



A Survey on Evaluation of the Quality of Software System by Using Fuzzy Logic Approach

By Ritika Vern & Sanjay Kumar Dubey

Amity Institute of Technology, Noida

Abstract - The demand for quality software system is increasing day by day because quality defines the capability of the software systems. Software systems are root of any developing systems. They are always checked on the basis of their quality. The quality characteristics are considered as very important aspects by development of software system. The evaluation of quality also plays important role in this context. Fuzzy logic is the new emerging technology for evaluating the efficiency of various systems. This paper presents various quality models and evaluation of quality models by using fuzzy logic technique. The present study may serve the purpose of reference for investigating evaluation of software quality models with their analytical description.

Keywords : *fuzzy logic, quality, software system, model.*

GJCST-C Classification : *H.3.4*



A SURVEY ON EVALUATION OF THE QUALITY OF SOFTWARE SYSTEM BY USING FUZZY LOGIC APPROACH

Strictly as per the compliance and regulations of:



RESEARCH | DIVERSITY | ETHICS

A Survey on Evaluation of the Quality of Software System by Using Fuzzy Logic Approach

Ritika Vern^a & Sanjay Kumar Dubey^a

Abstract - The demand for quality software system is increasing day by day because quality defines the capability of the software systems. Software systems are root of any developing systems. They are always checked on the basis of their quality. The quality characteristics are considered as very important aspects by development of software system. The evaluation of quality also plays important role in this context. Fuzzy logic is the new emerging technology for evaluating the efficiency of various systems. This paper presents various quality models and evaluation of quality models by using fuzzy logic technique. The present study may serve the purpose of reference for investigating evaluation of software quality models with their analytical description.

Keywords : fuzzy logic, quality, software system, model.

I. INTRODUCTION

Quality means ability of product to be able to satisfy the users. Quality is a Process of building the sustained relationships by assessing, anticipating, and fulfilling the needs. Quality is for the systematic monitoring and evaluation of the various aspects of a project, service, or facility to ensuring the standards of quality are being met. Quality is the totality of features and characteristics of a product or service that bears its ability to satisfy stated or implied needs [34]. Quality of a software system depends on their functional and non functional requirements. The requirements and characteristic plays a central role in the definition of the quality of software. The existence of the relationships between requirements and characteristics make statements about the quality of a possible product.

Fuzzy logic and its set theory have developed and implemented to many different research areas. Related issues to fuzzy theory have been handled by deterministic or by probabilistic approaches. Fuzzy logic is a multi valued logic that allows intermediate values to be defined between conventional evaluations like yes/no, high/low. Though fuzzy logic and software system quality are unknown field for each other but still a précised tool i.e. fuzzy logic will help out in evaluation of quality of system software.

In this paper we will be survey on various literatures which holds software quality through different aspects. There have been various analyses done in the field of software quality. In this work we study the various quality sources in software system that shows how

evaluation of quality is being done. These studies will help to improve software quality which helps us to plan and allocate testing resources in early phases of software development. Previously analyses have been done to carry out some of the important reviews.

II. QUALITY MODELS

Quality of a software system is defined on the basis of quality models. Software quality deals with software requirements which act as the foundation. They are appreciated way to deal with the quality issues in software developing field they are as followed:

a) McCall Model

McCall model was presented by Jim McCall. It basically connects users and developers by focusing on the number of software quality factor that reflects both user`s and developer`s view [24]. McCall model was the basic quality model which is into three main categories-

i. Product Operator

- a) Correctness-the functionality matches with specification.
- b) Reliability-the extent to which the system fails.
- c) Efficiency-system resource usage.
- d) Integrity-protection from unauthorized access.
- e) Usability-ease of use.

ii. Product Revision

Product revision includes maintainability (the effort required to locate and fix a fault in the program within its operating environment), flexibility (the ease of making changes required by changes in the operating environment) and testability (the ease of testing the program, for ensuring that it is an error-free and meets its specification).

- a) Maintainability-the ability to find and fix a defect.
- b) Flexibility-the ability to make changes required as defined by business.
- c) Testability-the ability to validate the software requirements.

iii. Product Transition

Product transition is all about portability (an effort require to transferring a program from one environment to another), reusability (a comfort of reusing software in a different context) and

interoperability (effort required to couple the systems to another systems).

- a) Portability-the ability to transfer the software from one environment to another.
- b) Reusability-the ease of using existing software components in a different context.
- c) Interoperability-the extent to which software components work together.

b) Boehm Quality Model

This second quality model is presented by Barry Boehm. This model defines software quality by a set of attributes and metrics [2]. The high-level characteristics represent basic level requirements of actual use to which evaluation of software quality could be put together for the general utility of software. The high-level characteristic includes three main questions that a buyer of software has:

- i. As-is utility: How well (easily, reliably, efficiently) can I use it as-is?
- ii. Maintainability: How easy is it to understand, modify and retest?
- iii. Portability: Can I still use it if I change my environment?

The intermediate level characteristic represents Boehm's 7 quality factors that represent the qualities expected from a software system:

- i. *Portability (General utility characteristics)*: Code possesses the characteristic portability that it can be operated easily on other compute configurations.
- ii. *Reliability (As-is utility characteristics)*: Code possesses the characteristic reliability that it can be expected to perform its functions at a satisfied level.
- iii. *Efficiency (As-is utility characteristics)*: Code possesses the characteristic efficiency that it fulfills its purpose without wasting the resources.
- iv. *Usability (As-is utility characteristics, Human Engineering)*: Code possesses the characteristic usability that it is reliable, efficient and human-engineered.
- v. *Testability (Maintainability characteristics)*: Code possesses the characteristic testability that it verifies the criteria and supports evaluation of its performance.
- vi. *Understandability (Maintainability characteristics)*: Code possesses the characteristic understandability that it has purpose is to clears the inspector.
- vii. *Flexibility (Maintainability characteristics, Modifiability)*: Code possesses the characteristic modifiability that it facilitates the incorporation of changes.

c) FURPS

The FURPS model was presented by Robert Grady [5]. The FURPS-categories are of two different types: Functional (F) and Non-functional (FURPS). These categories can be used as both product requirements

as well as in the assessment of product quality. FURPS stands for:

- i. *Functionality* - This may include feature sets, capabilities and security.
- ii. *Usability* - This may include human factors, aesthetics, and consistency in the user interface, online and context sensitive. Help wizards and agents, user documentation, and training materials.
- iii. *Reliability* - This may include frequency and severity of failure, recoverability, predictability, accuracy, and mean time between failures (MTBF).
- iv. *Performances* - Imposes conditions on functional requirements such as speed, efficiency, availability, throughput, response time, recovery time, and resource usage.
- v. *Supportability* - This may include testability, extensibility, adaptability, maintain-ability, compatibility, configurability, serviceability, installs ability, localizability (internationalization).

d) Dromey's Quality Model

According to Dromey, high level quality attributes such as reliability or maintainability cannot be built into software directly [4]. What can be done is to identify and build in a consistent, harmonious, and complete set of product properties (such as modules without side effects), that result in the manifestations of reliability and maintainability. Dromey's quality framework, like the earlier models, relies on the decomposition of high-level quality attributes into tangible, quality-carrying properties of a product's components (requirements, design and implementation). Dromey's generic quality model deals with three principal elements including the product properties that influence quality, a set of high-level quality attributes, and a means of linking them. The model relies upon identifying a set of components that compose the product. Some of the components are atomic, while others are compositions of the simpler components. Identification of tangible properties with the components and association of the tangible properties with the component composition largely determine overall product quality. Component selection, component property identification, and component composition determine overall quality altogether. Since software quality is often discussed in terms of high-level attributes such as functionality, reliability, etc, a set of complete, compatible and no overlapping high-level quality attribute needs to be identified.

e) ISO 9126

ISO 9126 is an international standard for the evaluation of software which is divided into four parts [11, 12, 13]. Quality model; external metrics; internal metrics; and quality in use metrics [. ISO 9126 Part one, referred to as ISO 9126-1 which is a of previous work done by McCall, Boehm, FURPS and others in defining a set of software quality characteristics. ISO9126-1

represents the latest (and ongoing) research into characterizing software for the purposes of software quality control, software quality assurance and software process improvement (SPI). This article defines the characteristics identified by ISO 9126-1. The other parts of ISO 9126, concerning metrics or measurements for these characteristics, are essential for SQC, SQA and SPI but the main concern of this article is the definition of the basic ISO 9126 Quality Model.

The ISO 9126 documentation itself, from the official ISO 9126 documentation, can only be purchased and is subject to copyright. SQA.net only reproduces the basic structure of the ISO 9126 standard and any descriptions, commentary or guidance are original material based on public domain information as well as our own experience. The ISO 9126-1 software quality model identifies 6 main quality characteristics, namely:

- i. *Functionality*: A set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs.
- ii. *Reliability*: A set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time.
- iii. *Usability*: A set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users.
- iv. *Efficiency*: A set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions.
- v. *Maintainability*: A set of attributes that bear on the effort needed to make specified modifications.
- vi. *Portability*: A set of attributes that bear on the ability of software to be transferred from one environment to another.

III. FUZZY LOGIC

Fuzzy logic began with the proposal of fuzzy set theory by Lotfi Zadeh [30, 31]. Fuzzy logic has been applied to many fields, from control theory to artificial intelligence. Basically Fuzzy Logic is a multi-valued logic that allows intermediate values to be defined between conventional evaluations like true/false, yes/no, high/low, etc. Notions like rather tall or very fast can be formulated mathematically and processed by computers, in order to apply a more human-like way of thinking in the programming of computers. Fuzzy systems are an alternative to traditional notions of set membership and logic that has its origins in Ancient Greek philosophy.

The very basic notion of fuzzy systems is fuzzy set. In classical mathematics we are familiar with what we call crisp sets, For example, the possible interformetic coherence g values are the set X of all real numbers between 0 and 1. From this set X a subset A

can be defined, (e.g. all values $0 \leq g \leq 0.2$). The characteristic function of A , (i.e. this function assigns a number 1 or 0 to each element in X , depending on whether the element is in subset A or not) is shown in Fig.1. The elements which have been assigned the number 1 can be interpreted as the elements which have been assigned the number 1 can be interpreted as the elements which have been assigned the number 1 can be interpreted as the elements that are in the set A and the elements which have assigned the number 0 as the elements that are not in the set. Fuzzy Logic provides a different way to approach a control or classification problem. This Method focuses on what the system should do rather than trying to model how it works. One can concentrate on solving the problem rather than trying to model the system mathematically, if that is even possible. On the other hand the fuzzy approach requires a sufficient expert knowledge for the formulation of the rule base, the combination of the sets and the defuzzification. In General, the employment of fuzzy logic might be helpful, for very complex processes, when there is no simple mathematical model (e.g. Inversion problems), for highly nonlinear processes or if the processing of (linguistically formulated) expert knowledge is to be performed. According to literature the employment of fuzzy logic is not recommendable, if the conventional approach yields a satisfying result, an easily solvable and adequate mathematical model already exists, or the problem is not solvable.

IV. REVIEW OF LITERATURE

a) *Evaluating Software Quality of Vendors Using Fuzzy Analytic Hierarchy Process [36]*

The paper has discussed on how software quality of vendors can be evaluative by using fuzzy logic. Kevin K.F. Yuen*, Henry C.W. Lau has used modified fuzzy logarithmic least squares method (LLSM) which can help developers and testers to evaluate the vendors software application and can select the best among of them. This evaluation techniques works under uncertainty and this method is being limited to computational efficiency and complexity.

b) *EESE-An expert system for Software Evaluation [9]*

In this paper it represented a prototype expert system for software evaluation that depicts various aspects of MCDA (multiple criteria decision aid) methodology. This is flexible in problem modeling. This system has covered all important dimensions of software evaluation through the integration of different technologies.

c) *Issues & Factors for Evaluation of Software Quality Models [15]*

This literature considered software metrics and quality models and their issues into very large systems.

A quality model links together the various software metrics and measurement techniques. In this work software quality factors and some of these issues that are considered on large platform.

d) Software Performance Quality Evaluation of MINPHIS Architecture using ATAM [28]

This paper is describing a simple way on guiding the quality of software with the help of ATAM (architecture trade off analysis method) by providing the set of scenarios and their prioritization from brainstorming. The evaluation was based on the developed software Architecture Scenario-Based Performance Quality Model (SASPUM). This evaluation supports on the performance that can be identified as a software quality attribute.

e) Evaluating the quality of software in e-Book using the ISO 9126 Model [6]

This paper is a source of determining an appropriate model for evaluating the quality of software in e-book. According to this the best possible outcome from e-book, the quality of the software used should be of acceptable standard.

f) Principles for evaluating the quality attributes of software architecture [41]

This article has described that software possesses a desired combination of attributes. This includes certain principles which would analysis software architecture to determine if it has those quality attributes or not? Some principles provide a context for providing evaluation approaches such as scenarios, questionnaires and measurement so this paper has discussed these principles for attribute based architecture. A long term goal is to codify these principles into systematic procedures for architecture evaluation.

g) Towards the design quality evaluation of object oriented software systems [8]

This paper presents some advances towards the evaluation of design attributes of object oriented software systems. Here an experiment for collection analysis of those metrics (an OO design metrics set) is explained and other suppositions regarding the design are included. Statistical data is used to collect the data. Results show that some design which was analyzed through experience based techniques is included. It also clears that a number of follow up topics deserve further research.

h) Reusability Evaluation Model for Procedure Based Software Systems [31]

Anything that comes out of software development effort can be easily reused and potentially implemented again but the issue comes that how the reusable components can be identified from existing systems. In this research structural attributes of function

oriented software metrics and reusability of the software is evaluated by various neural network based approaches and taking metrics values as input.

i) Software Metrics-Usability and evaluation of software Quality [14]

This thesis provides a brief review on software quality. Here software metrics and its methods will predict and measure the specified quality factors of software. It discuss about the Quality as defined in ISO, principal elements required for the software quality and software metrics as the measurement technique to predict the software quality. Platform was Java using software metrics (size metrics, complexity metrics). This paper concluded that quality of software can be analyzed, studied and improved by the Usage of software metrics.

j) Establishing a quality model based evaluation process for websites [30]

This paper represents main aspects of ongoing project aimed at defining website which is independent of evaluation process. So this process used a quality model that is defined from existing proposals and general requirements for quality models.

k) A Novel Method for Quantitative Assessment of software Quality [1]

This paper explains with quantitative quality model that needs to be examined throughout the software development lifecycle at each phase. Here the quality goals are set by various measurements and metrics. Under development software is evaluated against for expected value of set of metrics and goals are set by measurements and metrics.

l) Refactoring-does it improve quality? [29]

Refactoring is one of the most important and commonly used techniques of transforming a piece of software in order to improve its quality. It is considered that the increase in quality achieved via refactoring is reflected in the various metrics; measurements on real life systems indicate the opposite. Author have analyzed source code to detect changes marked as refactoring and examine how the software metrics are affected by this process, in order to evaluate whether refactoring is effectively used as a means to improve software quality within the open source community.

m) Software Quality: Definitions and Strategic Issues [7]

This paper is based on two main sections relating to software quality issues. First one examines the quality and their alternatives are suggested. It also includes McCall Richards and Walters's quality model as well Boehm.

n) Evaluating Quality of software systems by Design Patterns Detection [26]

Software industry demands development of products at a very fast rate. So most of the components

are used again and again. Use of design patterns show can speed up the whole processing and basically their effectiveness from design phase to maintenance. This is a metric based study by proposing suitable metrics that could be suitable for all patterns. Higher is the quality of the used design pattern higher will be the quality of the product, with more flexibility towards change and maintenance.

o) An evolutionary approach to evaluate the quality of software systems [35]

An improvement can be carried out by means of defect prediction at the component level of the software systems. This paper discusses an evolutionary computing approach to model defects in complex adaptive software systems based on mathematical elements, graphs, sets, and rough sets, in addition to domain specific rules that are necessary for defect collections and defect analyses. This approach is applied to the evaluation of software quality, and it is fundamental for automation of such an evaluation tool.

p) Evaluating software quality attribute of communication components in an automated guided vehicle system [25]

The architecture of a large complex software system, i.e., the division of the system into components and modules, is crucial because it often affects and limits the quality attributes of the system, e.g., performance and maintainability. In this paper author has evaluated three software components for intra- and inter-process communication in a distributed real-time system, i.e., an automated guided vehicle system. They evaluated three quality attributes: performance, maintainability, and portability. The performance and maintainability are evaluated quantitatively using prototype-based evaluation, while the portability is evaluated qualitatively. This paper resulted that it might be possible to use one third-party component for both intra- and inter-process communication, thus replacing two in-house developed components.

q) Aspect-oriented Software Quality Model: The AOSQ Model [17]

The objective of this paper is to propose a new quality model for AOSD to integrating some new quality attributes in AOSQUAMO model based on ISO/IEC 9126 Quality Model is called Aspect oriented quality model. Analytic Hierarchy Process (AHP) is used to evaluate an improved hierarchical quality model for AOSD.

r) Estimating the Quality of an Object-Oriented software System Using Graph Algorithms [18]

This paper proposes a new set of structural software metrics to estimate the quality of an object oriented software. In the technique firstly the class diagram of the system is mapped into its graph-based structural model. In the developed graph-based

structural model the weights of edges are the dependence degrees of relationships among the classes of the class diagram. Then the graph algorithms are applied based on the longest path algorithm to propose a new set of structural software metrics.

s) Analysis of Quality of Object oriented Systems Using Object Oriented Metrics [27]

In this paper highlights the empirical evaluation of object oriented metrics in C++. They considered two management systems and create a scene. The metric values have been calculated using a semi automated tool. The resulting value comes to be significant about characteristics of the project.

t) Software Quality Estimation through Object Oriented Design Metrics [17]

Software Metrics are used to measure quality in terms of software performance and reliability. In this paper authors mainly studied three object oriented metrics –MOOD metrics, CK metrics and QMOOD metrics and these metrics are useful in determining the quality of any software designed by using object oriented paradigm.

u) Estimating of Quality for Software Components-an Empirical Approach [26]

Component based development is the main concern of this paper. This paper uses AHP for assigning the weight values to the characteristics for the proposed model. These weight values are then used to evaluate the quality contribution of sub characteristics, characteristics and finally consider the overall quality of the component by using appropriate metrics.

v) Estimating Software Quantity with Advanced Data mining Techniques [29]

This paper basically focuses on the use of advanced tool for data mining called Multi-method tool on the real data from Metrics Data Project Data (MDP) Repository. This technique helps to predict the software quality in a software measurement and quality dataset.



Published year	Title name	Quality measurement method	Observation
1996	Towards the design quality evaluation of object oriented software systems	Object oriented model	Only defined for object oriented software system
1999	EESE-An expert system for Software Evaluation	MCDA (multiple criteria decision aid)	Can be used on broad level.
2005	Establishing a quality model based evaluation process for websites	Any Quality model	Does not completely define the low-level and high level characteristic
2006	Estimating Software Quality with Advanced Data Mining Techniques	Data Mining	Effective methodology is used for better outcome
2007	Refactoring-does it improve quality?	Refactoring-techniques of transforming a piece of software	For better efficient result refactoring can be coupled with fuzzy logic
2008	Evaluating Quality of software systems by Design Patterns Detection	Design patterns detection	Performs better when higher quality are introduced
2008	Estimating of Quality for Software Components-an Empirical Approach	AHP	Fuzzy logic can more be enhanced here
2010	Reusability Evaluation Model for Procedure Based Software Systems	Various neural network based approaches	More efficient algorithms can be implemented
2011	A Novel Method for Quantitative Assessment of Software Quality	Measurements and Metrics	Required Output obtained but with more time.
2011	An Evolutionary Approach to evaluate the quality of software systems	An evolutionary computing approach mathematical aspects	
2011	Analysis of Quality of Object Oriented Systems using Object Oriented Metrics	Object oriented metrics	Results are evaluated on the basis of semi automated tool
2011	Software Quality Estimation Through Object Oriented Design Metrics.	Object Oriented Design metrics	Effective for Object Oriented methodology
2012	Software Performance Quality Evaluation of MINPHIS Architecture using ATAM	ATAM (architecture trade off analysis method)	Does not support frequent output
2012	Evaluating the quality of software in e-Book using the ISO 9126 Model	ISO 9126 model	Efficient Output obtained
2012	Estimating the Quality of an Object-Oriented software System Using Graph Algorithms	Graph algorithms	Detected class may need more time for the required output.
2012	Aspect oriented Software Quality Model: The AOSQ Model	Aspect oriented programming language	Supports the quality output

Table 1 : Observations on Literature review

V. CONCLUSION

A comprehensive study of various quality models and their evaluation is presented in this paper. The present study provides the help to users to understand quality models and the use/application of

fuzzy logic approach. Users will also be able to realize the importance of fuzzy logic for not only evaluating the quality but the factors of quality of software system. The observations presented in the paper will help the researchers about different perspectives and areas in the quality prediction of software system.

REFERENCES RÉFÉRENCES REFERENCIAS

1. Bawane, Neelam and Sririshna, C.V, "A Novel Method for Quantitative Assessment Of software Quality", International Journal of Computer Science and Security, Vol 3, Issue 6, 2011.
2. Boehm, Barry W., Brown, J. R, and Lipow, M.: "Quantitative evaluation of software quality", International Conference on Software Engineering, Proceedings of the 2nd international conference on Software engineering, 1976.
3. Crosby, P. B." Quality is free: the art of making quality certain", New York: McGraw-Hill, 1979.
4. Dromey, R. G., "A model for software product quality", IEEE Transactions on Software Engineering, no. 2, pp. 146-163, 1995.
5. Dubey, S.K, Ghosh, Soumi and Rana, Ajay, "Comparison of Software Quality Model: An Analytical Approach, International Journal of Emerging Technology and Advanced Engineering, Vol 2, Issue 2, 2012.
6. Fahmy, Syahrul, Haslinda, Nurul, Roslina, Wan and Fariha, Ziti, "Evaluating the quality of software in e-book using the ISO 9126 model", International Journal of Control and Automation, Vol 3, Issue 2, 2012.
7. Fitzpatrick, Ronan," Software Quality: Definitions and Strategic Issues", Dublin Institute of Technology, 1996.
8. Goulao, Miguel and Esteves, Rita,"Towards the design quality evaluation of object oriented software systems", INESC 1996.
9. Georgiopoulos M.,Dagher I., Heileman G.L., Bebis G., Vlahavas I., Stamelos I., Refanidis I. and Tsoukias A., "ESSE: an expert system for software evaluation", Knowledge based Systems, Vol 12, Issue 4, 1999.
10. Hoyer, R. W. and Hoyer, B. B. Y., "What is quality?" Quality Progress, no. 7, pp. 52-62, 2001.
11. ISO, International Organization for Standardization, "ISO 9001:2000, Quality management systems – Requirements", 2000.
12. ISO, International Organization for Standardization, "ISO 9004:2000, Quality management systems - Guidelines for performance improvements", 2000.
13. ISO, International Organization for Standardization, "ISO 9126-1:2001, Software engineering – Product Quality, Part 1: Quality model", 2001.
14. Jah, Muzamil," Software Metrics-Usability and evaluation of software Quality", Master`s thesis, Department of technology, 2008.
15. Jamwal, Ranbireshwar and Jamwal, Deepshikha, "Issue and factors for evaluation of software quality models", Proceedings of the 3rd National Conference INDIACom, 2009.
16. Khanahmadliravi, N. and Khataee, H.R, "Estimating the quality of an Object-oriented Software System using graph algorithms", International Journal of Computer and Electrical Engineering, Vol 4, Issue 4, 2012.
17. Kumar, Pankaj,"Aspect-oriented software quality Model: The AOSQ Model", Advanced Computing: An International Journal (ACIJ), Vol. 3, Issue 2 2012.
18. Kumar, Rajesh, Sharma, Arun and Grover, P.S, "Estimation of quality for software components-an Empirical Approach", ACM SIGSOFT Software Engineering Notes, Vol. 33, Issue 6, 2008.
19. Kayarvizhy, N. and Kanmani, S., "Analysis of Quality of object oriented systems using Object Oriented Metrics" IEEE Confrence, 2011.
20. Khanna, Pooja, Arora, Deepak, Tripathi, Alpika, Sharma, Shipra and Shukla, Sanchika, "Software quality estimation through Object Oriented Design Metrics", International Journal of Computer Science And Network Security, Vol. 11, Issue 4, 2011.
21. Kokol, Peter, Mertik, Matej, Lenic, Mitja and Stiglic, Gregor, "Estimating Software quality with Advanced Data Mining Techniques", ICSEA`06 Proceedings of the international Conference on Software Engineering Advances, 2006.
22. L.A. Zadeh, "Fuzzy algorithms," Info. & Ctl. Vol. 12, 1968, pp. 94-102.
23. L.A. Zadeh, "Making computers think like people," IEEE. Spectrum, 8/1984.
24. McCall, J. A., Richards, P. K., and Walters, G. F., "Factors in Software Quality", Nat'l Tech. Information Service, no. Vol. 1, 2 and 3, 1977
25. Martensson, F ,Grahn, H and Mattsson, M "Evaluating software quality attributes of communication components in an automated guided vehicle system", Engineering of Complex Computer Systems, 2005.
26. Singh Sandhu, Parvinder, Pal Singh, Parvinder, and Verma, A.K" Evaluating Quality of software systems by Design Patterns Detection.", International conference on Advanced Computer Theory and Engineering, 2008.
27. Sunset.backbone.olemiss.edu/~mistook/sfsysfm.htm-System Software.
28. Soriyan, Abimbola, Gombo, Ishaya and Achimugu, Philip, "Softwar performance quality evaluation of MINPHIS architecture using ATAM", International Journal Of Computer Applications, Vol. 46, 2012.
29. Stroggylos, Konstantinos and Spinellis, Diomidis, "Refactoring-does it improve quality?" IEEE Computer Society, 2007.
30. Trentanni, Gianluca, Biscoglio, Isabella, Fusani, Mario and Lami, Giuseppe "Establishing a quality model based evaluation process for websites".
31. Vashisht, Rajeev, Neeru, Nirvair, Manhas, Sonia and Sandhu, Parvinder S., "Reusability Evaluation Model for Procedure based software systems", 2010.
32. Vlahavas, Ioannis, Stamelos, Ioannis, Refanidis, Ioannis and Tsoukias, Alexis "Knowledge based

evaluation based Evaluation of Software Systems: a case study.

33. Weinstock, CharlesB., Klein, Mark H. and Barbacci, Mario R., "Principles for evaluating the quality attributes of a software architecture", 1997.
34. Wikipedia, <http://en.wikipedia.org>.
35. Wu,B.H "An evaluatioonary approach to evaluate te quality of software systems", 2011.
36. Yuen*, K.F Kevin and Lau, Henry C.W,"Evaluating software quality of vendors using fuzzy analytic heirarchy process", 2008.

