ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 9, September 2016

Impact Analysis of Development over Software Quality and Maintainability

Himangi[1], Surender Singh[2]

M.Tech Scholar, Department of CSE, OITM, Juglan Hisar, India[1]

Assistant Professor & HOD, Department of CSE, OITM, Juglan Hisar, India[2]

ABSTRACT: As software development is a complex process, with high variance on both methodologies and objectives, it is difficult to define or measure software qualities and quantities and to determine a valid and concurrent measurement metric, especially when making such a prediction prior to the detail design. Another source of difficulty and debate is in determining which metrics matter, and what they mean.
The practical utility of software measurements has thus been limited to narrow domains where they include: Schedule, Size/Complexity, Cost and Quality etc. Common goal of measurement may target one or more of the above aspects, or the balance between them as indicator of team's motivation or project performance. this work will explore the recent development related to software metrics. Main objective of research work is to analyze the software quality measurements and its requirements and to explore develop a quality metrics for each attribute, mentioned above. For experiment purpose, we will use a the relationship between various factors i.e. program size, ownership and developer quality and to software metric tool, called CCCC , in order to explore the attributes of software.

KEYWORDS: Software Quality, Metric, Program Size, SLOC

I. INTRODUCTION

As software development is a complex process, with high variance on both methodologies and objectives, it is difficult to define or measure software qualities and quantities and to determine a valid and concurrent measurement metric, especially when making such a prediction prior to the detail design. Another source of difficulty and debate is in determining which metrics matter, and what they mean.[3][4] The practical utility of software measurements has thus been limited to narrow domains where they include:

- Schedule
- Size/Complexity
- Cost
- Quality

Common goal of measurement may target one or more of the above aspects, or the balance between them as indicator of team's motivation or project performance.

Main objective of research work is to analyze the software quality measurements for the followings:

- To explore the relationship between program size, ownership and developer quality
- To develop a quality metrics for each attribute, mentioned above

II. LITERATURE SURVEY

Mohsin et al. [34] suggested an approach which explores effective code comprehension by combining Software metrics and technique called Program Slicing. Program slicing is static program analysis process for code automation which can develop efficient measures for coupling, cohesion, complexity. Such novel design of software metrics with analytical approach can insure reliable development of software system.

A. Abdi et al. [35] proposed some security metrics in different software development phases and validates them based on some standardized criteria. Different phases have different metrics that are defined based on their results and

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 9, September 2016

products. By using proposed security metrics during software development cycle, the final product will be secure and qualified.

K. Akingbehin et al. [36] presented a structured unifying framework for software metrics (numerical software measurements), based on the three "primary metrics" of function points (FP), person-months (PM), and lines of code (LOC). The framework is based on a layered model, with the three primary metrics constituting the lowest layer. An important property of the primary metrics, referred to as the "convertibility property" is that a primary metric can easily be converted to another primary metric. Time is also included in this layer as a fundamental (not necessarily software) primary metric. The second layer consists of general-purpose metrics such as productivity measures, which are computed from the primary metrics, and the third layer consists of specialpurpose metrics such as reliability and quality measures. This third layer is inherently extensible. The framework readily lends itself for use in both instructional and practitioner environments.

Franca, J.M.S et al. [37] proposed a solution to solve Code scattering and code tangling issues, based on specific software construction known as aspect. Aspect-oriented programming (AOP) has been widely studied since its introduction with the promise of improving modularization by addressing crosscutting concerns. Few studies on empirical evaluation of the benefits of aspect-oriented paradigm were published. Results presented in these studies are frequently subjective, and some studies are non-conclusive. In addition, these studies are based on the implementation of only one or two crosscutting concerns into aspects, and the evaluation is based on few software metrics. In this article, the evaluation of AOP implementation through software metrics is proposed. The main idea is to implement crosscutting concerns as aspects, with focus on those that were not given properly attention in the literature.

Anjana Gosain et al. [33] explored the different types of dynamic software metrics for object oriented approaches unde the constraints of validation and software quality. They sub divided the metrics into different categories i.e. size which can be defined as the size of the code on disk as well as memory size occupancy by same code at run time, complexity which can be defined by frequency, cohesion and coupling can describe the inside and outside bounding/dependency of existing modules over each other at run time, inheritance describes the frequency of interaction between parent and child classes and polymorphism metric can describe behavior index as ratio of polymorphic and non polymorphic dispatches to be executed. They also represented the Software quality attributes, metric types and their validation graphically.

Chandan Kumar et al. [17] represented a error estimation scheme for software based on BBN metrics. Authors focused on the early development stages at which it is quite hard to use metrics and considered the reliability and uncertainty in these phases. Analysis results show accuracy of proposed method in terms of error detection at early stages.

Tao Yue et al. [18] developed a framework to produce quality metrics for MOF meta models to measure the quality of models. Comparison of newly produced metrics with manually defined quality metrics using UML classes and sequence diagrams shows that it automatic produced metrics are more efficient and can ensure the software quality in more accurate way. Proposed work can be extended to develop metric for MOF-based languages.

Greiler, M. et al. [19] explored the concept of ownership metrics and identify the relationship between quality and ownership using a specific product family and they showed the relationship between them in the presence of errors. Their analysis proved the correlation between ownership and frequency of fixed bugs at file/directory level. They defined a prediction model for classification of defective and non-defective entities. Proposed work can be extended to analyze dynamic changes in ownership and its impact over code quality using different product teams.

Yilin Qiu et al. [20] explored a metric related to developer quality using various open source software and considered some important facts related to quality contribution distribution, evaluation and its correlation and impact over software. They also investigated the code complexity of generated code and its ownership for quality measurement. Finally they concluded that code complexity and frequency of errors in code are also effected by the developer's quality. Current research work can be further extended for defect prediction and software process management.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 9, September 2016

Sandeep Kaur et al. [21] explored the relation between size and object oriented metrics under different constraints i.e. abstractness, bug density, refactor method and dead code etc. Their analysis shows negative relationship of abstractness and bug frequency with size and object oriented metrics which is positive for refactor method and dead code. Accuracy of their results also depends upon the metric tools used for analysis purpose. Current research work can be extended for big data and other open source software.

Gulnara Zhabelova et al. [22] developed a metric for IEC 61499 function blocks which can be used for analysis of power system protection. They focused on suitable tools to be used to estimate the metrics i.e. Line of Code, Operands/ Operators, DIT, NOC, Fan-In/Fan-Out etc. However it is not capable enough to identify the hidden complexity

Sungdo Gu et al. [23] focused on the cost of development and maintenance w.r.t. bug density per LOC and presented a method which can select subsets of metrics on the basis of relationship and those can be further validated using Poisson Regression Model. As per the results, network based approach can easily recognize the relations using object oriented metrics and it can optimize the cost of different activities i.e. development, maintenance and can enhance efficiency. Proposed scheme can be extended to explore the relations between feature/response variable.

III. PROPOSED SCHEME

Code ownership, software quality and maintainability

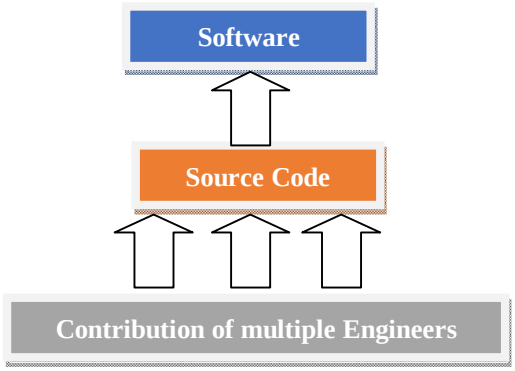


Figure: 4.1

Code Ownership can be defined as the percentage of changes made by each contributor. If anyone has made the highest changes that means percentage of Ownership increases over the developed code.'
Ownership can be subdivided in to two different categories:

- Total percentage of Ownership
- Component's Ownership

Total percentage of Ownership: $\frac{\sum contribution\ made}{\sum Total\ Contribution}$

Component's Ownership: $\frac{\sum contribution\ made\ for\ a\ specific\ module}{\sum Total\ Contribution\ \ made\ for\ a\ specific\ module}$

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 9, September 2016

Each category has its own impact over the software quality and its maintainability.

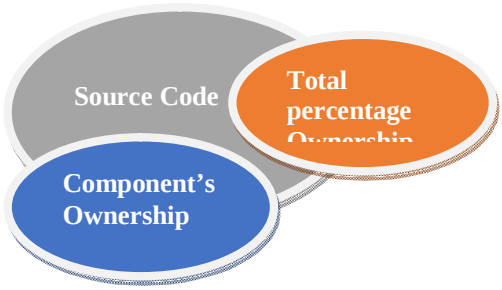


Figure: 4.2 Source code ownership

Code quality can be defined as

$$Quality = 1 - \frac{\sum Bugs\ introduced}{\sum contribution\ made}$$

$$\text{ontribution made for each commit} = \sum_{K=0}^{n\ \ i} \text{LOC(Commit)}$$

Where LOC defines the program size.
Contribution can be further subdivided:

- MINOR Contribution: Minimum number of changes made to code which can alter the functionality of a module
- MAJOR Contribution: Major number of changes made to code which can alter the functionality of a module
- TOTAL Contribution: Overall changes made to code which can alter the functionality of a entire software

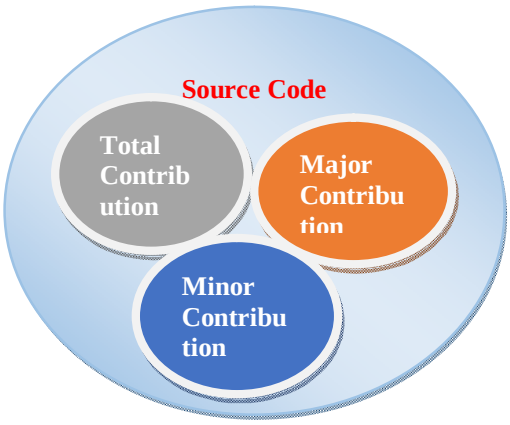


Figure:4.3 Code contribution

Percentage of Ownership over code is directly proportional to the number of commits made by contributor and its impact is calculated on the basis of the number of bugs fixed/number of bugs introduced (over each commit). Experienced contributor will introduce less bugs as compared to the contributor having no experience in the relevant field, thus may degrade the overall quality of software. Now the developed metrics shows the
Tags used:

NOM=Number of modules

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 9, September 2016

Number of non-trivial modules identified by the analyser. Non-trivial modules include all classes, and any other module for which member functions are identified.

LOC = Lines of Code

Number of non-blank, non-comment lines of source code counted by the analyser.

COM = Lines of Comments

Number of lines of comment identified by the analyser

MVG = McCabe's Cyclomatic Complexity

A measure of the decision complexity of the functions which make up the program.The strict definition of this measure is that it is the number of linearly independent routes through a directed acyclic graph which maps the flow of control of a subprogram. The analyser counts this by recording the number of distinct decision outcomes contained within each function, which yields a good approximation to the formally defined version of the measure.
L_C = Lines of code per line of comment

Indicates density of comments with respect to textual size of program
M_C = Cyclomatic Complexity per line of comment

Indicates density of comments with respect to logical complexity of program
IF4 = Information Flow measure

Large Scale Projects
Scheduling and dispatching project

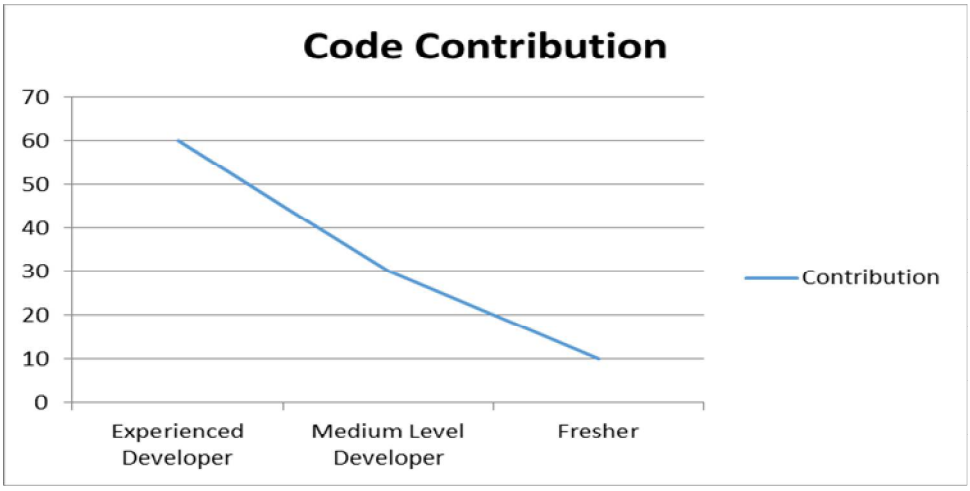


Figure: code contribution

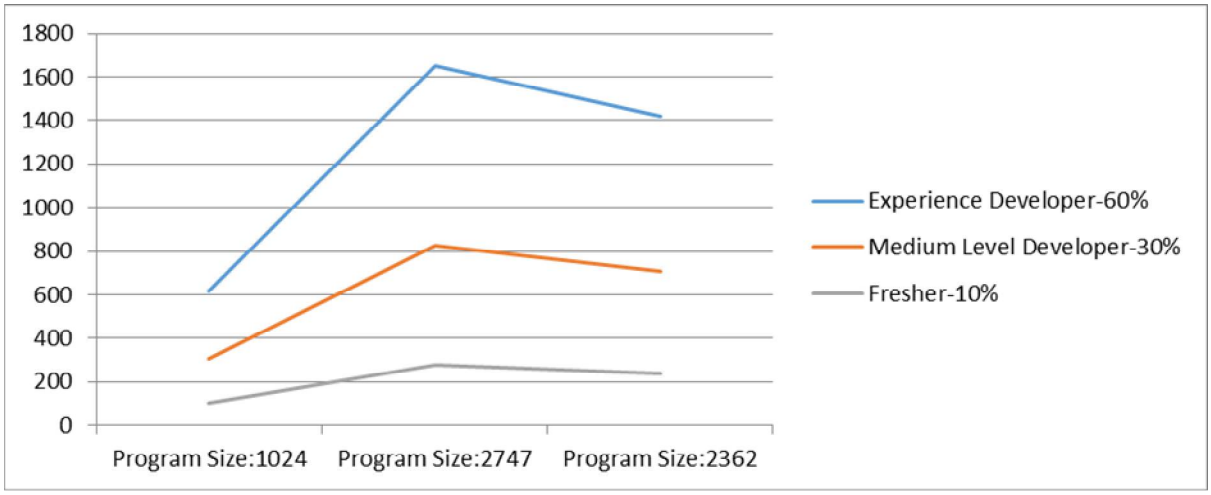
Figure: above shoes the contribution made by each developer. It can be observed that experienced developer has made the highest contribution in development as compared to others. So it indicates that it will also have the impact over the software quality.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 9, September 2016



IV. CONCLUSION

In this paper, we explored the various software metrics developed by other researchers and defined a metrics which can recognize the relationship between develops skill, experience in relevant field and its impact over software maintainability and quality.

We have explored the three different software projects having different program size. If there are 3 developers then we can calculate: If we assume that 60% commits are made by experience developer over LOC=1024 then Total ownership= (614.4/1024)*100=60% for experience level developer

Quality = (1- (10)/614.4)*100=98.37%. If we assume that 30% commits are made by medium level developer then Total ownership= (307.2/1024)*100=30%, Quality = (1- (30)/ 307.2)*100=90.23% and

If we assume that 10% commits are made by medium level developer then Total ownership= (102.4/1024)*100=10% and Quality = (1- (60)/ 102.4)*100=41.40%.

It can be observed that experienced developer has made the highest contribution in development as compared to others. So it indicates that it will also have the impact over the software quality. Experience developer will introduce less bugs as compared to other developers, medium level developer can utilize his skills to reduce the bugs but fresher depends upon the theoretical knowledge and starts learning from his experience.

If program size increases then the probability of bugs increases and fresher can introduce more bugs as compared to others.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 9, September 2016

It can also be observed that it is easy to maintain the small program size as compared to the rage one. As the program size increase, only experience developer can ensure the quality and maintainability of the software up to 60%. Medium experienced developer can ensure the maintainability up to 30% and for fresher's, it is only 10%, which is lowest as compared to others.

Finally, it can be concluded that experience and skills of the developer have an impact over the code ownership, code quality and its maintainability. All these factors are affected by the program size also.

For small program size, all developers can perform well, but if program size increases, code quality decrease w.r.t. experience and the contribution made in code ownership.

REFERENCES

1. Joan C. Miller, Clifford J. Maloney. "Systematic mistake analysis of digital computer programs". Communications of the ACM (New York, NY, USA: ACM), 1963
2. Glenford J. Myers (2004). The Art of Software Testing, 2nd edition. Wiley. ISBN 0-471-46912-2.
3. Position Paper CAST-10 (June 2002). What is a "Decision" in Application of Modified Condition/Decision Coverage (MC/DC) and Decision Coverage (DC)
4. MathWorks. Types of Model Coverage
5. M. R. Woodward, M. A. Hennell, "On the relationship between two control-flow coverage criteria: all JJ-paths and MCDC", Information and Software Technology 48 (2006) pp. 433-440
6. Dorf, Richard C.: Computers, Software Engineering, and Digital Devices, Chapter 12, pg. 15. CRC Press, 2006. ISBN 0-8493-7340-9, ISBN 978-0-8493-7340-4
7. RTCA/DO-178B, Software Considerations in Airborne Systems and Equipment Certification, Radio Technical Commission for Aeronautics, December 1, 1992
8. RTCA/DO-178C, Software Considerations in Airborne Systems and Equipment Certification, Radio Technical Commission for Aeronautics, January, 2012
9. http://en.wikipedia.org/wiki/Run_time_%28program_lifecycle_phase%29
10. exec". The Open Group Base Specifications Issue 6, IEEE Std 1003.1, 2004 Edition. The Open Group. Retrieved 2008-06-23
11. http://whatis.techtarget.com/definition/loader
12. How Many Lines of Code in Windows XP?". Microsoft. January 11, 2011
[13] Debbarma, M.K., "Static and dynamic software metrics complexity analysis in regression testing", Computer Communication and Informatics (ICCCI), 2012 International Conference, IEEE, 2012
[14] Yangsong Wu ; State Key Lab. for Novel Software Technol., Nanjing Univ., Nanjing, China ; Yibiao Yang ; Yangyang Zhao ; Hongmin Lu, "The Influence of Developer Quality on Software Fault-Proneness Prediction", SERE, IEEE-2015, pp.11-19
[15] Lee, Shou-Yu ; Li, Yihao, "DRS: A Developer Risk Metric for Better Predicting Software Fault-Proneness", TSA, IEEE-2015, pp.120-127
[16] ANJANA GOSAIN, GANGA SHARMA, "DYNAMIC SOFTWARE METRICS FOR OBJECT ORIENTED SOFTWARE: A REVIEW", INFORMATION SYSTEMS DESIGN AND INTELLIGENT APPLICATIONS, ADVANCES IN INTELLIGENT SYSTEMS AND COMPUTING, SPRINGER-2015, PP.579-589
[17] CHANDAN KUMAR, DILIP KUMAR YADAV, "SOFTWARE DEFECTS ESTIMATION USING METRICS OF EARLY PHASES OF SOFTWARE DEVELOPMENT LIFE CYCLE", SPRINGER-2014, PP. 1-9
[18] TAO YUE, SHAUKAT ALI, "A MOF-BASED FRAMEWORK FOR DEFINING METRICS TO MEASURE THE QUALITY OF MODELS", ECMFA, SPRINGER-2014, PP. 213-229
[19] GREILER, M. ; MICROSOFT CORP., REDMOND, WA, USA ; HERZIG, K. ; CZERWONKA, J.," CODE OWNERSHIP AND SOFTWARE QUALITY: A REPLICATION STUDY", MINING SOFTWARE REPOSITORIES (MSR), 2015 IEEE/ACM, PP. 2 - 12
[20] YILIN QIU, WEIQIANG ZHANG, WEIQIN ZOU, JIA LIU*, QIN LIU, "AN EMPIRICAL STUDY OF DEVELOPER QUALITY", INTERNATIONAL CONFERENCE ON SOFTWARE QUALITY, RELIABILITY AND SECURITY, IEEE-2015, PP.202-209
[21] SANDEEP KAUR, KANWALJEET KAUR, NAVJOT KAUR, "INTERNATIONAL CONFERENCE ON ADVANCES IN COMPUTING AND COMMUNICATION ENGINEERING", IEEE-2015, PP.639-643
[22] GULNARA ZHABELOVA, VALERIY VYATKIN, "TOWARDS SOFTWARE METRICS FOR EVALUATING QUALITY OF IEC 61499 AUTOMATION SOFTWARE", IEEE-2015, PP.1-8
[23] SUNGDO GU, SO YEON KIM, HYUN-HWAN JEONG, KYUNG-AH SOHN, "CONSTRUCTING AND EXPLOITING SOFTWARE METRICS NETWORKS FOR SOFTWARE QUALITY ASSESSMENT", IEEE-2015, PP.1-5
[24] Bettenburg, N. ,Sch. of Comput., Queen's Univ., Kingston, ON, Canada , "Studying the Impact of Developer Communication on the Quality and Evolution of a Software System: A Doctoral Dissertation Retrospective", ICSME, IEEE-2014, pp. 651 – 656
[25] Lavallee, M. ; Dept. de Genie Inf. et Genie Logiciel, Polytech. Montreal, Montréal, QC, Canada ; Robillard, P.N., "Why Good Developers Write Bad Code: An Observational Case Study of the Impacts of Organizational Factors on Software Quality", ICSE, IEEE-2015, Vol. 1, pp.677-687
[26] Lee, Shou-Yu ; Li, Yihao, "DRS: A Developer Risk Metric for Better Predicting Software Fault-Proneness", TSA, IEEE-2015, pp.120-127
[27] SOLIMAN, AHMED, S.H, "UTILIZING CK METRICS SUITE TO UML MODELS: A CASE STUDY OF MICROARRAY MIDAS SOFTWAR", INFOS, IEEE-2010, PP.1-6
[28] M. Greiler, Kim Herzig, Jacek Czerwonka, "Code Ownership and Software Quality: A Replication Study", IEEE/ACM 12th Working Conference on Mining Software Repositories, 2015 , pp.2-12
[29] J. H. Hayes, Wenbin Li, Tingting Yu, Xue Han, Mark Hays, Clinton Woodson, "Measuring Requirement Quality to Predict Testability", AIRE, IEEE, 2015, pp.1-8
[30] F. A. Fontana, Vincenzo Ferme, Marco Zanoni, "Towards assessing software architecture quality by exploiting code smell relations", IEEE/ACM 2nd International Workshop on Software Architecture and Metrics, pp.1-7
[31] S. Eldh, Brendan Murphy, "Code Ownership Perspectives", IEEE Journals & Magazines, 2015, Vol.32 (6), pp.18-19

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 9, September 2016

[32] C. Farag, Péter Heged˝us, Gergely Ladányi, and Rudolf Ferenc, "Impact of Version History Metrics on Maintainability", 8th International Conference on Advanced Software Engineering & Its Applications, IEEE, pp.30-35
[33] Z. Bukhari, Jamaiah Yahaya, Aziz Deraman, "Software Metric Selection Methods: A Review", 5th International Conference on Electrical Engineering and Informatics, IEEE-2015, pp. 433 – 438
[34] MOHSIN, S. ; DEPT. OF COMPUT. SCI., HANYANG UNIV., ANSAN, SOUTH KOREA ; KALEEM, " PROGRAM SLICING BASED SOFTWARE METRICS TOWARDS CODE RESTRUCTURING", COMPUTER RESEARCH AND DEVELOPMENT, 2010 SECOND INTERNATIONAL CONFERENCE, IEEE-2010
[35] ABDI, A. ; INF. & COMMUN. SECURITY DEPT., IRAN TELECOMMUN. RES. CENTER (ITRC), TEHRAN, IRAN ; SOUZANI, A. ; AMIRFAKHRI, M. ; MOGHADAM, "USING SECURITY METRICS IN SOFTWARE QUALITY ASSURANCE PROCESS", IST, IEEE-2012
[36] AKINGBEHIN, K. ; DEPT. OF COMPUT. & INF. SCI., UNIV. OF MICHIGAN-DEARBORN, DEARBORN, MI, USA, "A STRUCTURED FRAMEWORK FOR SOFTWARE METRICS BASED ON THREE PRIMARY METRICS", ICIS, IEEE-2010
[37] FRANCA, J.M.S. ; FAC. OF COMPUT., FED. UNIV. OF UBERLANDIA, UBERLANDIA, BRAZIL ; DOS SANTOS, C.A.R. ; DE OLIVEIRA, K.S. ; SOARES, M.S., "AN EMPIRICAL EVALUATION OF REFACTORING CROSSCUTTING CONCERNS INTO ASPECTS USING SOFTWARE METRICS ", (ITNG), IEEE-2013