I J C T A, 9(19) 2016, pp. 9221-9230
© International Science Press

Software Metrics and Soft Computing Models for Component Based Software: A Review

Shrddha Sagar[*], Pratistha Mathur[*] and Arun Sharma[**]

ABSTRACT

For the past ten years object oriented software development is not strong enough to handle changing requirements of the software. Component based software development deals with the different aspect that is use of components for the development of software. A reusable piece of software that is independent of any application is known as software component. Low cost, good quality and less time for the development software with others also are some of the important advantages. Literature review of software metrics for component based system and the soft computing models is done in this paper which are developed by number researchers. From the review we have concluded that more research in the field of soft computing can be done and implementation on the real projects can also be done.

Keywords: Component-Based Software Engineering, Component Based System, Software Component, and Software Metrics.

I. INTRODUCTION

Quality may be a product feature that meets the need of customers therefore provides the product satisfaction. So there is a need to maintain quality of the project at the appropriate label. Quality models help to effectively use the resources and also used in identifying program modules that are possibly to be defective [1].

For business success quality of software product is taken as an important element as the poor quality can be the reason for loss of financial loss, mission failure, permanent injury or human life [2].

Varieties of tasks or activities are described for such processes there are various models which take place during the process. This ISO/IEC 12207 aims at defining all the responsibilities that are necessary for developing and sustaining in the software [3].

Soft Computing is the combination of approaches that are designed for modeling and facilitates solutions for real world problems. Soft computing approaches such as Neural Network, Fuzzy Logic, Genetic Algorithm, and Neuro Fuzzy [4] have been applied to predict quality characteristics of software developed using module-oriented and object-oriented approaches. But less effort has been made in applying soft computing techniques on software developed using Component-Based Development (CBD) approach.

Paper is organized into 5sections. Section 2Component based Software Engineering, Sections 3 Software Metrics for Predicting Quality of Component Based Software Development, Section 4 Literature Review and Section 5 conclusion and future work.

II. COMPONENT BASED SOFTWARE ENGINEERING

In OOA code of object in used again and again, hence developer develops the modules once and does not require any change when the new object is added. In component based software engineering (CBSE), the

[*] Department of Computer Science, Banasthali University, Rajasthan, India, *E-mail: sagarshraddha@gmail.com; mathurprati@yahoo.com*

[**] Department of IT, Indira Gandhi Delhi Technical University for Women, Delhi, India, *E-mail: arunsharma2303@gmail.com*

existing component is reused number of times which reduces the effort of the software developer by implementing the usefulness of the component whose testing is already done [5, 6].

Software that are developed by using CBSE are cheaper and simpler then the software that are developed by using OOA. According to Ian [7], components are abstract in nature than object classes and can be considered as a stand-alone service provider.

In Component-based software development system are built by using number of software components. A component can be thought of as an independent module that provides information about what it does and how to it can be used with the interface without knowing the inner functionality of the code [8].

Now days, components are much better than an objects. But they go beyond objects by better addressing the requirement for replace-ability and by enabling the reuse the reuse of software parts that are larger-grained and at higher levels of abstraction. To solve the problem of fine-grained components, it is integrated and assembled which is a tedious task because the number of parts needed to construct a large application is huge in number [8].

A software component is a self-contained, reusable of part of software which includes interface surrounded by full specification which is independent from other software. An extra effort needs to be given on the additional functionality of the software components beyond the current application's requirements, for making the software component more useful [9].

III. SOFTWARE METRICS FOR PREDICTING QUALITY OF COMPONENT BASED SOFTWARE DEVELOPMENT

A set of characteristics and sub-characteristics are included in quality model in which the relationships between them contributes for specifying quality needs and for prediction of quality for the software components or the software system [10].

Software metrics objective can be defined in two parts: Firstly, the metrics are used for defining the advancement schedule and make fundamental regulation in the development phase which helps in reducing the classified risk. Secondly, the project metrics are joined for accessing the product quality and using these metrics, wherever required, alterations can be embed in the methodology that can be used for software quality enhancement process.

The four reasons for using metrics are for forecast, for distinguishing, for enhancing and for assessing [11]. The main purpose of metrics is to provide measurement that can improve the software process. Various methods are proposed by the researchers for the implementation of software metrics in the enhancement of software and design process which needs through study of the metrics.

There are large number of software metrics for measuring number of aspects, like reusability, maintainability complexity and others for traditional systems as well as for component-based systems. Barry Boehm predicts needs of the resource for small to large projects that is known as Constructive Cost Model [12]. According to Goodman [13], need of software metrics in every phase of software development process, product development gives required information time to time to the management for enhancing the processes and product quality. Sedigh-Ali et al. [14] defines the influence of the software metrics as usages throughout the SDLC process for handling the quantitative decision and accessing the quality assurance process for the financial anticipation.

Product and process metrics are the two broad categories of software metrics. Product metrics is collected from the requirement specification to the field operation and are analyzed at any phase of software development process. Product metrics comprises of size, documentation size, complexity etc. Characteristic's analysis and estimation is done through process metrics takes in the software development process [15].

The prediction of entities is crucial in software metrics analysis and design for software development planning [16]. The prediction is defined as an estimation of a particular entity [17]. Voas, [18] defines the predictability as an evaluation and forecasting of a future event. Some of the important metrics which needs prediction in software process are defects, quality, reliability, reusability, maintainability etc.

According to IEEE standard 982.2 [19]and Pfleeger and Lawrence [20], in any intermediate or final software product defect is an abnormality which is generated from an error or fault. In user documentation it is an erroneously test data set or incorrect data entry.

In few cases, a defect cannot be unprotected for a longer time. When an exacting code or logic of software is executed where the defect exists, it may be revealed during the execution support of that particular software module depending upon the particular situation, it can cause single or numerous failure at any point of time [21].

As per Basili and Rombach [22], reuse of software modules or components does not include software reuse but it includes use of the whole lot i.e. methodologies, knowledge, processes etc. Software reuse is a systemic methodology in which association creates the accessible defined operating procedures for generating, indicating, adjusting and quantifying to the facts of use in the software development process [23]. Reusability is a quantification that defines that how much a component can be reused for reducing the cost and schedule by taking into consideration the existing interfaces and code [24].

With no difficulty the degree to which software can be changed or enhanced is known as software maintainability [19]. Reuse is the main aim of CBSD of the existing components in the software development process for reducing the time and effort. It can be concluded that maintainability can be considered as one of the most important feature of software quality in CBSD.

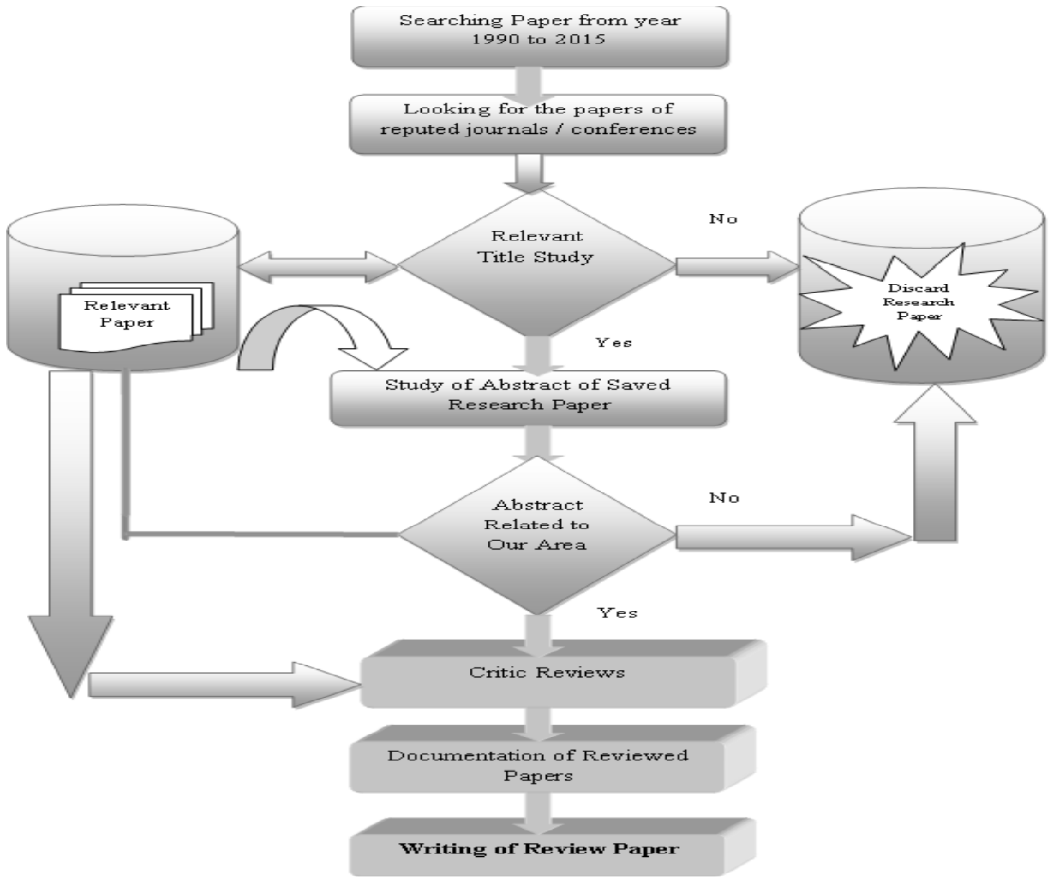
Searching Paper from year 1990 to 2015
Looking for the papers of reputed journals / conferences
Relevant Title Study
No
Relevant Paper
Discard Research Paper
Yes
Study of Abstract of Saved Research Paper
Abstract Related to Our Area
No
Yes
Critic Reviews
Documentation of Reviewed Papers
Writing of Review Paper

Figure 1: Methodology For Reviewing Paper

Fig. 1 shows the process which we have used for reviewing the research papers which are included in this paper and at last we have written this written this paper.

IV. LITERATURE REVIEW

For literature survey, we have considered research papers based on software quality characteristics of CBSD from various reputed journals and conferences as discussed below.

There are number of quality models proposed till now. In McCall Model [25], the authors have suggested software quality structure and classified the quality factors into three set, namely: product revision, product transition and product operation. Second quality model has been proposed by Boehm and others [26], which in addition to factors included in McCall Model authors have included hardware factors and others such as generality, documentation, economy and many more. FURPS model [27] decomposes software quality into two diverse segments of needs, namely: non- functional requirements such as reliability, usability, etc. and functional requirements defined by giving the input and the expected output. Nagib and Callaos [28] suggested quality cube model which describes the complete concept of software quality for the development of software. Dromey [29] proposed a new model which includes characteristics such as: process maturity and reusability. Author has classified the factors into four groups: descriptive, correctness, internal, and contextual.

ISO 9126 is the proposed a quality standard by ISO which gives the general definition of software quality, in relation to six main factors. ISO model is a general model and various researchers have used ISO model by addition or removal of attributes/sub- attributes from this, which can be used for component based systems.

Davis and Bansiya [30] proposed Quality Model for Object-Oriented Design (QMOOD), which is a hierarchical quality model that is devoted to the estimation of object-oriented intended quality. Cai, et al. [31], authors have also addressed quality assurance (QA) issues for CBSD. In this paper authors have proposed a QA model which comprises of component development, component requirement analysis, component customization, integration, system architecture design, testing & maintenance, and component certification.

Alvaro, et al. [32] have recognized the harms of the selection of the component on the basis of integration, formalization of the component catalogs and the vague quality of third-party developed software components, which brought new challenges in the software engineering domain. The quality factors and correlated software metrics for the quality prediction of the software component are discussed in the proposed component quality model.

Pande, et al. [33] suggested advices for enhancing software quality in COTS (component off-the-shelf) based products by designing software quality metrics which will be helpful in managing and enhancing quality in CBSD and also estimated various quality models. Mathieu and Pierre [34] have communicated the preface results of a methodical review was done by authors for determining the impact of Software Process Improvements (SPI) on developers' behaviors and on structural quality.

Alvaro, et al. [35] projected a software component quality structure for evaluating the software components quality in a well-organized way. The conduct experimental study also planned, defined, analyzed, interpreted and operated in order to estimate the feasibility of the process for software component evaluation.

McCabe [36] proposed a Cyclomatic Complexity (CC) metrics, which is pedestal on the connected and number of nodes of the software components in a graph of a software program. Kafura and Henry [37] also anticipated on the number of local information-flow while entering called as fan-in and while exiting called fan-out by using complexity metrics for each procedure. For measuring the program and unit complexity Halstead complexity metrics was used that can be directly implemented on the source code where main aim is to emphasis on the computational complexity [38]. Weyuker [39] for evaluation of syntactic software complexity the author have anticipated a set of properties. Few standard complexity measures are used for the evaluation adjacent to these properties.

Kumari and Bhasin [40] have discussed different features of complexity of CBS for proposing composite complexity metrics. Interface and type among component, nesting level of control structures involved in codes of software components, size of component are some of number of attributes of complexity.

Bertoa and Vallecillo [41] represented the assembly of software component metrics and identified that the main quality characteristic usability for any software. Authors defined, usability metrics on the ISO 9126 quality model for component based software.

Bertoa, et al. [42] presented a set of foundational and unoriginal measures that can be used for estimating usability for software components, and some indicators for classifying them according to IEEE categories. Gill and Tomar [43] have identified testing and testability of the software components that are the most important features for predicting reusability and quality for software components in CBSD. They have introduced an innovative process for building testable component interfaces which are well-defined standard.

Lisa and Delugach [44] have discussed the conceptual graph in relation to dependency depiction. Conceptual graphs(CG) are semantic network, logic based and formal representation. Abdellatie, et al. [45] have mainly worked for bridging the differences among software component provider and software component user specific to the domain of component information flow (CIF) prediction and multidimensional methodologies for the estimation explanation.

Narasimha, et al. [46] discussed number of software metrics which are proposed by several researchers which can be used for the evaluation by using number of large-scale freely available software systems. Research work done by Zhan, et al. [47] focuses on the possible fundamental features of the software quality. The fundamental features of the software quality are discussed in five aspects that is process management, staff experience, project characteristics, code and document review, stakeholders or customer involvement.

As the size and complexity of the software is increasing day by day, hence, there is the urgent requirement of the software reuse. Reuse of anything which is related to the software project which includes knowledge is known as software reuse [48]. The process in which an organization defines a set of methodical operating measures for producing, classifying, retrieving, adapting and specifying software object for the need of utilizing them in the development process can be defined as reusability [49]. Doing of less coding and more integration of software components in the software development process, will reduce the development cost [50].

Selby and Porter [51] studied number of projects from NASA in which the concept of software reuse is used mostly and has less dependency on the other modules. Author found that high-quality documentation was also one of the property important factor of reusability. Caldiera and Basili [52] proposed a model for reusability which considers the three factors such as: usefulness of reusable components, quality of the reusable components and cost of reuse.

Gill [53] has suggested some guidelines for high reusability for software components which are as follows: performing cost-benefit analysis for reuse, selecting pilot projects for deployment of reuse, adoption of standards for software components, identifying the reuse metrics and at last conducting reuse assessment.

Rotaru and Dobre [54] have proposed mathematical model and software metrics for estimating the adaptability, compose-ability, flexibility and reusability of the software components. The authors have also proposed software metric for the adaptability and complexity for solving the problem by using component based use cases. Hristov, et al. [55] had identified that till now; no apparent structure for estimation of reusability for software and formalization of suitable software metrics are rarely found in literature. Authors projected software reusability prediction model and software metrics but, empirical validation of the proposed model and software metrics is not done by the authors.

Jatain and Gaur [56] developed fuzzy logic model for the prediction of software component reusability for CBS systems. Authors have implemented the proposed model on the real time project for the validation. Khairuddin and Elizabeth [57] proposed a maintainability model which includes the quality factors such as: readability, level of validation and testing, modularity, complexity, programming language and traceability are used for the prediction of software maintainability for software systems. Fioravanti and Nesi [58] projected an additional model for adaptive maintenance. The effort of the proposed model for predicted on the number of operations such as: addition, deletion of facts, understanding, modification or changes in the code of other part of software.

Singh, et al. [59] proposed Artificial Neural Network model for the estimation of the software maintainability of the software system. Authors have considered following factors like: documentation quality, understandability of the software, average cyclomatic complexity and readability of source code are the input factors for the estimation of the software maintainability. Authors have implemented back propagation algorithm of Artificial Neural Network. Aggarwal, et al. [60] considered principal components of eight object-oriented metrics which are taken as input for estimating the software maintenance of object-oriented systems. Authors have used back propagation algorithm of Artificial Neural Network for training the network. Shukla and Mishra [61] had trained the Artificial Neural Network by using fourteen factors of cost drivers for their study and trained the network by taking into account different number of neurons and hidden layers. Sharma, et al. [62, 63] added one sub-factor named trackability under Maintainability and extended the ISO 9126 model. Authors have proposed a fuzzy logic based model for prediction of software maintainability for the CBS.

Nerurkar, et al. [64], suggested a fuzzy logic based model for prediction of software reusability. Authors have identified the factors which are affecting reusability of the software components and after implementation on the real time applications the results found to be quite satisfactory. Sharma and Baliyan [65] surveyed existing quality models like, Boehm quality model, ISO 9126 model and McCall quality model, and have identified the quality factors which are affecting maintainability of the CBSS. Authors have also included new sub factors of maintainability for the CBSS like reusability, scalability, taliorability and trackability.

Singh, et al. [66] proposed a model for estimation of software reusability levels by using soft computing approaches namely: Neuro-fuzzy, Artificial Neural Network and Fuzzy Logic. Authors have identified four factors as interface complexity, documentation quality, changeability and understandability of software. Authors have validated on the real time project.

Tahir and MacDonell [67] identified some of the critical issues related to the design, implementation and selection of the software metrics for the prediction of the quality. Pande, et al. [68] have introduced the pliability metric, that can be implemented on the component-based orientation and expand preceding metrics.

Mathur, et.al., [69]has compared different soft computing techniques and proposed a model for selection of the component by evaluating the software maintainability of software component. Root mean square error (RMSE) performance measure is used for validation

Kaur, et.al., [70] evaluated and compared the application of different soft computing techniques like ANNs, FIS & ANFIS for developing a model for effort estimation of maintainability . Tagyi, et.al., [71] has used ANNs & Fuzzy system for experimental study in software engineering. Yuan, et.al [72] has used fuzzy subtractive clustering to making rules for an FIS and calculated number of faults in telecommunication system by using proposed model.

Sharma, et al [73] has proposed Fuzzy-Analytic Hierarchy Process (Fuzzy-AHP) model for estimation of reusability of the software component and rank the components according to the reusability values. Validation was done on six components.

V. CONCLUSION

Quality factors like maintainability, reusability, reliability, etc contributes to the large part of development process of software whereas demand for software concentrated systems namely: application programs are expected to increase.

Based on the literature review it is found that soft computing approaches such as artificial neural network, genetic algorithm and fuzzy logic are widely used. These soft computing techniques have not been used widely in predicting quality characteristics of CBSD. From literature review we can conclude that more use of soft computing techniques like AHP, ANP, fuzzy-ANP, fuzzy-MOORA etc to design and evaluate the quality characteristics / models of CBS.

REFERENCES

[1] Tomar, A. B., and Thakare, V. M., (2011), "A systematic study of software quality models.", International Journal of Software Engineering & Applications2, Issue 4, pp: 1-61.

[2] Pensionwar, Rutuja K., Anilkumar Mishra, and Latika Singh (2013), "A Systematic Study Of Software Quality–The Objective Of Many Organizations."International Journal of Engineering Research and Technology. Vol. 2, Issue 5.

[3] Maheshwari, S. and Prof. Dinesh Ch. Jain, D.Ch., (2012), "A Comparative Analysis of Different Types of Models in Software Development Life Cycle", International Journal of Advanced Reseach in Computer Science and Software Engineering, Vol. 2, Issue 5.

[4] Zadeh and Lotfi A., (1994), "Fuzzy Logic, Neural Networks, and Soft Computing", Magazine Communications of the ACM CACM Homepage archive Vol. 37, Issue. 3, pp: 77-84.

[5] Zhang.G, (2000), "Component-Based Software Engineering", Thesis.

[6] Mili.H, Mili.F, Mili.A, (1995), "Reusing Software: Issues and Research Direction", IEEE Transaction on Software Engineering, Vol.21, Issue 6, pp: 528-561.

[7] Ian Sommerville, (2004) "Software Engineering", 7th edition.

[8] Piscataway, N.J., IEEE 1517, (1999), "Introduction to IEEE Std. 1517- Software Reuse Process", IEEE.

[9] Gill, N. S., Grover, P. S., (2004), "Few Important Considerations for Deriving Interface Complexity Metric for Component-Based Systems", ACM SIGSOFT Software, Vol. 29, Issue: 2, pp: 4-4.

[10] Losavio, F., Ortega, D., Perez, M., (2002), "Modeling EAI [Enterprise Application Integration]", Computer Science Society, Proceedings: 22nd International Conference of the Chilean, pp: 195-203.

[11] Park, R. E., Goethert, W. B., Florac W. A., (1996), "Goal Driven Software Measurement - A Guidebook", Software Engineering Institute, Carnegie Mellon University.

[12] Boehm, B., (1981), "Software Engineering Economics", NJ: Prentice- Hall.

[13] Goodman, P., (1993), "Practical Implementation of Software Metrics", McGraw Hill, London.

[14] Sedigh-Ali, S., Ghafoor, A., Paul, R. A., (2001), "Metrics-guided quality management for component-based software systems", Proceedings of the 25th International Computer Software and Applications Conference on Invigorating Software Development, pp: 303-308.

[15] Pressman, R. S., (2005), "Software Engineering: A Practitioner's Approach", 6th Edition, McGraw Hill Book Co.

[16] Furulund K.M. and Molokken-Ostvold, K., (2007), " Increasing Software Effort Estimation Accuracy-Using Experience Data, Estimation Models and Checklists", In Proceedings of QSIC 2007, pp:342-347.

[17] Jorgensen, M. and Shepperd, M., (2007), "A Systematic Review of Software Development Cost Estimation Studies", IEEE Transaction on Software Eng, Volume 33, Issue 1, pp.33-53.

[18] Voas, J., (2000), "Can chaotic methods improve software quality predictions?", IEEE Software, pp: 20-22.

[19] IEEE Std. 610.12-1990, (1993), "Standard Glossary of Software Engineering Terminology", IEEE Computer Society Press, Los Alamitos, CA.

[20] Pfleeger, Lawrence, S, (1992), “Measuring software reliability”, Spectrum, IEEE, 1992, Vol 29, Issue 8, pp:56-60.

[21] Kan H.S., (2003), ” Metrics and Models in Software Quality Engineering”, 2nd ed. Edition, MA: Addison-Wesley.

[22] Basili, V.R., Rombach, H. D., (1988), “Towards A Comprehensive Framework for Reuse: A Reuse-Enabling Software Evolution Environment”, Technical Report CS-TR-2158, Dept. of Computer Science, Univ. of Maryland, College Park, MD 20742.

[23] Mili, H., Mili, F., Mili, A., (1995), “ Reusing Software: Issues and Research Directions”, IEEE Transaction on Software Engineering, Vol. 21, Issue 6, pp: 528-561.

[24] Wang AJA., (2002), “Reuse Metrics and Assessment in Component-based Development”, Proceedings of 6th IASTED International Conference on Software Engineering and Applications IASTED, Cambridge, Massachusetts, USA, pp:583–588.

[25] McCall, J. A., Richards, P, K., and Walters, G, F., (1977), “Factors in Software Quality”, Springfield, VA, National Technical Information Service, Vol. 3, Issue. 5, pp: 133-139.

[26] Boehm, B, W., (1978), “Characteristics of Software Quality, TRW Series of software Technology”, Amsterdam, North Holland.

[27] Grady, R, B., (1992), “Practical Software Metrics for Project Management and Process Improvement”, Publication: Book, Practical software metrics for project management and process improvement Prentice-Hall, Inc. Upper Saddle River, NJ, USA.

[28] Nagib, C., Callaos, B., (1994), “Designing with Systemic Total Quality”, Educational Technology, Vol. 34, Issue.1, pp: 29-36.

[29] Dromey, R.G., (1995), “A Model for Software Product Quality”, Published in: IEEE Transactions on Software Engineering, Vol. 21, Issue. 2, pp: 146 – 162.

[30] Bansiya, J., and Davis, C.G., (2002), “A Hierarchical Model for Object-Oriented Design Quality Assessment”, Published in: IEEE Transactions on Software Engineering, Vol. 28 , Issue 1, pp: 4 – 17.

[31] Cai X., Lyu M.R., Kam-Fai Wong, Roy Ko, (2000), “Component-Based Software Engineering Technologies, Development Frameworks, and Quality Assurance Schemes”, IEEE Computer Society, Lecture Notes, pp:372—379.

[32] Alvaro, A., de Almeida, E.S. ; Meira, S.L., (2006), “A Software Component Quality Model: A Preliminary Evaluation”, Published in: Software Engineering and Advanced Applications, 2006. SEAA ’06. 32nd EUROMICRO Conference, pp: 28-37, Cavtat, Dubrovnik.

[33] Pande. J., Bisht. R. K., Pant. D., Pathak. V. K., (2010), “On Some Quality Issues of Component Selection in CBSD”, J. Software Engineering & Applications, , Vol. 3, pp: 556-560.

[34] Mathieu L., and Pierre N, R., (2011), “Do Software Process Improvements Lead to ISO 9126 Architectural Quality Factor Improvement”, Proceeding: ACM Proceedings of the 8th international workshop on Software quality, (WoSQ ’11), pp: 11-17.

[35] Alvaro, A., Almeida, E, S., Meira, S, R, L., (2010), “A Software Component Quality Framework”, ACM SIGSOFT Software Engineering Notes, Vol. 35, Issue 1, pp: 1-18.

[36] McCabe, T.J., (1976) , “A Complexity Measure”, Published in: IEEE Transactions on Software Engineering, Vol. 2 , Issue. 4, Page(s): 308-320.

[37] Kafura, D., and Henry, S., (1981), “Software Structure Metrics Based on Information Flow”, Published in: IEEE Transactions on Software Engineering, Vol. 7 , Issue: 5, pp: 510-518.

[38] Halstead, M, H., (1977), “Elements of Software Science (Operating and programming systems series)”, Publication: Book Elements of Software Science (Operating and programming systems series) Elsevier Science Inc. New York, NY, USA.

[39] Weyuker. E.J., (1988), “Evaluating Software Complexity Measures”, Published in: IEEE Transactions on Software Engineering, Vol.14 , Issue. 9, pp: 1357-1365.

[40] Kumari, U., and Bhasin, S., (2011), “A Composite Complexity Measure for Component-Based Systems”, ACM SIGSOFT Software Engineering Notes, Vol. 36, Issue. 6, pp: 1-5.

[41] Bertoa, M, F., and Vallecillo, A., (2004), “Usability Metrics for Software Components”, QAOOSE’04: Proceedings of the 8th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering. pp: 1- 10.

[42] Bertoa, M, F., Troya, J, M., Vallecillo, A., (2006), “Measuring the Usability of Software Components”, Journal of Systems and Software Vol. 79, Issue 3, pp:427-439.

[43] Gill, N, S., and Tomar, P., (2011), “New and Innovative Process to Construct Testable Component with Systematic Approach”, Newsletter: ACM SIGSOFT Software Engineering Notes archive, Volume 36 Issue 1, pp: 1-4.

[44] Lisa, C., and Delugach, H. S., (2001), “Dependency Analysis Using Conceptual Graphs”, In Proceedings of the 9th International Conference on Conceptual Structures, (ICCS92001), pp: 117-130.

[45] Abdellatief, M., Sultan, A, B, M., Abdul Ghani, A.A., Jabar, M.A., (2011), “ Multidimensional Size Measure for Design of Component-Based Software System”, www.ietdl.org, IET Software, 10.1049/iet-sen.2011.0122.

[46] Narasimhan, V.L., Parthasarathy, P, T., and Das, M., (2009), “Evaluation of a Suite of Metrics for Component Based Software Engineering(CBSE)”, Issues in Informing Science and Information Technology 6.5/6 (2009), pp: 731-740.

[47] Zhan.J., Zhou.X., Zhao.J., (2010), “The Impacts of Some Distrustable Factors on Software Quality”, Published in: Multimedia Information Networking and Security (MINES’10), Nanjing, Jiangsu, pp: 772-776.

[48] Basili, V.R., and Rombach, H.D., (1988), “The TAME Project: Towards Improvement-Oriented Software Environments”, Published in: IEEE Transactions on Software Engineering, Vol. 14 , Issue. 6, pp: 758-773.

[49] Mili, H., Mili, F. ; Mili, A, (1995), “Reusing Software: Issues and Research Directions”, Published in: IEEE Transactions on Software Engineering, Vol.21 , Issue.6 , pp: 528-562.

[50] Wang.Q., Mei.H., Yang.F., Chen.F., (2002), “An Architecture-Based Approach for Component-Oriented Development”, Published in: Computer Software and Applications Conference, . Proceedings. 26th Annual International, (COMPSAC’02), pp: 450-455.

[51] Selby.R.W., and Porter. A.A., (1989), “Software Metric Classification Trees Help Guide The Maintenance of Large-Scale Systems”, Published in conference proceeding of Software Maintenance, 1989., pp: 116-123.

[52] Caldiera, G., and Basili, V.R., (1991) “Identifying and Qualifying Reusable Software Components”, Published in: Computer Vol: 24 , Issue: 2, pp:61-70.

[53] Gill, N, S., (2003), “Reusability Issues in Component-Based Development”, ACM SIGSOFT Software Engineering Notes, Vol. 28, Issue 4, pp: 1-5.

[54] Rotaru.O.P., and Dobre.M., (2005), “Reusability Metrics for Components”, In Computer Systems and Applications, 2005. The 3rd ACS/IEEE International Conference in IEEE, pp: 24.

[55] Hristov, D., Hummel, O., Huq, M, W.., Janjic, (2012), “Structuring Software Reusability Metrics for Component-Based Software Development”, The Seventh International Conference on Software Engineering Advances, (ICSEA’12), Lisbon, Portugal, pp: 421-429.

[56] Jatain, A., Gaur, D., (2012), “Estimation of Component Reusability by Identifying Quality Attributes of Component – A Fuzzy Approach”, Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology, (CCSEIT ’12), pp: 738-742.

[57] Khairuddin, H., and Elizabeth K., (1996), “A Software Maintainability Attributes Model”, Malaysian Journal of Computer Science, Vol. 9, Issue. 2, pp: 92-97.

[58] Fioravanti, F., and Nesi, P., (2001), “Estimation and Prediction Metrics for Adaptive Maintenance Effort of Object-Oriented Systems”, Published in: IEEE Transactions on Software Engineering, Vol.27, Issue. 12 , pp: 1062-1084.

[59] Singh.Y., Bhatia.P.K., Sangwan.O., (2009), “ANN Model for Predicting Software Function Point Metric”, Newsletter: ACM SIGSOFT Software Engineering Notes archive, Vol. 34, Issue.1, pp: 1-4.

[60] Aggarwal, K, K., Singh, Y., Kaur, A., and Malhotra, R., (2006), “Application of Artificial Neural Network for Predicting Maintainability using Object-Oriented Metrics”, World Academy of Science, Transactions on Engineering, Computing and Technology Vol.15, pp: 285-289.

[61] Shukla.R., Misra.A.K., (2008), “Estimating Software Maintenance Effort: a Neural Network Approach”, Proceeding: ACM Proceedings of the 1st India software engineering conference, (ISEC ’08), pp: 107-112.

[62] Sharma. A., Kumar. R., Grover.P. S., (2009), “Reusability Assessment for Software Components”. ACM SIGSOFT Software Engineering Notes V0l. 34, Issue 2, pp: 1-6.

[63] Sharma. A., Kumar.R., and Grover.P.S., (2007), “A Critical Survey of Reusability Aspects for Component-Based Systems”, Proceedings of the World Academy of Science, Engineering and Technology (2007), Vol. 21, pp: 35-39.

[64] Nerurkar.N.W., Sharma, A., Sagar.S., (2010) , ” A Soft Computing Based Approach to Estimate Reusability of Software Components”, ACM SIGSOFT Software Engineering Notes, Vol. 35, Issue. 5, pp: 1-5.

[65] Sharma.V., and Baliyan.P., (2011), “Maintainability Analysis of Component Based Systems”, International Journal of Software Engineering and Its Applications , Vol. 5, Issue. 3, pp:107-118.

[66] Singh.Y., Bhatia.P, K., Sangwan.O., (2011), “Software Reusability Assesssment Using Soft Computing Techniques”, ACM SIGSOFT Software Engineering Notes, Vol. 36, Issue.1, pp: 1-7.

[67] Tahir.A., and MacDonell.S.G., (2012), “ A Systematic Mapping Study on Dynamic Metrics and Software Quality”, Published in: 28th IEEE International Conference on Software Maintenance, (ICSM’12), Trento, pp: 326-335.

[68] Pande, J., Garcia, C, J., Pant, D., (2013), "Optimal Component Selection for Component Based Software Development using Pliability Metric", ACM SIGSOFT Software Engineering Notes, Vol. 38, Issue. 1, pp: 1-6.

[69] Mathur.P., Sagar.S., and Sharma.A., (2016), "Neural Network for Estimation of Maintainability for Component Based System: A Comparative Analysis", International Journal of Advancements in Computing Technology (IJACT), Vol. 8, Issue 1, pp: 21-30.

[70] Kaur, Arvinder, Kamaldeep Kaur, and Ruchika Malhotra, (2010) "Soft computing approaches for prediction of software maintenance effort." International Journal of Computer Applications Vol. 1, Issue 16.

[71] A.I Tagi, M. Khoshgoftaar, A. Abran, (2002), "Can Neural Networks be easily interpreted in Software Cost Estimation", IEEE Transactions on Software Engineering, pp: 1162-1167.

[72] Yuan X. , Khoshgoftaar T.M., Allen E.B., Ganesan K., (2000), " An Application of Fuzzy Clustering to Software Quality Prediction" Proceedings of IEEE Symposium on Application Specific Systems and Software Engineering Technology, pp: 85-90.

[73] Sharma.A., Sagar.S., and Mathur.P., (2015), " Mutli-Criteria Selection of Software Components Fuzzy-AHP Approach", International Journal Of Innovative Computing, Information and Control, Vol. 11, Iusse 3, pp: 1045-1058.