Identification, Categorization and Weighting of Software Engineering Attributes for Quality Evaluation of Software Project Documentation

Vikas S. Chomal
[1]Assistant Professor
[2]Research Scholar
[1]The Mandvi Education Society Institute of Computer Studies, Mandvi, Gujarat, India
[2]Singhania University, Pacheri Bari, District – Jhunjhunu, Rajasthan
Email: vikschomal80@gmail.com
Dr. Jatinderkumar R. Saini
[1]Director I/C & Associate Professor
[2]Research Supervisor
[1]Narmada College of Computer Application,
Bharuch, Gujarat, India
[2]Singhania University, Pacheri Bari, District – Jhunjhunu, Rajasthan
Email: saini_expert@yahoo.com

---ABSTRACT---

Software project documentation is an editorial whose intention is to converse information about the software system. An elementary objective of software engineering is to construct the finest potential operational software along with the most excellent supporting documentation. This paper highlights the results of analysis of software project documentations of large software projects. Documentations of final year students of Masters level course have been considered for the research purpose. These documentations consist of the artefacts like requirement analysis, technical environment, database design, structural and object oriented modelling techniques, screen layouts and testing techniques along with test case and data. The results are compiled from more than 500 large software project documentations developed during a period of academic years from 2001-2002 to 2011-2012. After compilation of results and studying various artefacts in software project documentation, we categorized artefacts into two broad categories (a) Quantifiable attributes and (b) Non-quantifiable attributes. Further, after categorization, weights are assigned to these attributes for scoring documentation of student software project.

Keywords: Container Relationship, Non – Quantifiable, Quantifiable, Software Attributes, Software Documentation, Software Development, Software Engineers, Software Projects, Weight Assignment

I. INTRODUCTION

Software documentation is an essential feature of both software projects and software engineering in common. In piece of evidence, documentation engineering has become an accepted sub-domain in the software engineering society. The task of documentation in a software engineering milieu is to commune information to its spectators and instils knowledge of the system it describes [1]. According to Sommerville [8], documents associated with a software project and the systems being developed have a number of associated requirements:

1. They should act as a communication medium between members of the development team.
2. They should be a system information repository to be used by maintenance engineers.
3. They should provide information for management to help them plan, budget and schedule the software development process.
4. Some of the documents should tell users how to use and administer the system.

Software development is partly a learning and communication process. Software developers need to communicate with each other and also with various interest groups of the system to be developed, such as customers, marketing people, end users, service personnel, and authorities. Documentation is the basis for communication in software development organizations as well as between development organizations and the interest groups of the system to be developed. To ensure efficient communication, all communicating parties need to be able to identify various software documents, and, to ensure that the right information is found, all communicating parties should be able to anticipate what information is in each document [10][14]. Cock & Visconti [11] elucidate that empirical data shows that software documentation products and processes are key components of software quality. These studies show that poor quality, out of date, or missing documentation is a major cause of errors in software development and maintenance. For example, the majority of defects discovered during integration testing are design and requirements defects, e.g. defects in documentation that were introduced before any code was written.

The paper is divided into five sections; first section is introductory, followed by literature review. Section three represents methodology followed by finding and analysis and at last concluding section.

II. RELATED LITERATURE REVIEW

Forward and Lethbridge [2,3] states that, documentation attributes describe information about a document beyond the content provided within. Example attributes include the document's writing style, grammar, extent to which it is up to date, type, format, visibility, etc. Documentation artefacts consist of whole documents, or elements within a document such as tables, examples, diagrams, etc. An artefact is an entity that communicates information about the software system. According to Boer [13], the effectiveness of documentation within a development process is determined by the way in which the intentions of the authors correspond to the expectations of the potential readers. In a typical software development process, many different kinds of documents are produced and consumed at various points in time. The contents of those documents necessarily exhibit a certain amount of overlap. People may lose track of the meaning of individual documents; which information it contains and what its role is in the development process.

Sulaiman and Sahibudding [15] puts forward that system documentation (SD) is undoubtedly vital as one of the sources in software understanding. Despite its importance, practitioners are often confronted with the problems related to system documentation. A number of tools have been introduced in order to assist documenting activities. However such tools are still not widely used because they generally fail to meet users' needs. Briand [5] focuses that, it is a well-known fact that software documentation is, in practice, poor and incomplete. Though specification, design, and test documents-among other things-are required by standards and capability maturity models (, such documentation does not exist in a complete and consistent form in most organizations. When documents are produced, they tend to follow no defined standard and lack information that is crucial to make them understandable and usable by developers and maintainers. Then a fundamental practical question, which motivated this keynote address, is to better understand what type of documentation is required, what is needed to support its completeness and consistency, and what is the level of precision required for each type of document. These questions cannot be investigated at that level of generality though. Answers are likely to be very context-dependent if they are to be precise. Briand [5] research work focuses on object-oriented development and the Unified Modeling Language (UML).

Nasution and Weistroffer [12] lays down that, a well planned and documented systems development project is more likely to result in a system that meets the expectations of both the intended users and the software engineers. Arthur and Stevens [4] in their work projected that, the investigation focuses on assessing the adequacy of project documentation based on an identified taxonomic structure relating documentation characteristics. Previous research in this area has been limited to the study of isolated characteristics of documentation and English prose, without considering the collective contributions of such characteristics. The research described takes those characteristics, adds others and establishes a well-defined approach to assessing the `goodness' of software documentation.

Chomal and Saini [18] in their work stated that, if requirements are not properly specified, analyzed and properly documented, then it will lead to software as a failure. Delaney and Brown [7] proposed a technical report which outlines the contents of a minimal set of software development documents, tailored for use by students in software engineering projects, and firmly based on IEEE standards. The document set is designed to support software development activities. It provides a framework for use in undergraduate software engineering projects, both individual and team-based, that helps students to learn best practice. A supplementary report describes the content of each document in more detail. They also suggested and identified the minimal core set of software, and identified the activities that produce them, which is described in Table – 1.

Table – 1: [9]

Document	**Deliverables**	**Description Activities**
Software Project Management Plan (SPMP)	Description of the software approach and associated milestones.	System requirement analysis Software requirement analysis
Software Requirements Specifications (SRS)	Description of the expected software features, constraints, interfaces and other attributes.	Process implementation
Software Design Description (SDD)	Description of how the software will meet the requirements. Also describes the rationale for design decisions taken.	System architectural design Software architectural design Software detailed design
Software Test Documentation (STD)	Description of the plan and specifications to verify and validate the software and the results.	Software qualification testing System qualification testing

According to Chomal and Saini [17, 19] and Forward [3], software documentation is an essential feature of both software projects and software engineering in common. In piece of evidence, documentation engineering has become an accepted sub-domain in the software engineering society. The task of documentation in a software engineering milieu is to commune information to its spectators and instils knowledge of the system it describes. Abdulaziz et al [9] conducted an empirical investigation using a comparative case study research method. The basis for the work was concerned with the requirements for information system documentation. Jazzar's work resulted in eight hypotheses that attempt to model the requirements for achieving effective, high quality documentation products and processes.

Chomal and Saini in [21] focuses that, documentation is the written record of what the software is supposed to do, what it does, how it does it and how to use it. Virtually everyone agrees that good documentation is important to the analysis, development and maintenance phases of the software process and is an important software product. Forward [3] discusses how certain attributes contribute to a document’s effectiveness. They conducted a survey and asked the participants how important particular document attributes contribute to its overall effectiveness. Participants gave rating between 1 (least important) and 5 (most important). Table - 2 lists the attributes considered in the question in descending order based on the attributes perceived contribution to a document’s effectiveness.

According to Chomal and Saini [16, 20] and Visconti and Cook [6] points up that, documentation seems to be considered a second class object and not as important as the software itself. However, empirical data shows that low quality or missing documentation is a major cause of errors in software development and maintenance. Low quality or missing documentation is a major cause of errors in software development and maintenance.

Table – 2: Document attributes and effectiveness [3]

Document Attribute	Mean	Std.	dev.	% Rate
Content – the document’sInformation	4.85	1.57	85 %	0 %
Up-to-date	4.35	0.89	46 %	0 %
Availability	4.19	0.79	41 %	4 %
Use of examples	4.19	0.85	37 %	4 %
Organization – sections /Subsections	3.85	0.64	30 %	4 %
Type – req, spec, design,etc.	3.78	0.63	26 %	11 %
Use of diagrams	3.44	0.60	15 %	22 %
Navigation – quality of internal / external links	3.26	0.44	19 %	33 %
Structure – arrangement of text, diagrams, figures	3.26	0.60	11 %	22 %
Writing Style – sentence / paragraph structure, Grammar	3.26	0.67	7 %	19 %
Length – not too long or Short	3.15	0.64	7 %	22 %
Spelling and grammar	2.93	0.85	0 %	22 %
Author	2.63	0.41	7 %	48 %
Influence to use it	2.62	0.48	12 %	50 %
Format – pdf,, doc, txt, xml, etc.	2.42	0.58	0 %	54

III. METHODOLOGY

Chomal and Saini [17] in their work considered documentation of software projects prepared by students as a source for data collection. Specifically, documentations of large software projects of only final year students of Masters level course have been considered for the research purpose. The duration of these software projects is six months. The said documentations of software projects were procured from college libraries. These documentations include complete project profile along with the following elements:

1) Requirement analysis
2) Technology used
3) Database design
4) Structural and Object Oriented Modelling Techniques
5) Screen layouts
6) Testing techniques along with test case and data

We analyzed and reviewed 505 large software project documentations developed during a period of academic years from 2001-2002 to 2011-2012. During our exploration we considered all of the above described elements. For simplicity and better exhaustive analysis of the documentations, the phased process was followed. As each project is a uniquely different definition from other projects, it is noteworthy here that this was repeated for each of the 505 project reports under study. These phases are presented below:

1) Exploration of Project Profile
2) Exploration of Existing System and Proposed System
3) Exploration of Requirement Gathering Techniques
4) Exploration of Requirement Analysis done by Students
5) Exploration of Technology on which Software Project carried out
6) Exploration of Process Model adapted for Software Project Development
7) Exploration of Data Dictionary (including Database Design)
8) Exploration of various Structural and Object Oriented Modelling Techniques
9) Exploration of Screen Layouts
10) Exploration of Generated Reports
11) Exploration of Testing Techniques and Test data

In the present work, we identified 103 software attributes from software project documentations which are mentioned in Table – 3.

Table – 3: Software Attributes

Sr No.	Attribute	Sr	Attribute	Sr No	Attribute
1	Acceptance Testing	36	Functional Requirement	71	Software Life Cycle
2	Activity Diagram	37	Functionality	72	Software Project
3	Adaptive Maintenance	38	Gantt Chart	73	Software Quality
4	Alpha Testing	39	Integration Testing	74	Software Requirement
5	Beta Testing	40	Levels of Testing	75	Software Size
6	Black Box Testing	41	Milestone	76	State Based Testing
7	Boundary Value Analysis	42	Non-Functional	77	State Diagram
8	Branch Testing	43	Normal Requirement	78	Structured Design
9	Bug	44	Object Oriented Analysis	79	System
10	Class Diagram	45	Object Oriented Design	80	System Testing
11	Code	46	Operational Feasibility	81	Table Relationship
12	Control Flow Based Testing	47	Path Testing	82	Technical Feasibility
13	Corrective Maintenance	48	Perfective Maintenance	83	Test Case Design
14	Critical Path Method	49	Process model	84	Test Case Execution and
15	Data Dictionary	50	Process Specification	85	Test Case Generation
16	Data Flow Diagram	51	Project Monitoring and	86	Test Case Review
17	Debugging	52	Project Planning	87	Test Case Specification
18	Defect	53	Project Progress	89	Test Cases
19	Defect Removal Efficiency	54	Project Scheduling	90	Test Data
20	Design	55	Project Tracking	91	Test Driven
21	Design Constraints	56	Quality Function	92	Test Plan
22	Document Structure	57	Regression Testing	93	Testing
23	Economic Feasibility	58	Reliability	94	Testing Process
24	Effort	59	Reports	95	Time Line Chart
25	Entity Relationship Diagram	60	Requirement Analysis	96	Unified Modelling
26	Equivalence Class	61	Requirement Validation	97	Unit Testing
27	Errors	62	Requirements	98	Usability
28	Excited Requirement	63	Risk Management	99	Use Cases
29	Expected Requirement	64	Sequence Diagram	100	User Interface
30	External Interface	65	Size Estimation	101	Validation
31	Failure	66	Smoke Testing	102	Verification
32	Fault Tolerance	67	Software	103	White Box Testing
33	Faults	68	Software Documentation		
34	Feasibility Study	69	Software Engineering		
35	Formal Technical Review	70	Software Environment		

The next section presents the findings obtained through analysis of documentation reports.

IV. FINDINGS AND ANALYSIS

Chomal and Saini [17] in their work considered documentation of software projects prepared by students as a source for data collection. During presenting and analysing, they also identified points which can be termed as the errors. There were eleven broad categories under which various errors were found. These broad categories are:

1) Process Model
2) Data Flow Diagram
3) Process Specification
4) Entity Relationship Diagram
5) Form Design / User Interface

6) Database Design
7) Code Design
8) Exception Handling
9) Reports
10) Testing
11) Documentation

Further, from Table – 3 which consists of software attributes, we categorize them into two broad categorization (a) Quantifiable attributes and (b) Non-quantifiable attributes. Quantifiable attributes are those attributes which are considered as a metrics for measuring software project documentations. Whereas, Non-quantifiable attributes are those attributes which are not regarded as a metrics for evaluating software project documentations. From Table – 3, we present quantifiable and non – quantifiable attributes in Table – 4 (a) and Table – 4 (b), which are further classified on the basis of container relationship and they are arranged in alphabetical order. Container relationships characterize regarding category and their sub categories, for example in Table – 4(a) we stated requirement analysis as main category and its various sub categories are (i) excited requirement, (ii) expected requirement, (iii) functional requirement and so on. For the current work, we have considered only those attributes which can be quantified easily and the other attributes have been treated as non-quantifiable attribute.

Table – 4: (a) Quantifiable Attributes

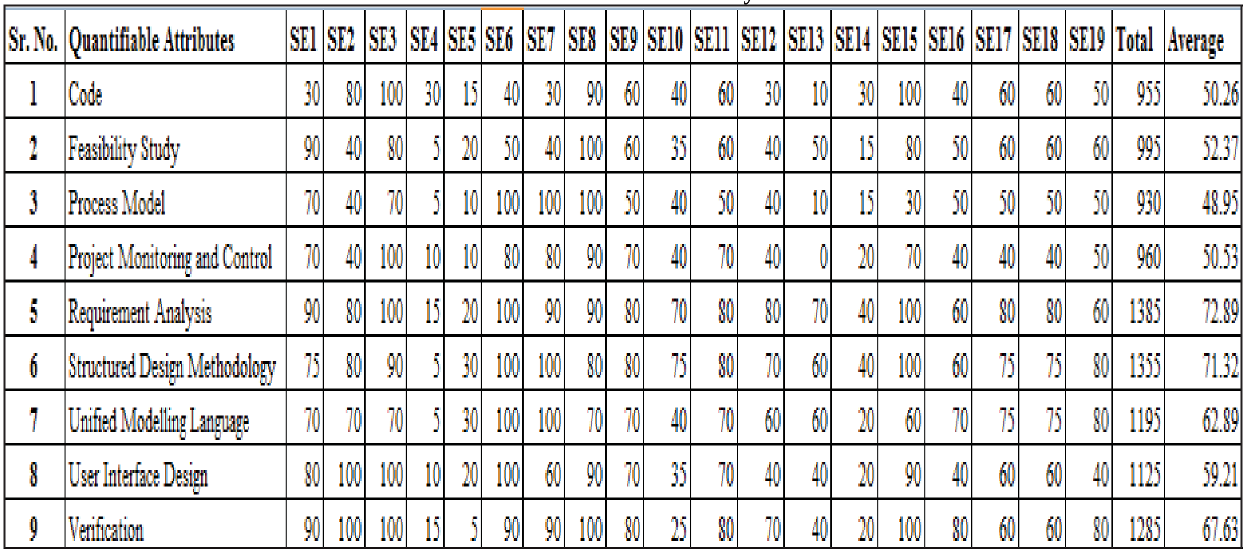
Sr No.	Attributes	Sr No.	Attributes
1.	Code	6.	Requirement Analysis (a) Expected Requirement (b) External Interface Requirement (c) Functional Requirement (d) Non-Functional Requirement (e) Normal Requirement (f) Requirement Validation
2.	Design (a) Design Constraints (b) User Interface	7.	Structured Design Methodology (a) Data Dictionary (b) Data Flow Diagram (c) Entity Relationship Diagram (d) Process Specification (e) Table Relationship Diagram
3.	Feasibility Study (a) Economic Feasibility (b) Operational Feasibility (c) Technical Feasibility	8.	Unified Modelling Language (a) Activity Diagram (b) Class Diagram (c) Object Oriented Analysis (d) Object Oriented Design (e) Sequence Diagram (f) Use Cases
4.	Process model	9.	Verification
5.	Project Monitoring and Control (a) Critical Path Method (b) Gantt Chart (c) Project Planning (d) Project Progress (e) Project Scheduling (f) Project Tracking (g) Time Line Chart		

Table – 4: (a) Non-Quantifiable Attributes

Sr No.	Attributes	Sr No.	Attributes
1.	Maintenance (a) Adaptive Maintenance (b) Corrective Maintenance (c) Perfective Maintenance	5.	Testing (a) Acceptance Testing (b) Alpha Testing (c) Beta Testing (d) Black Box Testing (e) Boundary Value Analysis (f) Branch Testing (g) Bugs (h) Control Flow Based Testing (i) Defect (j) Equivalence Class Partitioning (k) Errors (l) Failure (m) Faults (n) Integration Testing (o) Levels of Testing (p) Path Testing (q) Regression Testing (r) Smoke Testing (s) State Based Testing (t) System Testing (u) Test Case Design (v) Test Case Execution and Analysis (w) Test Case Generation (x) Test Case Review (y) Test Case Specification (z) Test Cases (aa) Test Data (bb) Test Driven Development (cc) Test Plan (dd) Testing Process (ee) Unit Testing (ff) Usability (gg) Validation (hh) White Box Testing
2.	Risk Management	6.	Validation
3.	Size Estimation (a) Effort (b) Software Size	7.	Others (a) Debugging (b) Defect Removal Efficiency (c) Fault Tolerance (d) Formal Technical Review (e) Functionality (f) Milestone (g) Quality Function Deployment (h) Reliability (i) Reports
4.	Software Engineering (a) Documentation Structure (b) Software (c) Software Environment (d) Software Life Cycle (e) Software Project (f) Software Quality (g) Software Requirement (h) Specification System		

Further, after classifying attributes into quantifiable and non-quantifiable categories, we assigned weights only to quantifiable attributes. For assigning weights, we randomly selected one quantifiable attribute to begin with and proceeding to other attributes while keep on comparing the already assigned weights and the attributes to which weights are to be assigned. This practice was affirmed by conducting a small survey for assigning weights to 9 quantifiable attributes by 19 software engineers. The results of the survey are presented in tabular format in Table 5, wherein 'SEn' indicates the values provided by nth Software Engineer, with n ranging from 1 to 19.

Table – 5: Survey Result



Sr. No.	Quantifiable Attributes	SE1	SE2	SE3	SE4	SE5	SE6	SE7	SE8	SE9	SE10	SE11	SE12	SE13	SE14	SE15	SE16	SE17	SE18	SE19	Total	Average
1	Code	30	80	100	30	15	40	30	90	60	40	60	30	10	30	100	40	60	60	50	955	50.26
2	Feasibility Study	90	40	80	5	20	50	40	100	60	35	60	40	50	15	80	50	60	60	60	995	52.37
3	Process Model	70	40	70	5	10	100	100	100	50	40	50	40	10	15	30	50	50	50	50	930	48.95
4	Project Monitoring and Control	70	40	100	10	10	80	80	90	70	40	70	40	0	20	70	40	40	40	50	960	50.53
5	Requirement Analysis	90	80	100	15	20	100	90	90	80	70	80	80	70	40	100	60	80	80	60	1385	72.89
6	Structured Design Methodology	75	80	90	5	30	100	100	80	80	75	80	70	60	40	100	60	75	75	80	1355	71.32
7	Unified Modelling Language	70	70	70	5	30	100	100	70	70	40	70	60	60	20	60	70	75	75	80	1195	62.89
8	User Interface Design	80	100	100	10	20	100	60	90	70	35	70	40	40	20	90	40	60	60	40	1125	59.21
9	Verification	90	100	100	15	5	90	90	100	80	25	80	70	40	20	100	80	60	60	80	1285	67.63

We now present the weights averaged based on the values provided by 19 software engineers in Table 6. It is noteworthy to mention that each of the 9 quantifiable attributes were assigned weight out of 100 and it was not necessary to have the total of weights of 9 attributes as break-up of 100. In scientific research community, this practice is technically known as based on human perception and general intelligence.

Table – 6: Weight Assignments to Quantifiable Attributes

Sr No.	Quantifiable Attribute	Average (%)
1.	Code	50.26
2.	Feasibility Study	52.36
3.	Process Model	48.94
4.	Project Monitoring and Control	50.52
5.	Requirement Analysis	72.89
6.	Structured Design Methodology	71.31
7.	Unified Modelling Language	62.89
8.	User Interface Design	59.21
9.	Verification	67.63

Based on the average values presented in Table 6, it has been found that the software engineers give maximum weight to Requirement Analysis (72.89 %), while Structured Design Methodology (71.31%) was found to have achieved the second highest weight. Similarly, the minimum weight was found to be assigned to Process Model (48.94%) while the second lowest weight was found to be achieved by Code (50.26%).

V. CONCLUSION

In the present work, we identified 103 software attributes from software project documentations. Further we categorize these software attributes into two broad categorization (a) Quantifiable attributes and (b) Non-quantifiable attributes. Quantifiable attributes are those attributes which are considered as a metrics for measuring software project documentations. Whereas, Non-quantifiable attributes are those attributes which are not regarded as a metrics for evaluating software project documentations. The list of 103 software attributes, which we categorized into quantifiable and non-quantifiable are most relevant software attributes according to us. Further we do not claim that the lists of these 103 software attributes are exhaustive listing. The basic goal of assigning weights to quantifiable attributes is to score software project documentation.

REFERENCES

[1] Andrew J. Forward, *"Software Documentation – Building and Maintaining Artefacts of Communication"*, presented to the Faculty of Graduate and Postdoctoral Studies in partial fulfilment of the requirements for the degree Master in Computer Science, Ottawa – Carleton Institute of Computer Science, University of Ottawa, Canada, 2002.

[2] Andrew J. Forward, "The Relevance of Software Documentation, Tools and Technologies: A Survey"

[3] Arthur J.D , Stevents K.T, "Assessing the adequacy of documentation through document quality indicators", Software Maintenance, 1989., Proceedings., Conference on DOI: 10.1109/ICSM.1989.65192 Publication Year: 1989 , Page(s): 40 - 49 Cited by: Papers (6) | Patents (1) IEEE CONFERENCE PUBLICATIONS

[4] Briand. L. C, "Software documentation: how much is enough?" Published in: Software Maintenance and Reengineering, 2003. *Proceedings, Seventh European Conference on 26 – 28 March 2003, pages 13 – 15, ISSN – 1534 – 5351.*

[5] Curtis R. Cook, Marcello Visconti, "NEW AND IMPROVED DOCUMENTATION PROCESS MODEL", *Proceedings of the 14th Pacific Northwest Software Quality Conference, 1996.*

[6] Declan Delaney, Stephen Brown, "DOCUMENT TEMPLATES FOR STUDENT PROJECTS IN SOFTWARE ENGINEERING", Department of Computer Science, National University of Ireland, Maynooth Date: August 2002 *Technical Report: NUIM-CS-TR2002-05*

[7] Ian Sommerville, Software Documentation, Lancaster University, UK Issue 3, August 2000.

[8] Jazzar, Abdulaziz , Walt Scacchi, "Understanding the requirements for information system documentation: an empirical investigation", *COOCS `95, Sheraton Silicon Valley, California, USA, ACM Press, p268 – 279.*

[9] Kari Laitinen, "Document Classification for Software Quality Systems", Technical Research Centre of Finland (VTT) Computer Technology Laboratory, ACM SIGSOFT SOFTWARE ENGINEERING NOTES vol 17 no 4 Oct 1992 Page 32

[10] Marcello Visconti , Curtis Cook., "Software System Documentation Process Maturity Model", *Proceeding CSC '93 of the 1993 ACM conference on Computer Science Pages 352 – 357 New York, USA, (1993).*

[11] Nasution, M.F.F. ; Weistroffer, H.R., "Documentation in Systems Development: A Significant Criterion for Project Success"

[12] Remco C. de Boer, "Writing and Reading Software Documentation: How the Development Process may Affect Understanding"

[13] Scheff, Benson H. and Tom Georgon., "Letting software engineers do software engineering or freeing software engineers from the shackles of documentation", p81 – 91, SIGDOC '88, Ann Arbor, Michigan, USA, ACM Press, 1988.

[14] Sulaiman. S, Sahibudding. S "Production and maintenance of system documentation: what, why, when and how tools should support the practice", Published in: Software Engineering *Conference, 2002. Ninth Asia-Pacific*, pages 558 – 5667, ISSN – 1530 – 1362.

[15] Vikas S. Chomal, Dr. Jatinderkumar R. Saini, " Cataloguing Most Severe Causes that lead Software Projects to Fail", *International Journal on Recent and Innovation Trends in Computing and Communication , May – 2014* ISSN: *2321-8169* Volume: 2 Issue: 5 pages 1143– 1147

[16] Vikas S. Chomal, Dr. Jatinderkumar R. Saini, ―Finding Trend of Both Frequency and Type of Errors from Software Project Documentation‖, *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)ISSN 2278-6856*, Volume 2, Issue 5, September – October 2013

[17] Vikas S. Chomal, Dr. Jatinderkumar R. Saini, ―Identification and Analysis of Causes for Software Failures‖, *NATIONAL JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY]* Volume: 04 | Issue: 02 | July – December – 2012

[18] Vikas S. Chomal, Dr. Jatinderkumar R. Saini, ―Software Quality Improvement by Documentation – Knowledge Management Model‖, *National Journal of System And Information Technology* ISSN : 0974 – 3308, Volume 6, Number 1, June 2013, Page Number: 49 – 68

[19] Vikas S. Chomal, Dr. Jatinderkumar R. Saini, ―Software Template to Improve Quality of Database Design on basis of Error Identification from Software Project Documentation‖, *International Journal of Engineering and Management Research ISSN No.: 2250-0758*,Volume-4, Issue-1, February-2014, Page Number: 168-179

[20] Vikas Sitaram Chomal, Dr. Jatinderkumar R. Saini, "Significance of Software Documentation in Software Development Process" *International Journal of Engineering Innovation & Research, ISSN: 2277 – 5668,* Volume 3, Issue 4

ABOUT AUTHORS

Vikas Sitaram Chomal – M.Phil, MCA and Research Scholar at Faculty of Computer Science, Singhania University, Pacheri Bari, District – Jhunjhunu, Rajasthan – 333515. He has more than 7 years of rich teaching experience. Presently he is working as Assistant Professor at The Mandvi Education Society Institute of Computer Studies – MCA, Mandvi, District – Surat, Gujarat, India. Formely he was Assistant Professor (Ad hoc) at Narmada College of Computer Application – MCA, Bharuch, Gujarat, India. He worked as Principal (I/C) & Assistant Professor at Shri Manilal Kadakia College of Management & Computer Studies, Ankleshwar, Gujarat, India.

Jatinderkumar R. Saini was awarded Ph.D. in Computer Science by Veer Narmad South Gujarat University, Surat, Gujarat, India in the year 2009. He has more than 8 years of rich professional experience including working at Ministry of Info. Tech., New Delhi licensed CA under PKI at Ahmedabad, Gujarat, India. Presently he is working as Director (I/C) & Associate Professor at Narmada College of Computer Application, Bharuch, Gujarat, India. He is also the Director (I.T), GTU's Ankleshwar – Bharuch Innovation Sankul. Formely, he was Associate Professor & GTU Coordinator & HOD at S.P. College of Engineering, Visnagar, Gujarat, India.