Comparative Study of the Factors that Affect Maintainability

Soumi Ghosh, Sanjay Kumar Dubey, Prof. (Dr.) Ajay Rana
Department of Computer Science & Engineering
Amity School of Engineering and Technology
Amity University, Sec-125, NOIDA, (U.P.), India
soumi1069@rediffmail.com, skdubey1@amity.edu, ajay_rana@amity.edu

***Abstract*—The maintainability of the software system is becoming a very important characteristic due to growth in demand of quality software system. Software maintainability means the ease with which a software system or component can be modified to correct faults, improve performance or other attributes or adapt to a changed environment. The selection of best maintainability model is prime concern for developing quality software system. Every software maintainability model has some sub-characteristics which impact on maintainability directly or indirectly. This paper presents different maintainability models from various standards and quality models define its sub-characteristics and then perform analytical comparison of these sub-characteristics.**

Keywords-maintainability, software system, model, measurement

I. INTRODUCTION

The growth in demand of quality and efficient software system has increased greatly in recent years. The quality of the software system can be measured by using the quality attributes. For such purpose many researchers have proposed quality models to evaluate external software quality attributes viz. usability, reliability, maintainability, reusability, portability etc. Maintainability is a major factor that contributing in the quality of software system. Software maintainability is defined as "the ease with which a software system or component can be modified to correct faults, improve performance or other attributes changed environment" [28]. It is also defined as "the capability of software product to be modified". Modifications may include corrections, improvements or adoption of the software to changes in environment, and in requirements and functional specifications [29].

Maintainability is closely related to the term software maintenance because maintainability means how easy it is to perform maintenance on the system. The process of changing software after it has been delivered and in use is called software maintenance [26]. Maintenance is also defined as the process that occurs when software undergoes modifications to code and associated documentation due to an error or the need [31]. Maintenance consumes 40% to 80% of software costs and is therefore probably the most important phase of life cycle of software [11, 36, 50]. The 60% maintenance costs come from making enhancements, which is something that makes the systems provide additional value [11, 50]. Software maintenance is classified into four types: corrective, adaptive, perfective and preventive [59]. Corrective maintenance refers to fixing a program. Adaptive maintenance refers to modifications that adapt to changes in the data environment such as new product codes or new file organization or changes in hardware of software environment. Perfective maintenance refers to enhancements: making the product better, faster smaller, better documented; clear structured with more functions or reports. Preventive maintenance is defined as the work that is done in order to try to prevent malfunctions or improve maintainability. In [10], it is pointed out that approximately 85% of all maintenance costs come from adaptive and perfective maintenance efforts. High maintainability is important in the process of developing a software system, since it is a way to lower the cost for maintenance activities. Despite the importance of software maintenance, it is unfortunate that little work has been done to identify models and sub-characteristics of software maintainability. Thus, it is necessary to analyze software quality by considering maintainability factors effectively throughout the entire software lifecycle.

It is well known fact that maintainability is closely related to concept that it is more difficult to quantify. In view of concept this paper presents detail quantification and review of various maintainability models in broader way and identifies the various sub-characteristics that impact on maintainability directly or indirectly. The reason for this review is to show how researcher's view of maintainability has changed in more than four decades. The main objective of this paper is:

a) to decompose maintainability in different sub-characteristics;

b) to compose sub-characteristics; and

c) to identify those sub-characteristics, which perform greater impact on maintainability.

II. LITERATURE SURVEY

Since last four decades, a wide range of maintainability models have been proposed based on several parameters and sub-characteristics. In [34], the proposed model defines various software characteristics which were categorized into three parts product revision, product operation and product transition. In this model maintainability was classified into various sub- characteristics as simplicity, conciseness, modularity, self-descriptiveness. In [13], authors presented a quality model that primarily focuses on maintainability. Its sub-characteristics include testability, understandability and modifiability. In [18], a model was developed, which is basically a framework of software maintainability. This model shows that each software program is separately evaluated and consists of set of software components called modules. This model was based on modularity, descriptiveness, consistency, simplicity, expandability and instrumentation sub-characteristics. The model given by [62] states that software complexity is the primary factor that affects the three sub-characteristics understandability, modifiability and testability. Software complexity is closely related with modularity, information hiding, coupling and cohesion. In [24], proposed model considered the different design characteristics and this model pointed out that maintainability is a measure of software characteristics as source code readability, documentation quality and cohesiveness among source code and documents. This model shows integration of three different characteristics i.e. average no. of variables, average cyclomatic complexity and comments ratio. As per this model, the lower comment ratio, the better is the readability and maintainability. In [60], a model that is basically used in software maintenance environment acts as a basis for automated tools for modifying an existing software system. The various phases of software maintenance i.e. understanding program, generating maintenance, accounting for ripple effect and revalidation are described in this model. Capability Maturity Model (CMM) [15] is used for improving organization business areas like software maintenance, project management etc. The five maturity levels viz initial, repeatable, defined, managed and optimizing are defined in this model. The various common features of CMM are commitment to perform, ability to perform, activities performed, measurement and analysis and verifying implementation. The objective of this model is to show what is needed to be done for reliable maintenance of software rather than how it is to be done.

In [32], changeability, stability, testability, analyzability and maintenance compliance is defined as the main sub-characteristics of software maintainability. The model by [14] is based on software quality which comprises of internal and external qualities certain specific characteristics. Maintainability is considered as an important aspect of the internal qualities in this model. The model [27] is basically treated as a standard for software maintenance. Various sub-characteristics correctability, expandability, implementation, portability, process delivery, self descriptiveness, simplicity and testability of maintainability are defined in this model. In [48], a hierarchical model and multidimensional assessment technique was proposed, which evaluates the software maintainability by categorizing the characteristics in the form of hierarchical levels. In this model a polynomial equation is generated in which system maintainability is represented as function of related metric attributes used for developing of non linear polynomial maintainability assessment models. In [21], a reusability and maintainability model was proposed for C++ code. The sub-characteristics of maintainability in this model are consistency, self-descriptiveness, modularity, simplicity and testability. In [22], an important quality model was developed on software maintainability having different sub-characteristics like self-descriptiveness, modifiability and testability. The model by [23] may be used to highlight the need to improve the quality of the product so that proper and efficient maintenance is feasible without much difficulty. It can also be used to develop a measurement of maintainability before completion and delivery of a software product. In [7], a predictive model was proposed in order to evaluate the impact of object-oriented design on software quality characteristics maintainability and reliability. Various metrics were used for measuring object-oriented design characteristics called MOOD (Metrics for Object-Oriented Design). The model by [41] discussed relationship between various sub-characteristics and dimensions of maintainability of object-oriented systems and also how to measure them. Four sub-characteristics of maintainability viz. comprehensibility, ease of impact analysis, localizability and testability were defined in this model.

In [39], the model was developed to incorporate maintenance knowledge and experience in the field of planning, design, procurement, construction start–up and operation of facilities. A software maintainability prediction model was developed for assembling, evaluating and predicting software maintainability [58]. The various steps involved in the development process of this model are selection of metrics, data collection, correlation analysis, regression analysis and model development. Stochastic model [55] studies the effects of maintenance activities on software and it is based on theory of graphs to represent software system. It is also used to give some insight on development process based on refactoring including extreme programming. Software deterioration and Maintainability Model (SMM) developed by [42] considers both aspects of software maintainability and deterioration. It mainly focuses on how software should be maintained in order to prevent it from deterioration. This model shows the correlation between maintainability and its dimensions i.e. effort and change. In [20], Generic, Multilayered and Customizable Model was proposed that shows the need of different project developers in a very dynamic and flexible manner so as to enable every project developer to construct his own model reflecting the emphasis given for each attribute or requirement. The model by [49] validated three

metrics that is used in measuring object-oriented design complexity. These metrics are integration level, interface size and operation argument complexity. In this model the dependent variable maintenance performance is expressed in terms of independent variables design complexity, maintenance task and programmer ability. It also experimentally proved a relationship between the design complexity metrics and maintenance performance, so any of these three metrics can be used to predict maintenance performance. In [45], authors suggested that definition of quality attributes of maintainability is not only important but also the techniques meant for promotion of maintainability. According to them software may be divided into three abstractions namely system, architecture and component dimensions. This model also discussed impact of quality attributes on each of these dimensions. In [43], authors developed a maintainability model by using weighted sum method. This model shows that qualities of software design heavily affects on qualities of software ultimately developed. The two sub-characteristics of maintainability: understandability, modifiability are used in order to construct the object-oriented maintainability model. In [37], metric based software estimation model was developed basically to estimate adaptive maintenance efforts in terms of man hours. The model was derived using regression analysis based on the data collected on past software projects. In [2], a Web Application Maintainability Model (WAMM) was developed for traditional systems, which is adapted to architectural and structural web applications. In [38], maintainability prediction model was developed using new measure and two datasets. It accurately predicts the maintainability of 83% of the modules. In [1], author proposed a Software Maintenance maturity model (SMmm) which follows the Capability Maturity Model (CMM) for improving software maintenance which ultimately leads to consumer satisfaction. The six maturity levels are incomplete, performed, managed, established, predictable process and optimizing process are defined in this model. In [40], ANN (Artificial Neural Network) technique was used to construct maintainability models by using object-oriented metrics. This ANN model estimates the relationship between eight object-oriented metrics and maintainability efforts. A two dimensional model was developed by [8] integrates as well as explains a set of relevant criterions and describes their impact on actual maintenance activities. In [44], a framework for software maintainability has been described considering two aspects like product and process. In [51], a framework was also presented for assessing maintainability of software system by using Enterprise Architecture Models (EAM) which provides the diagrammatic description of the systems and their environment. The model proposed by [46] is a modified version of ISO-9126 [32] which provides a hierarchical tree structure of maintainability, its sub-characteristics and mapping between maintainability sub-characteristics and object-oriented source code metrics. A new practical model was developed by [25] which includes a well chosen source code measures. This model is also modified version of ISO-9126 [32] where all features are modified and system level characteristics are mapped to properties on the level of source code. This model has also overcome from the problem of Maintainability Index (MI). Fault detection / correction model developed by [57] aims at providing a framework to deal with both fault detection and correction process simultaneously and to evaluate the software maintainability which is a quantitative measure for software fault correction process. Fuzzy model developed by [65] is used to predict software maintenance by using four factors such as Readability of Source Code (RSC), Comments Ratio (CR), Documentation Quality (DQ), Understandability of Software (UoS) and Average Cyclomatic Complexity (ACC). Predelivery maintenance model was prepared by [54] for the purpose of predelivery maintenance process and in order to conduct post delivery maintenance step successfully. Maintainability Prediction Model was developed by [63] in order to predict object-oriented software maintainability. This model has presented the ongoing work using Projection Pursuit Regression (PPR). In [53], authors developed maintainability estimation model for object-oriented software in design phase, which estimates the maintainability of class diagrams in terms of their understandability and modifiability. The maintainability model [56] defined for object-oriented system by using CK Metrics. This model shows the relationship between maintainability and object-oriented metrics.

A. Sub-characteristics Definitions

The sub-characters play very important role in the measurement of quality factors. The quantification of the quality factors provides the help in the assessment of the overall quality of the software system. The definitions of various sub-characteristics from the literature survey of maintainability are defined in the Table 1.

III. METHODOLOGY

We undertook our methodology to explore published definitions of maintainability. We searched a citation index database Scopus, for specific publications with maintainability definitions and quantification. We selected 65 titles under the Computer Science and Information Systems category. For the searches, each journal title was entered into the “source title” field for Scopus. In addition, we combined the title search with the phrase “maintainability definition and maintainability attributes” in the “titles, abstracts and keywords” option in Scopus. The searches retrieved a total of 287 records. Individually we looked through the 287 articles to identify formal definitions of the sub-characteristics of ‘maintainability’. From the articles that contain definitions cited from other sources, we located those cited sources and included them in our review. The Table 2 gives the comparative analysis of sub-characteristics of maintainability.

TABLE I. THE DEFINITION OF SUB-CHARACTERISTICS OF MAINTAINABILITY

Sub-characteristics	Definitions	References
Accuracy	The capability of the software product to provide the right or agreed results or effects with the needed degree of precision.	[29]
Adaptability	The capability of the software product to be adapted for different specified environments without applying actions or means other than those provided for this purpose for the software considered.	[29]
Analyzability	A set of attributes for predicting the maintainer's or user's spent effort or spent resources in trying to diagnose for deficiencies or causes of failure, or for identification of parts to be modified in the software product.	[30]
Augmentability	The ability of the model to accommodates expansion in component computational functions or data storage requirements.	[9]
Availability	The degree to which a work is operational and available for use as a product or to uses.	[16]
Changeability	The characterization of the amount of effort to change a system.	[30]
Completeness	The degree to which full implementation of required function has been achieved.	[52]
Complexity	The degree to which a component or system has a design and internal structure that is difficult to understand, maintain and verify.	[61]
Comprehensibility	The quality of being able to be understood, intelligibility and conceivability.	[5]
Conciseness	The compactness of the program in terms of line of code.	[52]
Consistency	The use of uniform design and documentation techniques throughout the software development project.	[52]
Correctability	The ease with which minor defects can be corrected between major releases while the application or component is in use by its users.	[16]
Delivery	Any (work) product that must be delivered to someone other than the (work) product's author.	[61]
Documentation	Testing the quality of the documentation, e.g. user guide or installation guide.	[61]
Durability	It is a measure of product's life. All products deteriorate and degrade with time and /or usage.	[17]
Efficiency	The capability of the software product to provide appropriate performance, relative to the amount of resources used understated conditions.	[29]
Effort	The effort to change or modify a software product in order to adapt it to other environment or other applications different from that was designed.	[35]
Expandability	The degree to which architectural, data or procedural design can be extended.	[52]
Extensibility	The ease with which an application or component can be enhanced in the future to meet changing requirements or goals.	[16]
Flexibility	The effort required modifying an operational program.	[35]
Impact analysis	The activity of identifying which object to modify to accomplish a change, i.e., estimating the potential consequences of carrying out a change.	[4]
Implementation	Systematically structuredness approach to effectively integrate software based service or component into workflow of an organizational structure or an individual end user.	[64]
Instrumentation	The degree to which the program monitors its own operation and identifies errors those do occur.	[52]
Integrability	The ability to make the separately developed components of the system work correctly together.	[3]
Localization	The process of adapting internationalized software for a specific region or language by adding locale-specific components and translating text.	[64]
Maintainability compliance	The capability of the software product to adhere to standards or conventions relating to maintainability.	[29]
Modifiability	Corrections, improvements or adaptations of the software to changes in environment and in requirements and functional specifications.	[29]
Modularity	The functional independence of program components.	[52]
Perfectiveness	The modification of a software product after delivery to improve performance or maintainability.	[5]
Portability	The capability of the software product to be transferred from one environment to another.	[29]
Programming Language	A code written in a language where programmer not familiar with the results in difficulty in maintaining the reused code.	[23]
Readability	A set of attributes related to the difficulty in understanding software components source and documentation.	[33]
Reusability	The ease with which an existing application or component can be reused.	[16]
Self descriptiveness	It relates to the matter whether the model contains sufficient information for a reader to determine or verify its objectives, assumptions, constraints, inputs, outputs components and revision status.	[9]
Simplicity	The degree to which a program can be understood without any difficulty.	[52]
Stability	The capability of software product to avoid unexpected effects from modifications of software.	[29]
Standardization	A set of programming standards used as a guide in code. Programming standards, guidelines and practices used in writing program contributes to its readability; understand ability and directly affecting modifiability and maintainability.	[23]
Testability	The capability of the software product to enable modified software to be validated.	[29]
Traceability	The ability to trace a design representation or actual program component back to requirements.	[52]
Understandability	The capability of the software product to enable the user to understand whether the software is suitable and how it can be used for particular tasks and conditions of use.	[29]

TABLE II. COMPARATIVE SUMMANRY OF SUB-CHARACTERISTICS OF MAINTAINABILITY

Models → / Sub-characteristics ↓	J. A. McCall [34]	B. W. Boehm [13]	D. Peercy [18]	W. Harrison *et al.* [62]	H. Sneed *et al.* [24]	S. S. Yau *et al.* [60]	C. Ghezzi *et al.* [14]	ISO 9126-1 [32]	IEEE Std. [27]	E. Karlsson [21]	G. R. Dromey [22]	H. Khairuddin *et al.* [23]	L. Briand [41]	R. Land [42]	M. Matinlassi *et al.* [45]	M. J. Kiewkanya *et al.* [43]	M. K. Mattsson [44]	B. Manfred [8]	R. Lagerstorm [51]	I. Heitlager [25]	S. W. A. Rizvi *et al.* [53]
Accuracy																			X		
Adaptability							X														
Analyzability								X										X		X	
Augment ability		X															X				
Availability																			X		
Changeability								X						X						X	
Cohesiveness				X	X														X		
Complexity						X						X									
Comprehensibility													X								
Conciseness	X	X																			
Consistency		X	X							X									X		
Correctability							X		X												
Delivery																		X			
Documentation					X	X															
Durability																			X		
Effort														X			X				
Expandability			X						X								X				
Extensibility						X															
Flexibility															X						
Impact Analysis													X					X			
Implementation									X									X			
Instrumentation			X																		
Integrability															X			X			
Level of validation and testing												X									
Localizability													X								
Maintainability Compliance								X													
Modifiability		X		X							X				X	X	X		X		X
Modularity	X		X	X						X		X									
Perfectiveness							X														
Portability									X						X				X		
Process Delivery									X												
Programming Language												X									
Readability					X							X					X		X		
Reusability															X						
Self-descriptiveness	X	X	X			X			X	X	X										`
Simplicity	X		X						X	X											
Stability						X		X											X	X	
Standardization												X					X		X		
Testability		X		X		X		X	X		X		X		X		X	X		X	
Traceability												X					X		X		
Understandability		X		X												X					X

IV. CONCLUSION AND FUTURE SCOPE

We undertook our methodology to explore published definitions of maintainability. We searched a citation index database Scopus, for specific publications with maintainability definitions and quantification. We selected 65 titles under the Computer Science and Information Systems category. For the searches, each journal title was entered into the "source title" field for Scopus. In addition, we combined the title search with the phrase "maintainability definition and maintainability attributes" in the "titles, abstracts and keywords" option in Scopus. The searches retrieved a total of 287 records. Individually we looked through the 287 articles to identify formal definitions of the sub-characteristics of 'maintainability'. From the articles that contain definitions cited from other sources, we located those cited sources and included them in our review. The Table 2 gives the comparative analysis of sub-characteristics of maintainability.

REFERENCES

[1] A. Alain, J. H. Hayes, A. Abran and R. Dumke, "Software Maintenance Maturity Model (SMmm)-The software maintenance process model", Journal of Software Maintenance and Evolution Research and Practice, vol. 17, no. 3, pp. 197-223, 2005

[2] A. D. L. Gieuseppe, A. R. Fasolin, P. Tramontana and C. A. Visaggio, "Towards the definition of a maintainability model for web applications", Proceedings of the Eighth European Conference on Software Maintenance and Reengineering (CSMR'04) pp. 1534-5351, 2004

[3] A. Evesti , Quality-oriented software architecture development, VTT Technical Research Centre of Finland, 2007

[4] A. Samuel , "Software Maintenance An approach to Impact Analysis of Objects Change, Software Practice and Experience", vol. 25 no. 10, pp. 1155-1181, 1995

[5] ABS Guide, Integrated Software Quality Management (ISQM), 2011

[6] ANSI/IEEE Standard 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology, 1990

[7] B. F. Abreu and W. L. Melo, Evaluating the Impact of Object Oriented Design on Software Quality, 1996

[8] B. Manfred, D. Florian and P. Markus, Demysterifying Maintainability, Proceedings, 2006

[9] B. Osman, Credibility Assessment of Simulation Results, Proceedings of the 18th conference on winter simulation, pp. 38-44, 1986

[10] B. P. Lientz, E. B. Swanson, Software Maintenance Management, Addison- Wesley, Reading , MA, 1980

[11] B. P. Lientz, E. B. Swanson and G. E. Tompkins, Characteristics of Application Software Maintenance, Communication of ACM, vol. 21, pp. 466-471, 1978

[12] B. R. Grady and L. C. Deborah, Software metrics establishing a companywide program, Prentice -Hall, pp. 159, ISBN 0138218447, NJ, USA, 1987

[13] B.W. Boehm, Characteristics of Software Quality, TRW Series of software Technology, Amsterdam, North Holland, 1978

[14] C. Ghezzi, C. M. Jazayeri and D. Mandrioli, Fundamental of Software Engineering, Prentice-Hall, NJ, USA, 1991

[15] D. E. Lowe and M. C. Guy, Implementing the Capability Maturity Model (CMM) for software, 1987

[16] D. G. Firesmith, Common Concepts Underlying Safety, Security and Survivability Engineering, Technical Note CMU/SEI- 2003-TN-033, Software Engineering Institute, Pittsburg, Pennsylvania, pp. 1-75, 2003

[17] D. N. P. Murthy, W. R. Blischke, Warranty management & product manufacture, Google books, Business & Ecomomics, pp. 1-302, 2006

[18] D. Peercy and T. Paschich, Software Maintainability Analysis Program User's Manual, BDM/TAC-78-697-TR, 1978

[19] D. W. Drew, Tailoring the Software Engineering Institute's (SEI) Capability Maturity Model (CMM) to Software Sustaining Engineering Organization, Proceedings in Conference on Software Maintenance, Orlando, FL, pp. 137-144, 1992

[20] E. Georgiadou, GEQUAMO – A Generic, Customizable, Software Quality Model, International Journal of Cybernetics, vol.11, no.4, pp. 313-323, 2003

[21] E. Karlsson, Software Reuse- A Holistic Approach, John Weiley & Sons, England, 1995

[22] G. R. Dromey, A model for software product quality, IEEE Transaction on software Engineering, 1995

[23] H. Khairuddin and K. Elizabeth A software maintainability attributes model, Malaysian Journal of Computer Science, vol. 9, no. 2, pp. 92-97, 1996

[24] H. Sneed and A. Mercy, Automated Software Quality Assurance, IEEE Transaction of Software, 1985

[25] I. Heitlager, T. Kuipers and J. Visser, A Practical Model for Measuring Maintainability Software Improvement Group, Netherlands, 2007

[26] I. Sommerville (1996): Software Engineering, 5th Ed. Addison Wesley, USA , 1996

[27] IEEE Standard for Software Maintenance, Software Engineering Standards Subcommittee of the IEEE Computer Society, 1993

[28] IEEE Std. Glossary of software Engineering Terminology, Report IEEE Std. 610. 12-1990, IEEE Implications on XP Process, Proceedings of First International Conference on Extreme Programming & International workshop on software quality, ACM, NY, USA, 1990

[29] ISO/IEC 9126-1, Institute of Electrical and Electronics Engineers, Part1: Quality Model, 2001

[30] ISO/IEC TR 9126-3, Software Engineering Product Quality, 2002

[31] ISO/IEC, ISO/IEC 12207, Information Technology-Software Life Cycle Process, 1995

[32] ISO-9126, Software Product Evaluation Quality Characteristics and Guidelines for their Use, 1991

[33] J. A. Lowell, Software Evolution, the Software Maintenance Challenge, John Wiley and sons, 1951

[34] J. A. McCall, P. K. Richards and G. F. Walters, Factors in Software Quality, Springfield, VA, National Technical Information Service, 1977

[35] J. E. Gaffney, Metrics in Software Quality Assurance, no. 81, pp. 126-130, ACM Press, 1981

[36] J. Foster, Cost Factors in Software Maintenance, in Computer Science Department: University of Durham, NC, http:// www. Jsjf.demon.co.uk/thesis/Thesis.html, 1993

[37] J. H. Hayes and L. Zhao, Maintainability Prediction: a Regression Analysis of Measures of Evolving Systems, Proc. 21st IEEE International Conference on Software Maintenance, pp. 601-604, 2004

[38] J. H.Hayes, S. C. Patel and L. Zhao, A metrics based software maintenance effort model, Proc. 8th European Conference on Software Maintenance and Reengineering (CSMR'04), IEEE Computer Society, pp. 254-258, 2005

[39] J. R. Meier and J. S. Russell, "Model Process For Implementing Maintainability", Journal of Construction Engineering and Management, vol. 126, no. 6, pp. 21590, 2000

[40] K. K. Aggarwal, Y. Singh, A. Kaur and R. Malhotra, "An Application of Artificial Neural Network for Predicting Maintainability of Object Oriented Metrics", Transactions on Engineering, Computing and Technology, 15, pp.285-289, 2006

[41] L. Briand and F. Lanubile, Proceedings of 2nd Intl. Workshop on Empirical Studies of Software, 1997
[42] L. Rickard, Software Deterioration and Maintainability-A Model Proposal, Malardalen University, In Proceedings of 2nd Conference on software Engineering Research & Practice, Sweden (SERPS), 2002
[43] M. J. Kiewkanya and N. Muenchaisri, A Methodology for Constructing Maintainability Model of Object-oriented design, Proc. 4th International Conference on Quality Software, pp. 206-213, 2004
[44] M. K. Mattson, A. V. Deursen, R. Reiger, G. Canfora, T. Ihme, T. Engel, D. Chiorean, M. M. Lehman and J. Wernke, A Model of Maintainability Suggestion for Future Research, In Proceedings of Software Engineering Research & Practice, pp. 436-441, 2006
[45] M. Matinlassi and N. Eila, The Impact of Maintainability on Component Based Software System, Proceeding of the 29th EUROMICRO Conference New Waves in system Architecture, 2003
[46] P. Antonellis, D. Antoniou, Y. Kanellopoulous, C. Makris, E. Theodoridis, C. Tjortjis and N. Tsirakis, A data mining methodology for evaluating maintainability acc. To ISO/IEC 9126 Software Engineering Product Quality Standard, Proc. 11th IEEE Conference on Software Maintenance & Reengineering (CSMR), 2007
[47] P. Oman and J. Hagemeister, Metrics for Assessing a Software System's Maintainability, Proceedings of IEEE International Conference on Software Maintenance, IEEE Computer Society Press, Los Alamitos, CA, 1992
[48] P. Oman and J. Hagemeister, Construction and Testing of Polynomial Predicting Software, Maintainability Systems and Software, vol. 24, no. 3, pp. 251-256, 1994
[49] R. K. Bandini, V. K. Vaishnavi and D. E. Turk, "Predicting Maintenance Performance Using Object Oriented Design Complexity Metrics", IEEE Transactions on Software Engineering, 29(1), pp.77-87, 2003
[50] R. L.Glass, Facts and Fallacies of Software Engineering, Addison– Wesley, ETH Zurich Chair of Software Engineering, Boston, MA, vol. 2, no. 1, pp. 1-195, 2003
[51] R. Lagerstorm, Analyzing System Maintainability using Enterprise Architecture Models, Royal Institute of Technology, Osquldas vag 12, 10044 Stockholm, Sweden, 2007
[52] R. S. Pressman, Software Engineering a practitioner's Approach, Mc Graw-Hill Inc., 1992
[53] Rizvi, S. W. A, R. A. Khan, "Maintainability Estimation Model for Object-Oriented Software in Design Phase", Journal of Computing, 2(4), April 2010.
[54] S. A. Khan, M. K. Mattson and T. Tyrberg, Comparing EM3 Predelivery Maintenance Model with its Industrial Correspondence, International Multiconference on Computer Science and Information Technology (IMCSIT), pp. 573-582, 2009
[55] S. Focardi, M. Marchesi and G. Succi, A Stochastic Model of Software Maintenance and its Implications on XP Process, Proceedings of First International Conference on Extreme Programming & Flexible Processes in Software Engineering (XP 2000), Cagliari, Italy
[56] S. K. Dubey and Ajay Rana, "Assessment of Maintainability Metrics for Object Oriented Software System", ACM SIGSOFT Software Engineering Notes, 36 (4), September, 2011.
[57] S. Kazuya, R. Koichiro, T. Dohi and O. Hiroyuki (2007): Quantifying Software Maintainability Based on a Fault Detection/Correction Model, 13th IEEE International Symposium on Pacific Rim Dependable Computing, 2007
[58] S. Muthanna, S. Kontogiannis, K. Konnambalam and K. Stacey (2000): A Maintainability Model for Industrial Software Systems Using Design Level Metrics, Proceedings 7th Working Conference on Reverse Engineering (WCRE'00), Brisbane, Australia, pp. 248-256, 2000
[59] S. O. Olatunji, Z. Rasheed, K. A. Sattar, A. M. Al- Mana, M. Alshayeb and E. A.El-Sebakhy, "Extreme Learning Machine as Maintainability Prediction Model for object-oriented Software Systems", Journal of Computing, vol. 2, no. 8, pp. 49-56, 2010
[60] S. S. Yau and J. S. Collofello, "Design Stability Measures for Software Maintenance, Software Engineering", IEEE Transactions, IEEE Computer Society, vol. SE-11, no. 9, pp. 849-856, 1985
[61] Standard glossary of terms used in software testing, version 2.1, Glossary Working Party International Software Testing Qualifications Board, Netherlands, 2010
[62] W. Harrison, K. Magel, R. Kluczny and DeDock, Applying software complexity metrics to program maintenance, IEEE Computer, vol. 15, pp. 65-79, 1982
[63] W. L. Jin, H. X. Xin, N. Z. Yuan and K. H. Wen, Predicting object-oriented software maintainability using projection by pursuit regression, 1st International Conference on Information Science and Engineering (ICISE), Nanjing, pp. 3827-3820, 2009
[64] Wikipedia (2011), www. wikipedia.org, 2011
[65] Y. Singh, P. K. Bhatia and O. Sangwan, "Predicting Software Maintenance using Fuzzy Model", ACM SIGSOFT, Software Engineering Notes, vol. 34, 2009