ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 2, No 5, May 2013

Overview of Impact of Requirement Metrics in Software Development Environment

[1]Mohd.Haleem, [2]Prof (Dr) Mohd.Rizwan Beg, [3]Sheikh Fahad Ahmad

Abstract: **Requirement engineering is the important area of software development life cycle. Requirements engineering play an important role in maintaining software quality. Software quality depends on many factors like delivery on time, within budget and fulfilling user's needs. Software requirements are the foundations through which quality can be measured. Quality should be maintained from starting phase of software development. Software quality during requirement engineering process can be maintained through requirement metrics. The objective of this paper is to compare quality metrics involve in requirement engineering process and analysing various requirement metrics, and software quality metrics that are involve during requirement process of a software development life cycle and analysing impact of requirement metrics in software development environment.**

Keywords **Software quality, Requirements metric Quality Metrics, Requirements management.**

I. INTRODUCTION

Software metric is a field of software engineering that is associated with diverse measurements of computer software and its developments. Software metrics [1] [2] [3] is one of the important tools for analyzing the software product in an effective way. In other words software metrics are measures that enable software developers and software analyst to gain insight into the efficiency of the software process and projects that are conducted using the process as framework. Software metrics measures different aspects of software complexity and therefore play an important role in analyzing and improving software quality [3]. With the help of software metric we are able to understand the software product in an effective way. Software metric is a field of software engineering that is associated with diverse measurements of computer software and its development. According to Tom DeMarco that "You cannot control what you cannot measure".Software metric [4] [5] are helpful in improving the quality of software, planning the budget, its cost estimation etc. with the help of software metric we are able to understand the software product in an effective way.

II. DEFINING: METRICS

Famous quote of a management consultant Peter Drucker: "If you can't measure it, you can't manage it", if a project manager and team members are not able to precisely measure what they are going to do then it would not be possible for them to effectively manage and improve the performance of a project. The success of a software project is the primary goal. Metrics that help to measure the success or failure of a project are very diverse and these metric hardly have a good deal in commonalities [6]. Metrics are also helpful to determine current status of a project and evaluate its performance. Software metrics can be classified into three categories: product metrics, process metrics, and project metrics. Product metrics describe the characteristics of the product such as size, complexity, design features, performance, and quality level. Process metrics can be used to improve software development and maintenance. Project metrics describe the project characteristics and execution.

III. QUALITY METRICS

Software quality metrics are the subset of software metrics that focuses on the quality aspect of software. Software quality metrics are associated with process and product metrics than with project metrics. Software quality metrics can be divided further into end-product quality metrics and in-process quality metrics. The need of software quality engineering is to determine the relationships among in-process metrics, project characteristics, and end-product quality in order to improve quality in both process and product. Moreover, we should view quality from the entire software life-cycle perspective and, we should include metrics that measure the quality level of the maintenance process as another category of software quality metrics.

IV. SOFTWARE REQUIREMENT METRICS

Requirement engineering deals with elicitation, analysis, communication and validation of the requirements. As soon as errors are identified in initial stage, it is easy to fix them as compared to later identification of errors. And the cost of fixing these errors in initial stages is much lower than fixing them in later stages of software development. Metrics that can be collected during requirements are the size of the requirements, requirements traceability, requirements completeness and requirements volatility [7].

1) Size Metrics: Size is an important metrics that are used to measure requirements. LOC is common metrics used to measure size of the software. Comment lines and blank lines are not considered as code. A Non commented line is considered as a code but it also include executable commands, executable statements and data declarations. Use Cases can also be considered as a size measurement when they are used to describe requirements. For example number of use cases, number of functions covered etc.
Use case metrics are categorized into 3 categories [8].

- Size metrics: Size metrics includes number of atomic actions in main flow. Size of use case can be determined by counting number of actors weighted by their complexity.
- Environment metrics: Environment metrics are the factors which has influence on complexity level of the use cases. These are independent of the size of the use cases.

 Composite metrics: Composite metrics are derived from size metrics and environment metrics.

2) **Traceability Metrics:** Traceability is the ability to trace requirements in a specification to their origin from higher level to lower level requirements in a set of documented links. Traceability provides information which helps in determining whether all relationship and dependencies are addressed. This also makes requirements leakage - existence of lower level requirements with no valid origin from higher level visible. It helps in preventing wrong interpretations of other metrics. There exist 5 types of traceability metrics [9].

 Next level coverage metrics: This metric traces number of requirements to next level up and next level down. This metrics also trace number of requirements in both directions.

 Full depth and height coverage: This is similar to next level coverage metrics and best suited for 3 or less level specifications. It traces requirements to highest and lowest level of specifications instead of only immediate next level in both directions.

 Inconsistent traceability metrics: This includes the numbers of requirements in a specification that have at least one inconsistent link in both upward and downward directions.

 Undefined traceability metrics: It include the numbers of requirements in a specification that have no traceability links upward and download. However, the highest link will not have any upward link as it is derived from design document and the lowest level link has no down link.

 Linkage statistic metric: It measure the complexity level of traceability data by counting the number of higher or lower level requirements to which each requirements specification is traced.

3) Completeness Metrics: This metrics are used to assess whether a requirement is at wrong level of hierarchy or too complex. Requirements are considered as complete if all pages are numbered, all figures and tables have captions and numbers, all references are present, all sections and subsections are numbered. Requirements Completeness Metrics are of 3 types [8].

 Requirements Decomposition Metrics: This metric is used to know whether the requirements specified have been decomposed sufficiently to be used in next phase. A complex function will have many levels of specification and a simple function will have few levels.

 Requirements Specification Development Metrics: This metric is used to provide information about the work completed and the work remaining for a given specification.

 Requirements Specification Metrics: This metric provide information about the quantity of items for which issue resolution is needed. This metric used 'to be determined', 'to be supplied' tags used for completing the requirements document.

4) Volatility metrics: The degree to which requirement changes over a time period is called volatility of requirements. This metric is used to assess reasons for change of requirements over a period of time. Requirements degree of change is also checked by this metric. Both these factors are checked to know whether the changes are consistent with current development activities. It indicates changes such as addition, deletion and modifications. It helps in tracing future requirements, design and code volatility. Volatility can be high in the initial phase of software development. It should be reduced as the project progress so that further development should not be affected. This metric is a good indication of other metrics for example, if a specific portion of software is tested without knowing the volatility of the requirements than this testing would not be wrathful.

V. REQUIREMENT QUALITY METRICS

Use cases and other traditional tools are used to collect and frame the system architecture. However, to check the quality of collected requirements, many requirement quality metrics are discussed in the literature which remains as open issues for further improvement [10].

1) Unambiguous: Requirement collection is a crucial phase of SDLC since ambiguous and wrong requirements may cause system failure. Since analysing a scenario in to classes and objects is up to one's perception, ambiguity may get induced among reviewers leading to an undesired system.

$$QUA=\sum \frac{R_{idr}}{N}$$

Where R_{idr} is number of requirements reviewed identical by reviewers,
N is total number of requirements.

2) Correctness: A Use case Scenario explores numerous functional requirements. However, the right interpretation of user vocabulary will lead to correct set of valid requirements. Vague requirements like 'shall', 'multi-user', 'user-friendly', are thoroughly analysed as an input to design phase.

$$QC= Nv/(Nnv*N)$$

where
Nv is number of valid requirements,
Nnv is number of still not valid requirements and
N is Total number of requirements

3) Completeness: It reflects the depth/ breadth of requirements in a scenario. Each requirement is unique and need to serve the user as a complete package. Though many requirement metrics are discussed and practiced, it is hard to quantify the quality of requirements metric. Organisational guidelines and best practices will improvise the requirement collection to analysis process including their traceability and volatility. However, involving the stakeholders in requirement analysis would reduce the confusion and frequent changes due to uncertainty. Having a count on requirement changes other than business change will give an insight on process adapted.

Quality Metrics	Formula	Purpose	Conclusion
Unambiguous	$Q_{ua} = \frac{R_{id}}{R_t}$ Where R_{id} is the no of identical requirements R_t is the total no. of requirements.	To obtain the identical requirements that has been interpreted in a uniquely by all reviewers.	0 or close to 0 means requirements are Ambiguous. 1 or Close to 1 means requirements are Unambiguous
Completeness	$Q_{cm} = \frac{R_{un}}{R_t}$ Where R_{un} is the no of unique requirements, R_t is the total no. of requirements.	To measure the total number of functions currently specified.	As closer to 1, then more complete it becomes
Correct	$Q_{cr} = \frac{R_{ip}}{R_t}$ Where R_{ip} is the no of requirements having same interpretation R_t is the total no. of requirements.	To calculate the requirements that have been correctly validated	0 means incorrect 1 means correct
Consistent	$Q_{cn} = \frac{R_{un} - R_n}{R_n}$ where, Run is the number of unique functions specified, Rn is the number of unique functions that are non deterministic	To measure the percentage of unique functions that are deterministic assuming that SRS can be defined as a function that maps inputs and states in to outputs.	0 means incosistent 1 means consistent
Uniqueness	$Q_{un} = \frac{R_{di}}{R_t}$ where, Rdi is the requirement with distinct explanation,R_t is the total no. of requirements.	To obtain the no of unique requirement that have been explained by all reviews	As closer to 1, then more unique it becomes
Understandable	$Q_{un} = \frac{R_{us}}{R_t}$ where, Rus is the requirement understood by the users, R_t is the total no. of requirements.	To measure the number of requirements those are understood by all reviewers.	0 means No requirement understood. 1 means All requirements understood.
Modifiable	$Q_{un} = \frac{R_{mo}}{R_t}$ where, Rmo is the no of requirements modified R_t is the total no. of requirements.	To count the no. of requirements that are required to changed or modified	Not Defined
Traced	$Q_{tr} = \frac{R_{tr}}{R_t}$ where, Rtr is the no of requirements traced R_t is the total no. of requirements.	To measure level of tracing the requirement and any changes is difficult because this activity spans throughout the SDLC	Not Defined
Traced	$Q_{tr} = \frac{R_{tr}}{R_t}$ where, Rtr is the no of requirements traced,R_t is the total no. of requirements.	To measure level of tracing the requirement and any changes is difficult because this activity spans throughout the SDLC	Not Defined
Verifiable	$Q_{vr} = \frac{R_t}{R_t + C + T}$ R_t is the total number of requirements C is the cost necessary to verify presence of requirements T is the time necessary to verify presence of requirements	To measure the verifiability of a requirement.	0 means Very poor. 1 means Very good.

VI. COMPARISION OF QUALITY METRICS

In this section we have tried to compare a set of quality metrics in requirement engineering that are used to ensure the quality of the requirement and hence improving overall quality of the product and also analyse their meaning in terms of requirement.

Table 1: Quality Metrics In Requirement Engineering

VII . IMPACT OF REQUIREMENT METRICS

Requirements are the foundation of the software development. The success of software product, both functionally and financially is directly related to the quality of the requirements. Although the initial sets of requirements are well documented, requirements changes will occur during the software development lifecycle. Thus constant changes (addition, deletion, modification) in requirements during the development
lifecycle impact the cost, the schedule and the quality of final product.
One way to understand the importance of requirements is to look at the costs of correcting errors. As an example, the typical relative costs of correcting an error at a particular stage of software development is given in figure 1.1 [11]. As an example, an error detected in the requirements phase is 0.2

Phase	**Relative Cost**
Requirements	0.1-0.2
Design	0.5
Coding	1
Unit Testing	2
Acceptance Testing	5
Maintenance	20

Figure 1.1: The relative costs of correcting errors made at the requirements stage of a project.
times as costly to fix as an error detected in the coding stage. An error detected in the maintenance phase is 20 times as costly to fix as an error detected in the coding stage. Figure 1.1 gives the cost of detecting and repairing an error in the coding phase relative to the cost of detecting and repairing errors in other phases.

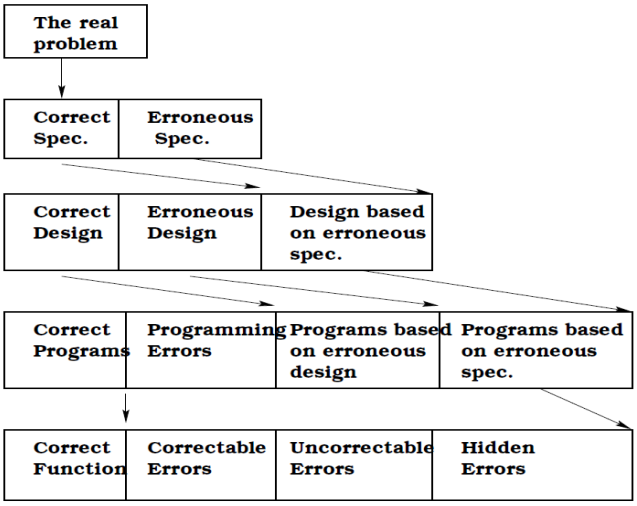

Figure 1.2: The cumulative effects of error!

Figure 1.2 shows the cumulative effects of errors made at the various stages. Other studies, such as the one reported in Sallis et.al. [12] estimate that the percentage of errors in released software due to specifically to requirements to be in excess of 50% of all errors detected. Similar evidence suggests that concentrating effort earlier in software development processes can have a dramatic effect

VII. CONCLUSION

Quality is an uncompromised factor in software development process. Software deliverables are checked for quality at every phase of development process to ensure the quality of end product. Quality process is organization-specific providing a basement for quality product. Numerous metrics are used by various industries to measure the quality of requirement phase to implementation phase to uphold them in the market. Requirements metrics discussed checks the degree of change of requirements, volatility. Traceability, process of linking high level to low level requirements. Incompleteness in requirements documents is also checked. Quality factors such as completeness, correctness, understanding etc. are discussed and their corresponding formulae are mentioned. These requirements metrics help in identifying and rectifying errors in requirements document. However, research in this area is in continual progress to provide better metrics for Product, People, Process to support the development of software. Implementation of quality metrics during the development process ensures production of high quality software.

REFERENCES

[1] H F Li, W K Cheung "An Empirical Study of Software Metrics" Software Engineering IEEE Transactions on (1987) Volume: SE- 13, Issue: 6, Pages: 697-708 [2] N E Fenton "Software Metrics" Conference Proceedings of on the future of Software engineering ICSE 00(2000) Volume: 8, Issue: 2, Publisher: ACM Press [3] Manik Sharma, Gurdev Singh, "Analysis of Static and Dynamic Metrics for Productivity and Time Complexity", IJCA, *Volume 30– No.1, September 2011*
[4] Manik Sharma, gurdev singh, "A Comparative Study of Static Object Oriented Metrics", IJoAT
[5] S.R Chidamber and C.F. Kemerer, "A Metrics Suite for Object Oriented Design", IEEE Transactions on Software Engineering, Vol. 20 No. 6
[6] C. Sirias, "Project Metrics for Software Development", **(2009)** Available online at: http://www.infoq.com/articles/project-metrics.
[7] Mohammad A. Rob, "Issues of Structured Vs. Object-Oriented Methodology of Systems Analysis and Design", Issues in Information Systems, Volume V, No.1, 2004, pp 275-280.
[8]. Zohra Bellahsene, Dilip Patel, and Colette Rolland. Object-oriented information systems. In OOIS, pages 409–421, 2002.

[9]. Rita J. Costello and Dar-Biau Liu. Metrics for requirements engineering. Journal of Systems and Software, 1995.
[10] Mohammad Ubaidullah Bokhari and Shams Tabrez Siddiqui, "Metrics for Requirements Engineering and Automated Requirements Tools", Proceedings of 5th National Conference-Computing for Nation Development, New Delhi, March 2011.
[11] Davis, A.M. Software Requirements: Objects, Functions and States. Prentice Hall, 1993. Revised Edition.
[12] Philip Sallis, Graham Tate, and Stephen McDonell. Software Engineering. Addison Wesley Publishing Co., 1995.

Author Profile

Mohd. Haleem
received B.Sc(Hons) degree from Aligarh Muslim University in 2005 and MCA degree from HBTI, Kanpur in 2009 and currently pursuing M.Tech degree in Computer Science & Engg from Integral University, Lucknow. From 2011 onwards he is working as a Lecturer in the department of Computer Application, Integral University, Lucknow,India.

Prof. (Dr.) Mohd Rizwan Beg
received B.Tech degree in Computer Science & Engg in 1995 and completed his M.Tech and PhD degree in Software Engineering. Currently he is working as the Dean of Computer Application Deptt and Head of Computer Science & Engg Deptt, Integral University, Lucknow. He is also the member of Editorial Boards, Program and Technical Committees of many International Journals as well as he is in the advisory committees of many International Journals. Currently he is supervising eight candidates in PhD Program and he has around 78 International Publications.

Sheikh Fahad Ahmad
received his B.Tech degree in Computer Science & Engg. from Integral University, Lucknow in 2008 and currently pursuing M.Tech degree in Computer Science & Engg from Integral University, Lucknow. From 2010 onwards he is working as a Assistant Professor in the department of Computer Science & Engg, Integral University, Lucknow, India.