Volume 3, Issue 7, July 2013 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Scope of Exploring CQE Dimensions in Reengineering of Legacy Program

Harmeet Kaur
Ph.D. (Computer Applications)
Research Scholar,
Punjab Technical University,
Jalandhar, Punjab, India

Shahanawaj Ahamad
Astt. Professor,
Dept. of C. S. & Soft. Engg
College of Comp. Sc. &Engg.
University of Ha,il,KSA, India

Gurvinder N. Verma
Professor & HoD
Applied Sciences
Shri Sukhmani Institute of Engg. & Tech.
Derabassi, Punjab, India

Abstract: Reengineering of legacy program is the most important issue faced by software industry. There are many challenges in the way of reengineering, so to overcome these problems some important dimensions are explored. There are many factors which identify the reengineering dimensions like environmental, size, economical, technical etc.In the present work an attempt has been made by review of literature to identify the dimensions and their parameters in the software Industry. In the present study these dimensions are analyzed by conceptual and empirical studies.

Keywords – LOC, CQE, Economical, Environmental, NFRs.

I INTRODUCTION

Software is one of the fastest growing sectors of the global economy and measuring CQE has always been the overriding consideration in the conduct of all reengineering activities. Due to the nature of the software industry, the total elimination of bugs or serious errors is unachievable. The purpose of the present study is to provide a theoretical framework and use this to explore CQE dimensions of reengineering legacy program. There are many factors which are responsible for exploring CQE parameters in software industry/ reengineering legacy program. One of them is economical factor which is of utter importance for the software industries dealing with reengineering of legacy program as the motive of the industries is to reduce the cost and enhance the quality of the software. Another factor which of great concern is human factor which is highly important in the reengineering of the legacy code. No human endeavor or human-made system can be free from risk and error, and failures will be expected to occur in spite of the most accomplished prevention efforts. The system must, however, seek to understand and control such risks and errors. The goal of this research is to understand these errors in reengineering. In order to understand how these parameters can assist in reengineering the legacy program, by reducing CQE and human error. CQE is the important issue in the software industry as far the reengineering of the legacy program is concerned. Maintaining the legacy program consumes significant amounts of time and money and contributes to the growing problems of measuring CQE for reengineering of legacy code.As far as the complexity of the program is concerned it is evident that there are many factors which are responsible for making the program complex like size, number of I/O etc. Actually, complexity of software products has been observed for decades. As such, Number of researchers has proposed variety of complexity metrics [7][8] for different types of software, software languages, software products and related technologies [5][6]. All the reported complexity measures are supposed to cover the correctness, effectiveness and clarity of a system and to provide good estimate of these parameters. With the emergence of the new technologies, new measurement techniques evolve. There is an ongoing effort to find such a comprehensive measure, which addresses most of the parameters for evaluating complexity quality and effort of the system. In addition, the quality objectives may be listed as performance, reliability, availability and maintainability [4][10] that are all closely related with software complexity.

Complexity of software product is much higher than that of other industrial products. It is always hard to control the quality if the code is complex[3][10].There are several quality attributes such as security, performance, reusability, availability, testability, correctness, maintainability, reliability, integrity and many others [4][10]. To achieve some of those quality attributes, complexity should be reduced. For example, to be able to test software easily it is necessary that the software is not complex. Otherwise, the testing process will be harder and thus the cost will be higher. What makes software quality assurance unique is product complexity, visibility, and development process. Visibility is another difficulty of software quality assurance, since other industrial products are visible but software products are not visible until the end is reached. Software development process differs with its development methodologies and difficulties in finding and removing defects [10]. Similarly, Hughes and Cotterell[12] state that intangibility, increasing criticality of software, and accumulating defects during development process make the software quality unique. Furthermore, software needs to be measured in order to understand CQE. Otherwise it may not be possible to make an effective reengineering of the legacy program. In fact, a reengineering of legacy program involves the integrity of a complex chain of CQE parameters and sub parameters. Problems or the failures in the software typically result from a combination of multiple inter-related factors and events, in which human error is often involved? Due to normalized deviance of human performance, errors and incident data alone cannot reveal how the reengineering fails and the human contribution to these failures.

II. PARAMETERS OF CQE IN SOFTWARE

With the development in technology, the recurrence of multicultural connections between and among commercial ventures will expand. The aforementioned distinctions will have worldwide implications for measuring and conveyances impact of CQE in reengineering operations. This scenario can almost always tend to polarize the participants in any discourse and can prompt add up to inaction or to inadequate and inadmissible bargains. Possibly of the aforementioned outcomes is altogether unsuitable to anybody fascinated by reengineering of legacy program. There are numerous CQE parameters and sub parameters in front of Software industry which should be acknowledged for successful reengineering of the legacy system.

A. Economical Factor: From the study it was observed that CQE is influenced by different prudent components. For example direct and indirect cost included in measuring the CQE of the program. It has been perceived that as the size of the software increments the complexity of the software also increases and likewise the cost of designing, supporting and maintaining the software increases. The complexity of the program increases with the expansion in program size and error proneness. It is observed that it is not conceivable to make the project 100 percent error free and the amount of effort needed to correct error and code quality influenced by complexity. Another variable which donates in expanding the cost of software is changes in client requirements with the change in client requirements the time and effort to improve the software increases if the industry chooses to convey the software product on time the cost of experienced staff increases and this may prompt cost invade. The client expectations changes with change in the technology and in this way it is exceptionally troublesome to meet clients level of expectations this is also responsible for cost overwhelms. The amount of classes, structures I/O variables and nesting levels additionally influence the cost of developing the software. The ultimate objective of any organization managing reengineering of program is to procure benefit by conveying the high quality software product to achieve this objective the complexity of the program must be reduced which will diminish the effort, cost and time included in developing the software.

B. Size: Software size estimation is an activity in software engineering that is utilized to gauge the size of a software application or segment in order to be able to implement other software project management activities such as estimating or tracking. Size is an inborn characteristic of a piece of software much the same as weight is a natural characteristic of an intangible material. As specified prior greater size brings about more complex and failure inclined or error prone software and lessens the quality and increments the effort for developing the software. Software size estimation is distinctive from software effort estimation. Size estimate the plausible size of a piece of software while effort estimation predicts the effort required to build it. The relationship between the size of software and the effort needed to produce its productivity. Case in point, if a software engineer has constructed a minor software application, we can state that the task effort was 275 man-hours. Notwithstanding, this does not give any informative data about the size of the software product itself. On the other hand, we can state that the application size is 5,000 LOCs (Lines of Code), or 30 FPs (Function Points) without distinguishing the effort needed to generate it. It is discovered that the number of variables and structures are the factors that enhance the size and impacts the procedural complexity where as object oriented complexity is influenced by attributes, structures, number of I/O variables and classes. The complexity of the class hinges on the amount of methods and their attributes. It has been observed that there is steer connection between the complexity and maintenance cost of the software i e larger the system size more will be number of lapses or errors and more will be cost of maintenance.

C. Quality: Essentially a software system’s utility is resolved by both its practicality and its non-functional characteristics, for example usability, flexibility, performance, interoperability and security. Quality traits might be measured by a software producer to guarantee that its software holds fast to certain principles or client necessities. The likelihood of accomplishing the needed specification or the functional requirement and imperatives relies on the complexity of design. The performance and simplicity of the process is influenced by its design and artifact complexity, lower artifact complexity more excellent will be the simplicity of the software. The fixation on simplicity brings about attaining high quality and reliability easily and at low cost. Although the requirements engineering community has arranged requirements as either functional or non functional, overwhelmingly existing requirements models and requirements detail dialects fail to offer a legitimate treatment of quality attributes.
Treating quality attributes all in all, and not only as functionally alone, has been a nexus center of works in the area of goal-oriented requirements engineering [1] [2], and specifically the NFR Framework [3] that treats non-functional requirements at an elevated level of abstraction for both the issue and the result .As far as the quality of the software is concerned it is influenced by its such as functionality, usability, reliability, portability, supportability. Usability of the software is identified with number of errors and the length of the user manual that is lesser the amount of error more will be its usability though portability is size of the project measured in LOC (Line of code) and number of parameters. Reliability is influenced by LOC, complexity and number of portability of error messages and usability are interrelated more is the portability more will be its usability which will improve the reliability and increments the functionality of the software and leads to the development of the quality software product. For any software system correctness is a standout

amongst the most important attribute which is needed by the user, for proper working of the software. Another element of concern is maintainability; which could be corrective and adaptive. Corrective maintainability manages errors while, adaptive maintainability manages changes in the requirement of user and adaptive changes manages necessity of the software. Maintainability is corresponds to the amount of re-work. Throughout the product operation phases correctness, reliability, usability, integrity and efficiency play essential part in the quality of the software product where as in product transition stage quality is influenced by interoperability, portability and usability. In product revision phase Maintainability, flexibility and testability are paramount components of the software product. In this phase the software product is tested for errors and it is closely observed whether it is maintainable and flexible.

D. Human consider: As it is well apparent that People and techniques are immensely significant to gain a high quality in the estimation outcomes. As a substitute for considering as elective means they ought to be acknowledged as synergic assets. Software development process is a human-centered activity. Taking into account software development being a human- human-centered activity, human components have an incredible effect on the process and its performance. This might be clarified by the effect of human role in development process i.e. influencing the software development market by being a client, impacting the development process phases by being a developer and at last having a recognizable effect on process performance and success by being a manager. Human consideration in this methodology could be concentrated on from diverse points of view for example mental, cognitive, administration, management and technical aspects. In addition, human elements have distinctive levels in the process simultaneously changing from organizational and interpersonal to individual.

Even though human components have been demonstrated to have affect on software development process, shockingly they have been disregarded by the researchers in the software engineering and development research areas. Consequently, there appears to be a need to distinguish and portray human factors and their effect on development process. Human element plays paramount part in the estimation of CQE of legacy program for effective reengineering. Experience of the team included in software development serves to diminish the complexity by decreasing the repetition in the code which will decrease the cost and effort and will improve the quality of the software product. Another element which assists in lessening the complexity and increasing the quality of the software is legitimate correspondence around the team members and good leadership. Absence of correspondence around the team members will hamper the productivity of the team and the organization likewise which will by one means or another influence the quality of software.

III. Preventive Measures

There is no universal solution for improving CQE of legacy program for reengineering , but from literature survey it was observed that there are many methods that has been shown to work in various software industries for reengineering of the legacy program. The uniqueness of the software industry is that a simple replica approach cannot be followed. It is the organization's responsibility to customize its methods based on multiple factors and implement it taking into consideration the ground realities.

Software industry that has experienced a phenomenal growth which is increasingly facing the problems of success: software size, quality and human concerns are growing. The software size impacts of software are growing, which include program size, number of classes, no of input and output variables, number of structures, number of attributes, nesting levels etc. The quality and NFR impacts include the reliability, usability, maintenance, precision, flexibility, performance etc. whereas human impacts include user requirements, user satisfaction ,manpower, team leadership qualities, external and internal customers, team skills etc. Size of the software increases the complexity of the software also increases which results in decrease in the quality and increase in the effort of reengineering the legacy program. The human factor plays a important role in reengineering of the legacy program which requires the proper training, understanding of the code to be reengineered as the requirement of the user changes the cost and time of reengineering the legacy code also increases. It was observed that it is not possible to measure the customer satisfaction level as customers expectations changes according to the change or with the advancement of the technology. This study analyzed the importance of size, quality attributes, non functional requirements, cost and humans in measuring the CQE of legacy code/software development. Human skills play an important role in measuring the complexity of the code which in turn will affect the overall quality and effort of reengineering legacy program In the software industry there are many parameters like human, technical, economic infrastructure etc which are of utter importance. To sustain the balance between investments with financial and nonfinancial returns is a challenge.

IV Literature Review

Most of the exploration has been undertaken to discover the influence of CQE on reengineering of legacy program. From the last few decades much consideration is given on the reengineering of legacy program. The CQE estimation proceeds as the years progressed and is a crux part of the work of software development and research throughout the world. Complexity is defined as " the degree to which a system or component has a design or implementation that is difficult to understand and verify ". [Basili, 1990] defines complexity as a measure of resources used by a software system during the interaction of the parts of the software, to perform a task. Provided that the interacting element is a PC, then complexity is related with the execution time and hardware assets needed to perform the task. Provided that the collaborating element is a programmer, then complexity is identified with the difficulty of coding, testing and altering the software.[Grady, 1992] furnished an answer to the inquiry "why measure software?". He presumed that software estimation is utilized to give a premise for evaluations, track progress, verify complexity, understand quality, examine the causes of deformities, and validate best practices. In another study [Parnas, 1975] recommended that modules ought to be structured so a module has no knowledge of the inside structure of different modules. The aforementioned variables influence cost and quality of software, as discussed by [Porter & Selby, 1990]. [Wolverton, 1974] made one of the earliest endeavors to formally measure programmer productivity utilizing LOCs. He proposed object guidelines per man-month as a guideline measure and inferred what he acknowledged to be typical code rates.

Since it was acknowledged that software development is a complex task. Accordingly, it is hard to accomplish a high level of quality. Because of its complexity, software quality has been a rising interest for a considerable period of time and certain definitions have been manifested all through software history. A software product may as well convey numerous quality attributes , for example correctness, reliability, efficiency, integrity, usability, maintainability, testability, flexibility, portability, reusability, and interoperability. Consistent with [Sommerville, 2007] the most fundamental software quality attribute is maintainability. To efficiently maintain a software system, the codes ought to be understandable. Briefly, to realize high quality, reduction of complexity is vital.

Software reliability is apparently the most basic attribute of a software system. Collecting that the software can't perform its required purpose dependly, and then its development costs (both time and effort) are squandered. In this manner, software must be improved with a high level of reliability at a sensible cost. [Sheppard, 1993] recommended that program size is an incredible indicator of indirect and more qualitative program characteristics for instance reliability and maintainability. High software reliability is an imperative characteristic of high affirmation systems.

Software quality models yield opportune forecasts of quality indicators on a module by module support, empowering one to keep tabs on finding faults early in the software development. In addition to software reliability, software renewal process has likewise vital part to play in measuring software quality the study conveyed by [Visaggio] on the application of renewal process to an extremely old legacy system he discovered that from the maintainers outlook the renewal technique renders the system easier to comprehend, so that the effort needed to enhance its quality can then be tailored to the benefits aimed at, even on an extremely old system. This study has made it plausible to draw a distinction between the restoration process and those of reverse engineering and reengineering, and it has additionally pointed out the requirement to carry out reverse engineering even on the systems which unquestionably require reengineering. The study had other spin off profits, giving guide lines for investigations to be made throughout reverse engineering process to the size of the system to be replenished and to extract the knowledge stored so as to understand and enhance the existing application. Although Non-Functional Requirements (NFRs) have been available in numerous software development strategies, they have been introduced as a second or even third class type of requirements, as often as possible covered up inside notes and thusly, frequently neglected or overlooked. Shockingly, in spite of the fact that non-functional requirements are the most expensive and difficult to manage there are still few works that keep tabs on NFRs as first class requirements.

[Bajpai & Gorthi ,2012] given the details on Non-Functional Requirements and its significance in different fields. They have broadly surveyed the different non-functional requirements and their imperativeness in the present competitive software market and have examined the impact of working out for the non- functional requirements which leads to the disclosure of new functional requirements. Though in another study [Hill et. al,2001] have proposed a schema for quantifying non-practical necessities (NFRs) the framework utilizes quality attributes of the execution domain, application domain and segment architectures to refine qualitative necessities into quantifiable ones.

They observed that throughout the refinement process clashes are determined and more concrete non-functional requirements are processed. They inferred that the upgraded refinement process could be utilized to quantify any nonfunctional requirements that could be communicated numerically. In an another study [Song et.al,2009]described their practical experiences in improving NFRs for large software platform, the challenging issues they confronted, and the procedures they used to address those issues. They recommended that practical procedures assisted with NFR compromise and administration, and enhanced the quality of the NFR definitions. The quality of the NFR specification has allowed mechanization of the platform performance testing for the past two years.

[Jørgensen &-Østvold,2004]have conveyed a study which aims to enhance investigations of why errors occur in software effort estimation, they gathered informative data about estimation errors through: 1) interviews with employees in diverse roles who are responsible for estimation, 2) estimation experience reports from 68 finished projects , and 3)statistical examination of relations between characteristics of the 68 finished projects and estimation error from one software development company. They discovered that the role of the respondents, the data collection methodology, and the type of investigation had a vital effect on the reasons given for estimation errors.

Effort estimation in software development is both basically critical and intensely flawed. Industry reviews propose that a large number, if not most software projects that are conveyed are either over time or over budget. Although numerous factors are involved in such outcomes, incorrect effort estimates underlie the majority of the aforementioned factors. The examination done by [Christopher Jarabek,2006] has investigated numerous aspects of expert estimation. First and foremost the idea of what constitutes expertise was examined. It was confirmed that ability has numerous definitions depending on the context in which it is utilized.

Precise estimation of the software effort and schedule influences the budget computation. Bidding for contracts depends mostly on the estimated cost. Wrong evaluations will lead to failure of making a benefit, expanded likelihood of project incompletion and defer of the project delivery date. [Sheta et.al,2008] have investigated the utilization of soft computing procedures to build a suitable model structure to utilize enhanced estimations of software effort for NASA software ventures and demonstrated that the model proposed by them furnish a good estimation capability as contrast with traditional models.

An examination of the estimation of software size, schedule of the project of the project of the corporation has been undertaken by [Carr,1997] based upon the information kept by the consultancy in over the past few years. From the investigation it was uncovered that the process of evaluating effort is effective yet process of assessing schedule is definitely not. schedule slip when bewildering details are executed; unanticipated change solicits are entertained from customers, because of absence of proper framework, personnel availability and training, and errors of those who make beginning evaluations. Since effort and schedule are quite closely related to each other. While, the estimation of effort has been correct the schedule evaluations slip. This implies that the successful person-week is not the particular case that has been assumed in the past. The true number of hours of viable work done per day is fewer than needed because of the specific external constraints operating in the organization.[Carr,1997] proposed that the problem can be handled in two ways. Firstly, make all future estimates of the schedule dependent upon the current effective and real person-week. This will help upholding project plans in the short run. Furthermore, taking steps to wipe out the superfluous impacts that stop schedule will encourage projects stick to schedule with ease.

V RAW INFORMATIONAL FRAMEWORK: FINDING AND CONCLUSIONS

Total number of journals studied related to CQE are listed in Table 1.The impact factors of theses journals varies from 0.17 to 6.81. From these research papers eight major dimensions are identified and sixty eight resultants parameters are explored and listed in table 2.These eight dimensions include size, NFRs, human, economical etc. From the table it is clear that factors affecting CQE depends upon size of program followed by NFRs, quality and human factors. This variability is shown in table 3.The conceptual and empirical study was done on these eight factors and from this analysis it is observed that size effects the CQE in conceptual and empirical study and the same is listed in table 4.Size is the major factors which effect CQE this is due to LOC, number of operators, identifiers, size of the software etc. Another factor of concern is NFRs and quality which is influenced by usability, portability, performance, flexibility etc. Human errors are also among the various factors which influence CQE these are due to team leadership, communication gap, errors in repair, maintenance and design. Quality and NFRs are important for software industry dealing in reengineering of legacy program and effective implementation of CQE will ultimately reduce the complexity and cost of software and will be ensure high quality software development with optimum effort and time.

Despite rapid gains in technology, size followed by NFRs, quality attributes and human factors are ultimately responsible for ensuring the success of reengineering of legacy program globally. However, the proper & timely education and training must be imparted and it should be knowledgeable, flexible, innovative and efficient. The effective training will definitely help in reducing the risk and effort and proper decision making by the software development team members in case of emergencies. Quality is the most important factor for the software industry. Still there are many challenges in the reengineering industry regarding CQE and furthermore, the likelihood of increase in CQE can be reduced and software efficiency, performance, security and communication systems can be greatly improved.

REFERENCES

[1] Adam A. Porter and Richard W. Selby, 1992,"Empirically Guided Software Development using Metric Based Classification Trees," Department of Information and Computer Science Univ. of California,pp.1-28.

[2] Alaa Sheta et.al, 2008, "Development of Software Effort and Schedule Estimation Models Using Soft Computing Techniques," IEEE Congress on Evolutionary Computation pp.1283-1289.

[3] Banker, R.D., Datar, S.M., Zweig, D., " Software Complexity and Maintainability" Cite Seer Scientific Literature Digital Library and Search Engine.

[4] Barbacci, M.R. et.al, "Quality Attributes of a Software Architecture"Available at: http://www.vrsiddhartha.ac.in/technology/quality.pdf

[5] Basci, D. and Misra, S.,2009, "Data Complexity Metrics for Web-Services," Advances in Electrical and Computer Engineering, Volume 9, Number 2, pp.9-15.

[6] Basci, D. and Misra, S.,2009, "Measuring and Evaluating a Design Complexity Metric for XML Schema Documents CODE," JOURNAL OF INFORMATION SCIENCE AND ENGINEERING, PP.1415-1425.

[7] CHIDAMBER S.R., KEMERER, C.F.,1994, " A METRIC SUITE FOR object oriented design," IEEE Transactions Software Engineering, SE-6,pp.476-493.

[8] Christopher Deephouse et.al, 1995, "The Effects of Software Processes on Meeting Targets and Quality," Proceedings of the 28th Annual Hawaii International Conference on System Sciences, pp.710-719.

[9] Costagliola G., Tortora, G.,2005, " Class points: An approach for the size Estimation of Object-oriented systems," IEEE Transactions on Software Engineering, vol 31, pp.52-74.

[10] David L.Parnas, 1979, "Designing software for ease of extension and contraction," IEEE trans. on software engineering, Vol. SE-5, No.2, pp.128-135.

[11] Galin, D.: Software Quality Assurance", Pearson Addison Wesley ,2004.

[12] Giuseppe Visaggio , " Comprehending the Knowledge Stored in Aged Legacy Systems to Improve their Qualities with a Renewal Process"

[13] Hughes, B., Cotterell, M.: Software Project Management, 4th Edition. McGraw-Hill, 2006.

[14] Magne Jørgensen and Kjetil Moløkken-Østvold, 2004, "Reasons for Software Effort Estimation Error: Impact of Respondent Role, Information Collection Approach, and Data Analysis Method," IEEE transactions on Software Engineering, vol. 30,No. 12, pp.993-1007.

[15] Mahil Carr, 1997, "Software Effort and Schedule Estimation: A Case Study."

[16] Raquel Hill et.al, 2001, "Quantifying Non-Functional Requirements: A Process Oriented Approach," Proceedings of the 12th IEEE International Requirements Engineering Conference, pp. 1-2.

[17] Robert B. Grady, 1992, Effort, Prentice Hall PTR, pp. 1-32

[18] [18] Ray W. Wolverton, 1974,"The Cost of Developing Large-Scale Software," IEEE Trans. on computers, Vol. c-23, No.6, pp.615-636.

[19] [19] Victor R. Basili and Karl Freburger, "Programming Measurement and Estimation in the Software Engineering Laboratory," The Journal of Systems and Software, Vol. 2, pp. 47-57.

TABLE I
LIST of JOURNALS CONTRIBUTED in STUDY

S. No.	Name of Journals	Impact Factor
1.	IEEE transactions on computers	1.103
2.	IEEE	6.81
3.	IEEE transaction on systems, man, and cybernetics—part a: systems and humans	2.123
4.	Can. J. Elect. Comput. Eng	0.241
5.	ACM transaction on information systems	1.085
6.	International journal of soft computing	0.5790
7.	IEEE transactions on information theory	3.009
8.	Communication of ACM	1.92
9.	Mathematical and computer modeling	1.346
10.	IEEE transactions on software engineering	1.98
11.	International journal of computer applications	0.835
12.	International journal of Technology	0.5021
13.	Journal of information science and engineering	0.17
14.	The journal of systems and software	0.836
15.	IEEE journal on selected areas in communications	3.413
16.	ICSE ACM	
17.	MIS quartely	4.447
18.	IEEE transactions on software engineering	
19.	Information and software technology	1.250
20.	International journal of information technology And management information system	1.7251
21.	International journal of computer science and information security	0.421
22.	International journal of computer science issues	0.242
23.	International journal of forecasting	1.485
24.	International journal of computers & technology	1.043
25.	Empirical software engineering	1.85
26.	International journal of forecasting	1.485
27.	International journal of computer engineering & Technology	3.9580
28.	Journal of system and software	0.836
29.	International Journal of Software Engineering And Knowledge Engineering	0.447
30.	International Journal of Information Security	0.421
31.	Journal of Software Maintenance And Evolution: Research And Practice	0.606
32.	Artificial Intelligence Review	1.213
33.	International Journal of Computer Application	0.835
34.	International Journal of Applied Mathematics And Computer Science	0.487
35.	Software Quality Journal	0.417
36.	ACM Transactions On Software Engineering And Methodology	3.958
37.	IEEE Software	1.51

38.	Journal of the ACM	2.353
39.	IEEE Transactions On Computers	1.103
40.	IEEE Computer	1.47
41.	Software-Practice & Experience	0.519
42.	International Journal of Computer Science Engineering and Information Technology Research	3.87
43.	International Journal of Computer Science Issues	0.242
44.	Software And Systems Modeling	1.06
45.	IEEE Transactions on Systems, Man & Cybernetics: B	3.08
46.	Information Systems Research	20146
47.	IBM Journal of Research and Development	0.72
48.	IEEE Transactions on Reliability	1.29
49.	Mathematical and Computer Modeling	1.346
50.	Journal of Software Maintenance and Evolution Research and Practice	0.84
51.	Software Quality Journal	0.417
52.	Information Theory	3.009
53.	ACM Journal on Emerging Technologies in Computing Systems	0.41
54.	Automated Software Engineering	0.857
55.	Journal of information technology	2.321

Table II
MAJOR DIMENSIONS and VARIABLES

Major Dimensions	**Variables**
Human Factor	1. Team experience 2. Team leadership 3. Redundancy 4. Consistency 5. Error proneness 6. Non error prone 7. Software development methodology 8. Design 9. Manpower 10. Customer satisfaction & expectations 11. Communication with users 12. Expert judgments
Quality attributes	13. Software reliability 14. Software productivity 15. Software reusability 16. Software flexibility 17. Software usability
Economic factor	18. Maintenance cost 19. Actual cost 20. Annual expenditure 21. Budgeted cost 22. Algorithmic & non algorithmic cost 23. Cost 24. Cost overruns 25. Average salary 26. Price
NFR(quantifiable NFRs, non quantifiable NFRs)	27. Performance 28. maintenance

	29. effectiveness 30. efficiency 31. Accuracy 32. Clarity & Precision 33. Security 34. Robustness 35. Portability
Size	36. Loc 37. No. Of operators & operands 38. Identifiers and nesting levels 39. Software size 40. Control structures 41. Descriptive and prescriptive components 42. No of I/O
Environment	43. Volatility 44. Environment dynamics 45. homogeneous environment 46. production environment 47. external or environmental consistency
Technical and risk	48. Reliability Toolkit 49. automatic or semi-automatic detection and correction techniques and tools 50. development tool 51. verification tools 52. Language Tool 53. software tool 54. Functional risk 55. Political risk 56. Financial risk 57. Technical risk 58. Project risk
Time	59. Development time 60. Elapsed time 61. Number of executions 62. Time to market 63. Resource allocation
Misleneous	64. Competition 65. Process training 66. Qualitative and quantitative information 67. Software field quality 68. Staff resources or effort

TABLE III
NUMBER and PERCENTAGE of STUDIES

Dimensions	**Number of Studies**	**Percentage of Studies**
Human Factor	19	13.86
Quality attributes	21	15.34
Economic factor	17	12.43
NFRs	29	21.18
Size	36	26.27
Environment	5	3.64
Technical and risk	6	4.37
Misleneous	4	2.91

TABLE IV
CONCEPTUAL and EMPIRICAL STUDIES PERCENTAGE

Dimensions	**Conceptual Studies**		**Empirical Studies**	
	Number of Articles	**%age**	**Number of Articles**	**%age**
Human Factor	12	8.75	07	5.09
Quality attributes	17	12.40	04	2.91
Economic factor	11	8.02	06	4.37
NFRs	21	15.32	08	5.83
Size	29	21.16	07	5.09
Environment	03	2.18	02	1.45
Technical and risk	04	2.91	02	1.45
Misleneous	03	2.18	01	0.72