metrics mainly focus on selecting the suitable reusable components and analyses the functions which can perform properly. Metrics provided the data to the system and increasing the quality of system. Metrics are mainly used for managing risk in the system. In this research work, we can understand how various metrics are used in CB development also that concentrate on the factors like complexity, size, reliability, reusability, understandability, maintainability etc. A systematic solution and environment helps the automatic system level measurement.
The following are key points that are concluded

- Component characterization follow those steps: increase understanding of architecture, improve the usage of component, better retrieval, performance cataloguing used for reusability.
- Mostly metrics suggested for CBSE has been defined on the basis of theoretical considerations. However, the practical paradigm should be considered and theory must be validated.

ACKNOWLEDGMENTS

The authors would like to thank the Deanship of Scientific Research at Prince Sattam Bin Abdulaziz University, Alkharj, Saudi Arabia for the assistance.

References

[1] R. S. Pressman, “Software Engineering, A Practitioner’s Approach”, Sixth Edition, Mc Graw. Hill, 2005.

[2] Divya Chaudhary, Prof. Rajender Singh Chillar, “Component Base Software Engineering Systems: Process and Metrics”, International Journal of Advanced Research in Computer Science and Software Engineering, July 2013, Vol. 3, Issue 7, pp. 91-95.

[3] Gurdev Singh, Dilbag Singh, Vikram Singh, “A Study of Software Metrics”, International Journal of Computational Engineering & Management, Jan 2011, Vol. 11, pp. 22-27.

[4] C. Neelamegam, M. Punithavali, “A survey on object oriented quality metrics”, Global journal of computer science and technologies, pp. 183-186, 2011.

[5] B. Henderson, seller, “object oriented metrics: measure of complexity”, Prentice Hall, 1996.

[6] R.S.Pressman, ”Software Engineering-A practitioner’s Approach” Eight Edition, Mc. Graw Hill International Edition 2014.

[7] J. Bansiya, C. G. Davis, “A Hierarchical Model for Object-Oriented Design Quality Assessment”, IEEE Transactions on Software Engineering, 4-17,2002

[8] Z. Wu, J. Tang, Jiafu and C.K. Kwong, and C.Y. Chan, “A model and its algorithm for software reuse optimization problem with simultaneous reliability and cost consideration”, International Journal of Innovative Computing, Information and Control, Volume 7, Issue 5, 2011.

[9] M. Iyyappan, A. Kumar, S. Ahmad, S. Jha, B. Alouffi et al., "A component selection framework of cohesion and coupling metrics," Computer Systems Science and Engineering, vol. 44, no.1, pp. 351–365, 2023.

[10] Miguel Goulao, Fernando Brito e Abreu, “Composition Assessment Metrics for CBSE”, Proceeding EUROMICRO '05 Proceedings of the 31st EUROMICRO Conference on Software Engineering and Advanced Applications, 2005, pp-96- 105.

[11] Jianguo Chen, Wai K. YEAP, Stefan D. Bruda”A Review of Component Coupling Metrics for Component based Development” , Proceeding WCSE '09 Proceedings of the 2009 WRI World Congress on Software Engineering - Volume 04, pp. 65-69.

[12] P.K.Suri, NeerajGarg, “Software Reuse Metrics: Measuring Component Independence and its applicability in Software Reuse” International Journal of Computer Science and Network Security, Vol. 9, No.5, May 2009, pp. 237- 248.

[13] V. Lakshmi Narasimhan, P.T. Parthasarathy and M.Das, “Evaluation of a Suite of Metrics for Component Base Software Engineering”, Issues in Information Science and Information Technology, 2009, Vol. 6, 2009, pp. 732-740.

[14] P. Edith Linda, V.ManjuBashini, S.Gomathi, “Metrics for Component-Based Measurement Tools”, International Journal of

Scientific and Engineering Research, May 2011, Vol. 2, Issue 5, pp. 1-6.

[15] AbhikritiNarwal, “Empirical Evaluation of Metrics for Component-Based Software Systems”, International Journal of Latest Research in Science and Technology, Dec 2012, Vol 1, Issue 4, pp. 373-378.

[16] SidhuPravneet, “Quality metrics Implementation in Component based Software Engineering using AI Back Propagation Algorithm Software Component”, International Journal of Engineering and Management Sciences, 2012, Vol. 3(2), pp. 109-114.

[17] HeshamAbandah and IzzatAlsmadi, “Call Graph based Metrics to Evaluate Software Design Quality”, International Journal of Software Engineering and its Applications, Jan 2013 Vol.7, No.1, pp.1-12.

[18] TaranjeetKaur, RupinderKaur, “Comparison of various Lacks of Cohesion Metrics”, International Journal of Engineering and Advanced Technology, Feb 2013, Vol. 2, Issue 3, pp. 252-254.

[19] DivyaChaudhary, Prof. Rajender Singh Chillar, “Component Base Software Engineering Systems: Process and Metrics”, International Journal of Advanced Research in Computer Science and Software Engineering, July 2013, Vol. 3, Issue 7, pp. 91-95.

[20] Ermiyas Birihanu Belachew, Feidu Akmel Gobena and Shumet Tadese Nigatu, “ Analysis Of Software Quality Using Software Metrics”. International Journal on Computational Science & Application (IJCSA) Vol 8, No. 4/5, October 2018.

[21] Pooja P and D.A. Phalke, “Survey On Software Defect Prediction Using Machine Learning Techniques”. International Journal of Science and Research, Vol 3, December 2014

[22] Halstead, MH. Elements of software Science, New York, North-Holland; 1978.

[23] McCable TJ. A complexity measure, IEEE Transaction on Software Engineering, 1976; SE-2(4):308-320.

[24] Henry S, Kafura D. The evaluation of software systems’ structure using qualitative software metrics, Software- practice and Experience. 1984;14(6):561-573.

[25] Ming-Chang Lee “Software Quality Factors and Software Quality Metrics to Enhance Software Quality Assurance “ British Journal of Applied Science & Technology 4(21): June 2014

[26] Rahman HU, Raza M, Afsar P, Alharbi A, Ahmad S, Alyami H. Multi-criteria decision making model for application maintenance offshoring using analytic hierarchy process. Applied Sciences. 2021 Sep 14;11(18):8550.

[27] Muqeem M, Sultan A, Nazeer J, Farooqui MF, Alam A. Selection of Requirement Elicitation Techniques: A Neural Network based Approach. International Journal of Advanced Computer Science and Applications. 2022;13(1)

[28] Iyyappan M, Ahmad S, Jha S, Alam A, Yaseen M, Abdeljaber HA. A Novel AI-Based Stock Market Prediction Using Machine Learning Algorithm. Scientific Programming. 2022 Apr 1;2022.

[29] Ahmad S, Hasan M, Shahabuddin M, Tabassum T, Allvi MW. IoT based pill reminder and monitoring system. International Journal of Computer Science and Network Security. 2020 Jul;20(7):152-8.