Software Template for Evaluating and Scoring Software Project Documentations

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

Jatinderkumar R. Saini, Ph.D.
[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

ABSTRACT
Significant documentation is been formed with almost all software projects, irrespective of application. Software project documentation is a perspective whose purpose is to communicate information about the software system. For research purpose 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 were compiled from 505 large software project documentations developed during a period of academic years from 2001-2002 to 2011-2012. The duration of these software projects was six months. Errors from these software project documentations were found and these errors were classified into 11 broad error categories. After compilation of results and studying various artefacts in software project documentations 103 software attributes were recognized. These software attributes were classified into two broad classes (a) Quantifiable attributes and (b) Non-quantifiable attributes. Out of 103 software attributes, 39 were quantifiable attributes and 64 non- quantifiable attributes. Subsequent to categorization, weights were assigned to these quantifiable software attributes only for which a survey was conducted. The basic goal of assigning weights to these quantifiable attributes was to score software project documentation. Further, software template is been proposed for evaluating and scoring student's software projects documentation.

General Terms
Software Documentations, Software Engineering, Software Project

Keywords
Errors, Error Category, Non – Quantifiable Attributes, Quantifiable Attributes, Software Attributes

1. INTRODUCTION
Software documentation is a essential aspect 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. According to the author, the task of documentation in a software engineering milieu is to communicate information to its spectators and to instil knowledge of the system it describes [13]. Even though every software development project is exclusive and produces diverse categories of documents, different amount of documentation and may employ different documentation methods and notations, we need to be able to control the documentation produced in software development projects in a uniform manner [14, 15]. According to Cock and Visconti [11] a fundamental aim of software engineering is to generate the best possible working software along with the best supporting documentation. Software development is partly a learning and communication process. Software developers need to converse 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 base for communication in software development organizations as well as between development organizations and the interest groups of the system to be developed. To ensure well-organized 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 [8, 21].

2. RELATED LITERATURE REVIEW
Kipyegen and William as well as Laitinen [17, 18] put forward that software development is supposed to be documentation-oriented, which means that documents are considered to be the most essential and valuable products of the development process. On the other hand, a product such as executable machine code is regarded as a by-product in the development process, because in a development environment we can always derive correct executable machine code when we have the correct documents. The executable machine code is essential to the user of a computer system, but it is considered less important to the software developers. According to Boer [2], 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. Visconti and Cock [22] 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. Sulaiman and Sahibudding [21] put 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 [3] 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.

Nasution and Weistroffer [19] lay 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 [1] projected that, the investigation focuses on assessing the adequacy of project documentation based on an identified taxonomic structure relating documentation characteristics. Chomal and Saini [5] 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 [12] 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. According to Chomal and Saini [4, 6] and Arthur [3], software documentation is an essential feature of both software projects and software engineering in common. Abdulaziz et al [16] 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. Chomal and Saini [9] advocated 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 [14] discusses how certain attributes contribute to a document's effectiveness. According to Chomal and Saini [4,8] and Visconti and Cook [22] 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.

Chomal and Saini [7] state that the basic goal of software engineering is to produce the best possible working software along with the best possible supporting documentation. Hilburn and Towhidnejad [15] place forward that computer science curriculum designers and developers treat software as a central element of their curricula. Software quality should be emphasized in the beginning and throughout the curriculum. We believe that it is essential that a software development process that emphasizes quality techniques, methods, and analysis be taught to and used by students throughout their projects. Software documentation is an critical activity in software engineering. Documentation improves on the quality of a software product [20].

3. METHODOLOGY

Chomal and Saini [5] considered 505 large software project documentations prepared by students during period of academic year 2001-2002 to 2011-2012 as a source for data collection. Particularly, documentations of large software projects of only final year students of Masters level course were considered for the research purpose. The duration of these software projects was six months. The said documentations of software projects were procured from college libraries. These documentations included complete project profile covering Requirement analysis, Technology used, Database design, Structural and Object Oriented Modelling Techniques, Screen layouts and Testing techniques along with test case and data. During the exploration of projects, all these elements were reviewed. 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 documentations 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

Also during exploration of these software project documentations, 103 software attributes were identified which are listed in Table 1.

Table 1: Software Attributes [10]

Sr No.	Attribute	Sr No.	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 Specification
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 Requirement	77	State Diagram
8	Branch Testing	43	Normal Requirement	78	Structured Design Methodology
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 Diagram
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 Analysis
15	Data Dictionary	50	Process Specification	85	Test Case Generation
16	Data Flow Diagram	51	Project Monitoring and Control	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 Development
21	Design Constraints	56	Quality Function Deployment	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 Language
26	Equivalence Class Partitioning	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 Requirement	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		

From Table 1, we categorize them into two broad categorization (a) Quantifiable attributes; are those attributes which are considered as a metrics for measuring software project documentations, and (b) Non-quantifiable attributes; are those attributes which are not regarded as a metrics for evaluating software project documentations. These attributes are further classified on the basis of container relationship and they are arranged in alphabetical order. Container relationships characterize category and its sub categories. Quantifiable and Non – quantifiable attributes are described below:

Quantifiable Attributes:

1. Code
2. Database Design
3. Design

a) Design Constraints

b) User Interface

4. Feasibility Study

a) Economic Feasibility

b) Operational Feasibility

c) Technical Feasibility

5. Process Model
6. 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

7. Requirement Analysis

a) Expected Requirement

b) External Interface Requirement

c) Functional Requirement

d) Non-Functional Requirement

e) Normal Requirement

f) Requirement Validation

8. Structured Design Methodology

a) Data Dictionary

b) Data Flow Diagram

c) Entity Relationship Diagram

d) Process Specification

e) Table Relationship Diagram

9. Unified Modelling Language

a) Activity Diagram

b) Class Diagram

c) Object Oriented Analysis

d) Object Oriented Design

e) Sequence Diagram

f) Use Cases

10. Verification

Non - Quantifiable Attributes:

1. Maintenance

a) Adaptive Maintenance

b) Corrective Maintenance

c) Perfective Maintenance

2. Risk Management

3. Size Estimation

a) Effort

b) Software Size

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

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) White Box Testing

6. Validation
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. FINDINGS AND ANALYSIS

Chomal and Saini [5] identified main errors and error categories from the review of project documentations. There were eleven broad categories described below under which various errors were found.

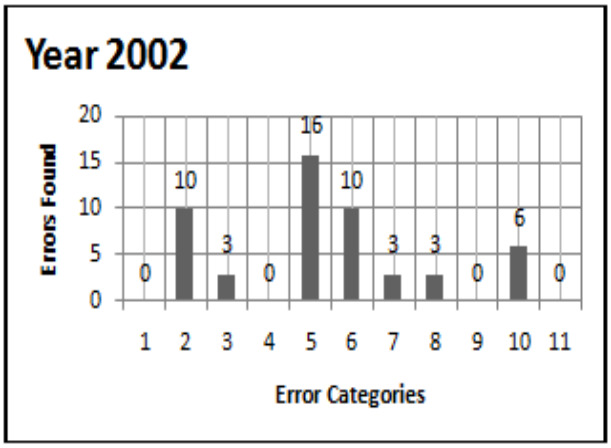
1. Process Model
2. Data Flow Diagram
3. Process Specification
4. Entity Relationship Diagram
5. Form Design / User Interface
6. Database Design
7. Code Design
8. Exception Handling
9. Reports
10. Testing
11. Documentation

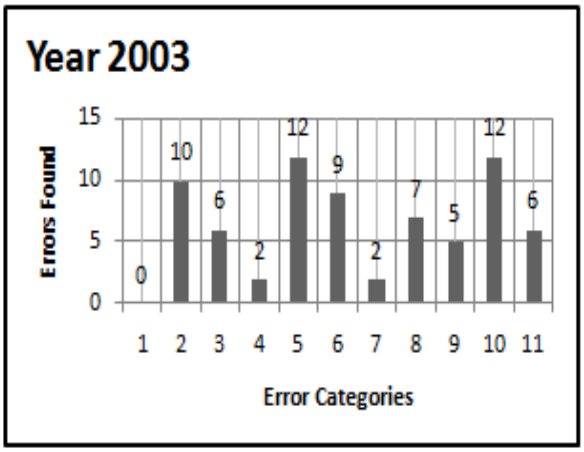
The total number of errors found during exploration which belongs to the said error categories along with the summation of total number of errors encountered for the academic years from 2001-2002 to 2011-2012 is highlighted in Table 2

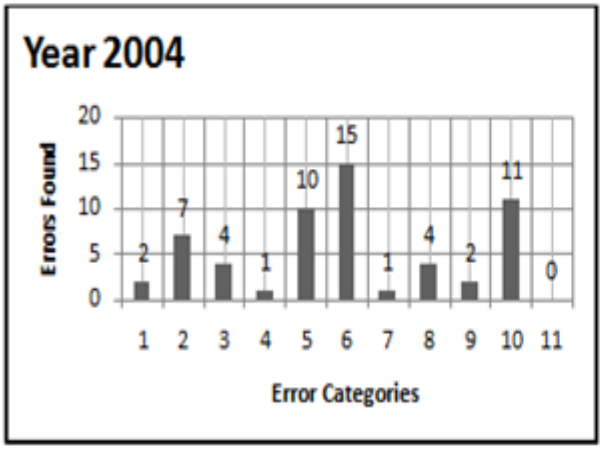
Table 2: Error Category–Wise Year-Wise Number of Errors

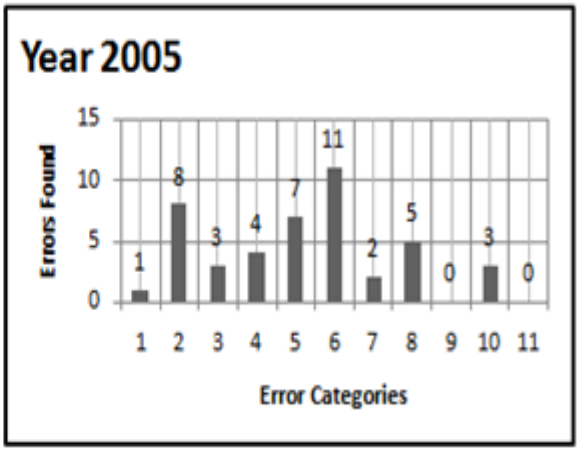
Error Categories	Year											Total
	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	
1	0	0	2	1	1	0	5	7	6	10	15	47
2	10	10	7	8	13	24	13	19	8	19	11	142
3	3	6	4	3	9	3	2	2	5	7	3	47
4	0	2	1	4	0	2	5	5	5	14	15	53
5	16	12	10	7	26	31	15	14	21	62	43	257
6	10	9	15	11	22	47	45	69	65	46	45	384

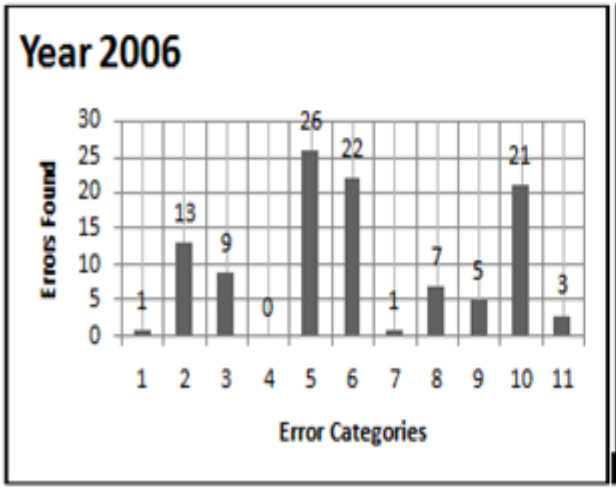
7	3	2	1	2	1	0	0	2	2	4	5	22
8	3	7	4	5	7	8	5	6	13	11	9	78
9	0	5	2	0	5	8	1	4	4	15	16	60
10	6	12	11	3	21	16	10	7	13	12	13	124
11	0	6	0	0	3	8	5	7	10	7	5	51
Total	51	71	57	44	108	147	106	142	152	207	180	1265

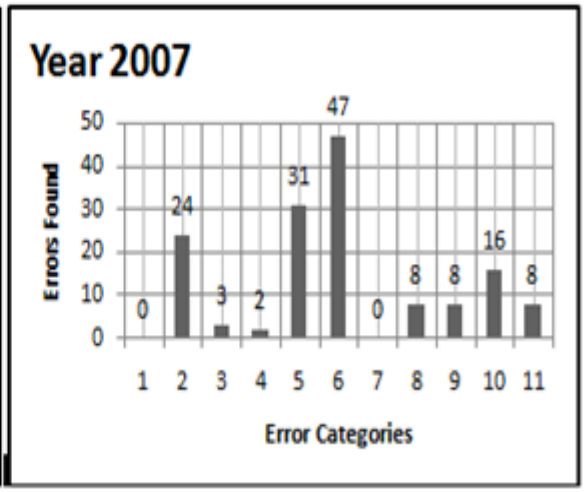












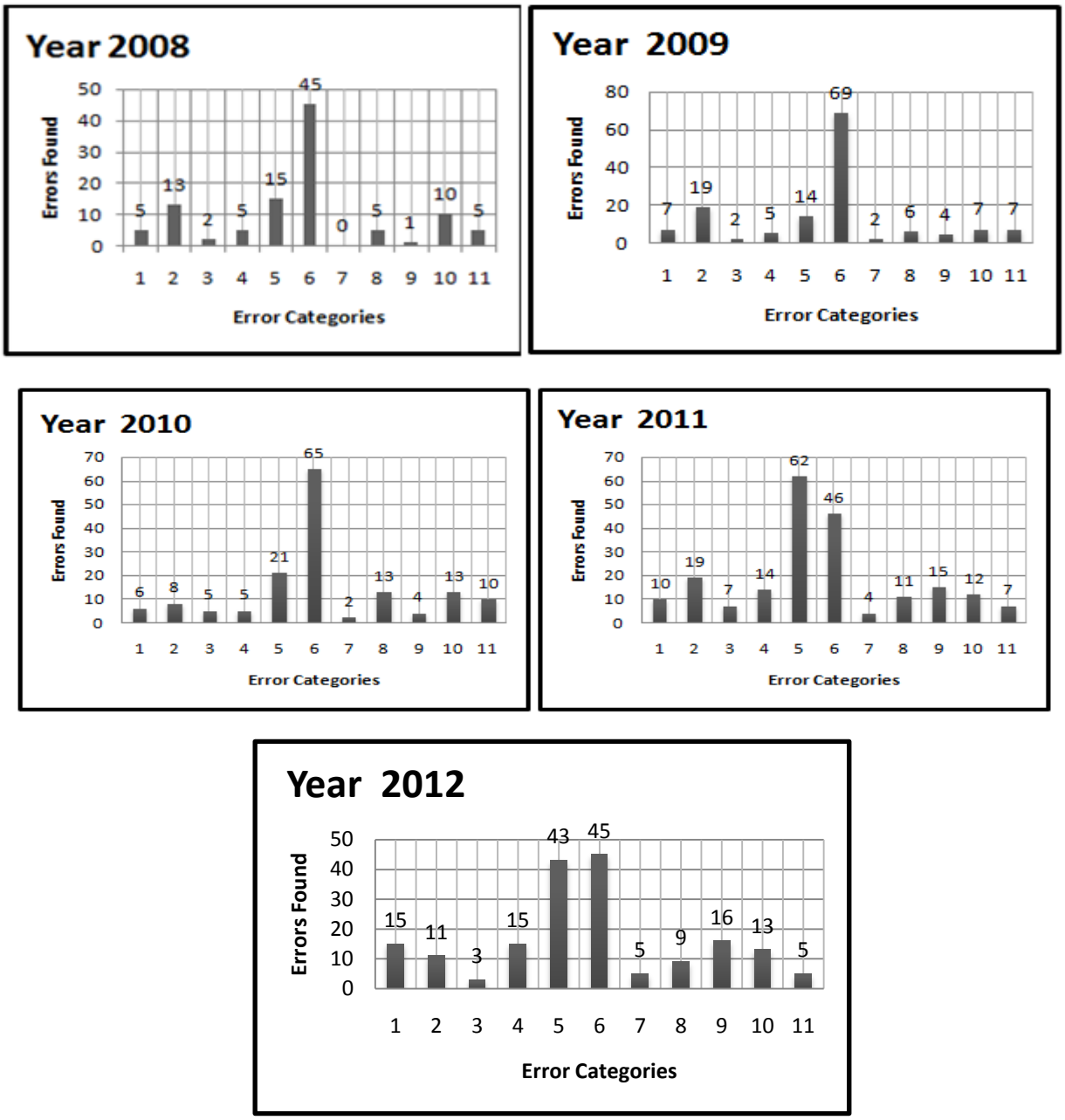


Figure 1: Error Categories versus Number of Errors Found in Respective Category

(a) Year – 2002 (b) Year – 2003 (c) Year – 2004 (d) Year – 2005 (e) Year – 2006

(f) Year – 2007 (g) Year – 2008 (h) Year – 2009 (i) Year – 2010 (j) Year – 2011

(k) Year – 2012

Figure 1 disclose that the sources of errors vary significantly for the eleven academic years from 2001-2002 to 2011-2012. No charts are alike. In Table 3, we present the top two error categories for the academic years from 2001-2002 to 2011-12 where maximum errors were found.

Table 3: Year–Wise Most Frequent and Second Most Frequent Error Category

Year	Most Frequent Error Category	Error Category No.	Second Most Frequent Error Category	Error Category No.
2002	Form Design / User Interface	5	Data Flow Diagram Database Design	2 6

2003	Form Design / User Interface Testing	5 10	Data Flow Diagram	2
2004	Database Design	6	Form Design / User Interface	5
2005	Database Design	6	Data Flow Diagram	2
2006	Form Design / User Interface	5	Database Design	6
2007	Database Design	6	Form Design / User Interface	5
2008	Database Design	6	Form Design / User Interface	5
2009	Database Design	6	Data Flow Diagram	2
2010	Database Design	6	Form Design / User Interface	5
2011	Form Design / User Interface	5	Database Design	6
2012	Database Design	6	Form Design / User Interface	5

From Table 3, as the error category number 6 is having maximum occurrence, we conclude that 'Database Design' is the error category where highest number of errors have been encountered during last eleven years.

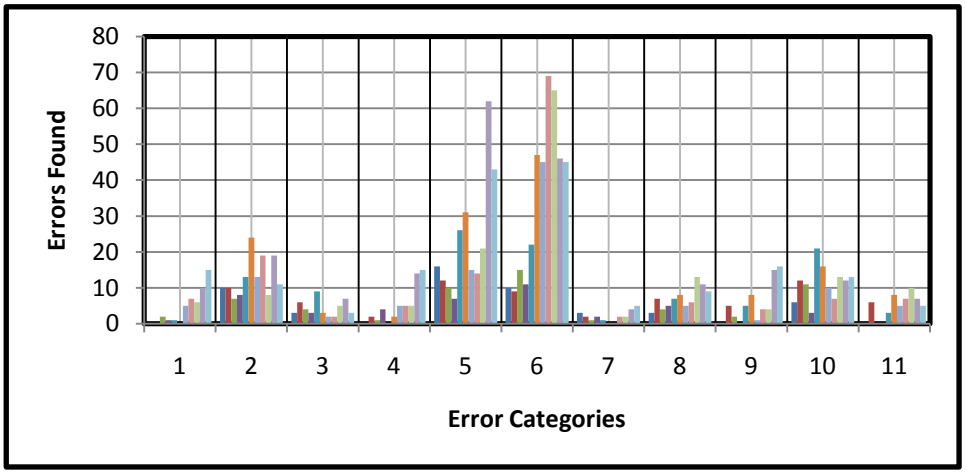


Figure 2: Error Categories and Number of Errors for Year 2002 To Year 2012

Figure 2 show Error Categories and Number of Errors from Year 2002 To Year – 2012. Error category number 6 'Database Design' shows the tallest spike followed by error category number 5 'Form Design / User Interface'.

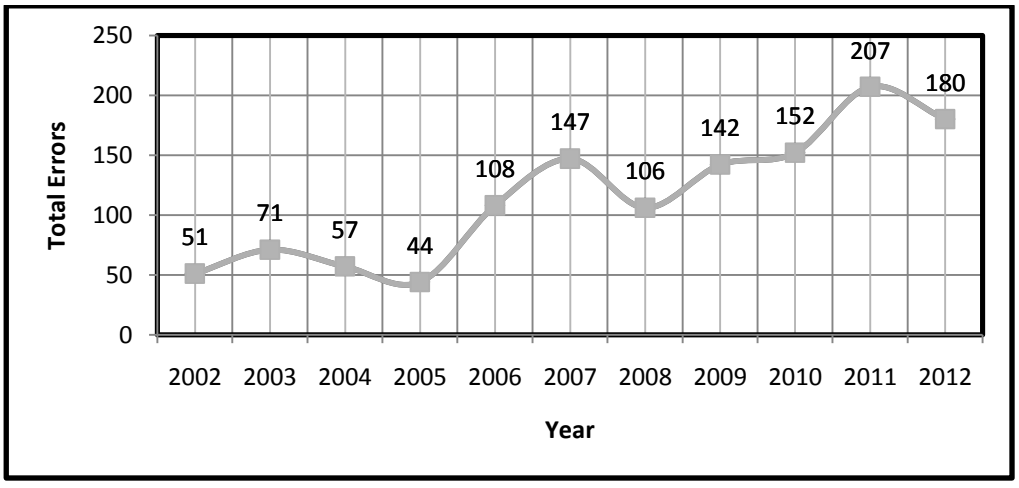


Figure 3: Errors are having an upward trend from 2002 to 2012

After identifying 103 software attributes into quantifiable and non-quantifiable attributes, only quantifiable attributes were considered for assigning weights. The results of the survey are presented in tabular format in Table 4, wherein 'SEn' indicates the values provided by n^{th} Software Engineer, with n ranging from 1 to 73.

Table 4: Survey Result

Sr. No.	Quantifiable Attributes	SE1	SE2	SE3	..SEn	Total	Average
1.	Code	30	80	100	...	4444	60.87
2.	Feasibility Study	90	40	80	...	4460	61.09
3.	Process Model	70	40	70	...	4220	57.80
4.	Project Monitoring and Control	70	40	100	...	4030	55.20
5.	Requirement Analysis	90	80	100	...	5850	80.13
6.	Structured Design Methodology	75	80	90	...	5105	69.93
7.	Unified Modelling Language	70	70	70	...	4598	62.98
8.	User Interface Design	80	100	100	...	5040	69.04
9.	Verification	90	100	100	...	5109	69.98
10.	Database Design	80	85	80	...	6138	84.08

The weights averaged based on the values provided by 73 software engineers are presented in Table 5. It is noteworthy to mention that each of the 10 quantifiable attributes were assigned weight out of 100 and it was not necessary to have the total of weights of 10 attributes as break-up of 100. In scientific research community, this practice is technically known as based on human perception and general intelligence.

Table 5: Weight Average to Quantifiable Attributes

Sr No.	Quantifiable Attribute	Average (%)
1.	Code	60.87
2.	Feasibility Study	61.09
3.	Process Model	57.80
4.	Project Monitoring and Control	55.20
5.	Requirement Analysis	80.13
6.	Structured Design Methodology	69.93
7.	Unified Modelling Language	62.98
8.	User Interface Design	69.04
9.	Verification	69.98
10.	Database Design	84.08

Based on the average values presented in Table 5, it has been found that the software engineers gave maximum weight to Database Design (84.08 %), while Requirement Analysis (80.13%) was found to have achieved the second highest weight. Similarly, the minimum weight was found to be assigned to Project Monitoring and Control (55.20%) while the second lowest weight was found to be achieved by Process Model (57.80%). Now weights are calculated for each quantifiable attributes by dividing average weight of each quantifiable attribute by total weight average as shown in Table 6.The formula for assigning score to each quantifiable attributes is presented using data set as in Table 7; same procedure is repeated for rest of other attributes.

Table 6: Weights to Quantifiable Attributes

Sr No.	Quantifiable Attribute	Average	Weight
1.	Code	60.87	0.0907
2.	Feasibility Study	61.09	0.0910
3.	Process Modl	57.80	0.0861
4.	Project Monitoring and Control	55.20	0.0822
5.	Requirement Analysis	80.13	0.1194
6.	Structured Design Methodology	69.93	0.1041
7.	Unified Modelling Language	62.98	0.0938
8.	User Interface Design	69.04	0.1028
9.	Verification	69.98	0.1042
10.	Database Design	84.08	0.1252
	Total	671.1506	1.0000

Table 7: Formula for Scoring Weight to Each Quantifiable Attributes

Sr No.	Quantifiable Attribute	Total Marks	Marks Obtained	Weight	Score (Marks Obtained * Weight)
1	Code	100	70	0.0907	6.3490
2	Feasibility Study	100	65	0.0910	5.9150
3	Requirement Analysis	100	75	0.1194	8.9550

The Software template for evaluating and scoring software project documentation is developed using Microsoft Visual Studio 2010 version 10.0 as front end and Microsoft Access 2007 as backend. The template initially prompts users to input valid username and password. User type is classified into two category (a) Administrator and (b) public. Administrator is having the authorized responsibility of (a) creating/blocking users (b) setting template data. Template having Masters Menu with following options: (a) Colleges (b) Make Result and (c) View. Also searching criteria can be used to search details for any college.

For evaluating student software project documentation "Make Result" option of Masters Menu is used as described in Figure 4. The screen is divided into two sections. In the upper section student basic details are to be filled down. The second section contains the evaluation criteria where evaluator is provided with ten quantifiable attributes which are considered as metrics for evaluating software project documentation. For scoring these attributes each attribute is provided with range marking which are assigned to each category and their sub categories. The range marking are described as Very Poor (0), Poor (20), Fair (40), Average (50), Good (60), Very Good (80), and Excellent (100).

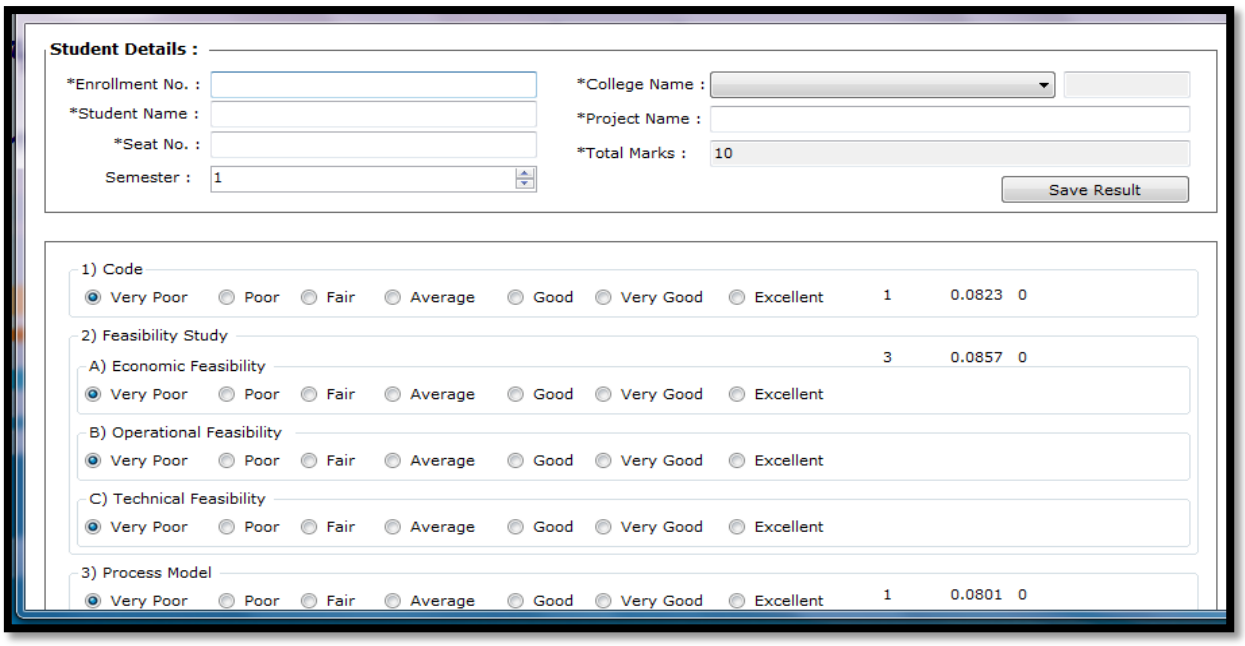


Figure 4: Evaluating and Scoring Documentation

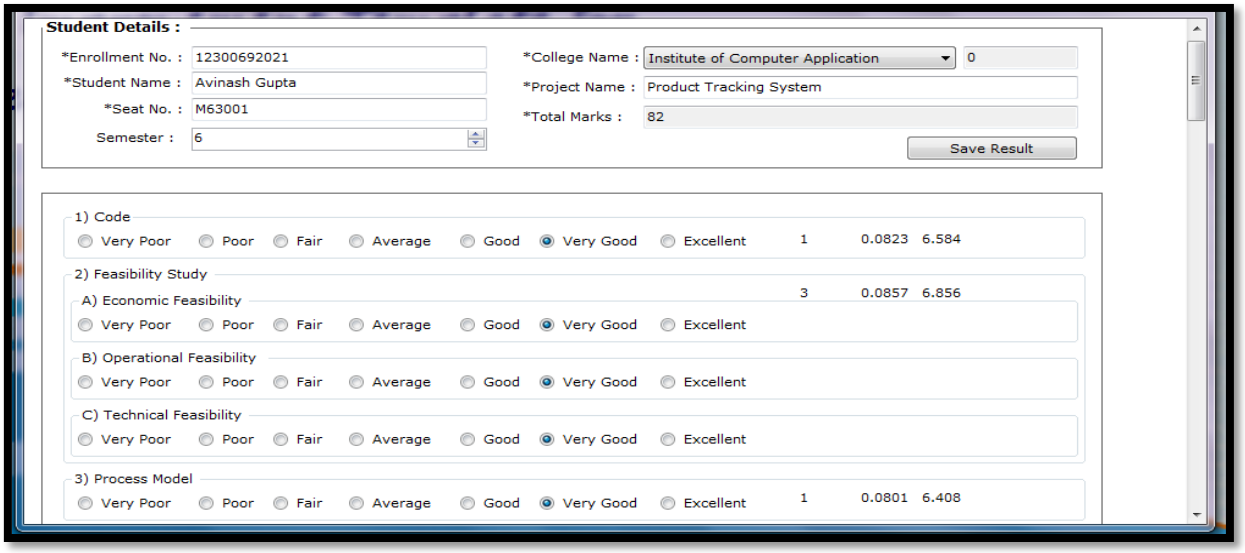
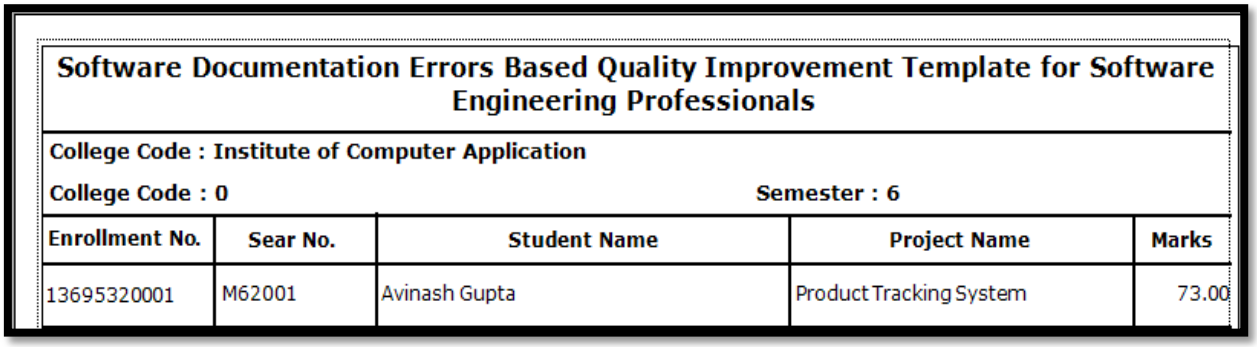


Figure 5: Scoring each attributes and displaying documentation score

Score are to be selected for each quantifiable attributes. Each score marks are being multiplied with the weight age assigned to each attributes and at last summation is done. This procedure is to be repeated for rest of the attributes and finally score of the documentation is been calculated and displayed. This template was used by examiners for examining 27 software documentations, the result of template usage by one of the examiner is shown in Figure 5.



Software Documentation Errors Based Quality Improvement Template for Software Engineering Professionals

College Code : Institute of Computer Application

College Code : 0 **Semester : 6**

Enrollment No.	Sear No.	Student Name	Project Name	Marks
13695320001	M62001	Avinash Gupta	Product Tracking System	73.00

Figure 6: Report

Evaluated and scored documentation results are stored in database which can be viewed using "View Result" of Masters Menu. Figure 6 describes the same.

5. CONCLUSION

Documentation of large software projects prepared by final year students of Master level course have considered as a basis of data collection. The duration of these software projects was six months. Errors were analysed from these software projects documentation. We found many errors and classified them into error categories via... Process Model, Data Flow Diagram, Process specification, Entity Relationship Diagram, Form Design / User Interface, Database design, Code design, Exception Handling, Reports, Testing and Documentation. We analysed number of errors in each category and carried out this procedure for the academic years 2001-2002 to 2011-2012. Based on our study for these academic years, we conclude that sources of errors vary significantly over a period of time. Among the extremely significant error categories derived from our finding and analysis, we conclude that 'Database Design' is the error category where highest number of errors have been encountered during last four years.

Form Design / User Interface is the second most notorious error category been identified from finding and analysis. We also identified 103 software attributes from software project documentations. Further these software attributes were classified 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 identifying and assigning weights to quantifiable attributes was to score software project documentation. Finally, a software template model is implemented for evaluating and scoring software project documentation. In future, we will be implementing with software template in industries as well as academic environment.

6. REFERENCES

[1] Arthur J. D. , Stevens 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.

[2] Boer R., "Writing and Reading Software Documentation: How the Development Process may Affect Understanding", proceeding of 2009 ICSE workshop on Cooperative and Human Aspects on Software Engineering.

[3] 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.

[4] Chomal V.S. , Saini J.R., " 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

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

[6] Chomal V.S. , Saini J.R., "Identification and Analysis of Causes for Software Failures‖, National Journal Of Computer Science And Technology] Volume: 04 | Issue: 02 | July – December – 2012

[7] Chomal V.S. , Saini J.R., "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

[8] Chomal V.S. , Saini J.R., "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

[9] Chomal V.S. , Saini J.R., "Significance of Software Documentation in Software Development Process" International Journal of Engineering Innovation and Research, ISSN: 2277 – 5668, Volume 3, Issue 4

[10] Chomal V.S. , Saini J.R., "Identification, Categorization and Weighting of Software Engineering Attributes for Quality Evaluation of Software Project Documentation", International Journal of Advanced Networking Applications (IJANA), ISSN No: 0975 – 0290, page – 53, 2014.

[11] Cook C. R., M. Visconti M., "New and Improved Documentation Process Model", Proceedings of the 14th Pacific Northwest Software Quality Conference, 1996.

[12] Delaney D. , Brown S. , "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

[13] Forward A. J., "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.

[14] Forward A. J., "The Relevance of Software Documentation, Tools and Technologies: A Survey", Proceeding 2002 ACM Symposium on Document Engineering.

[15] Hilburn T. B. , Towhidnejad M. , "Software Quality: A Curriculum Postscript?, Proceeding SIGCSE 2000 at thirty first SIGCSE technical symposium of computer science and education.

[16] Jazzar A., Scacchi W., "Understanding the requirements for information system documentation: an

empirical investigation", COOCS `95, Sheraton Silicon Valley, California, USA, ACM Press, p268 – 279.

[17] Kipyegen N.J, William P.K.K. , "Importance of Software Documentation", International Journal of Computer Science, Vol. 10, Issue: 5, No.1, September, 2013, ISSN: 1694 - 0784

[18] Laitinen K. , "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

[19] Nasution M.F.F , Weinstroffer H.R. "Documentation in Systems Development: A Significant Criterion for Project Success" HICSS 2009 42nd Hawaii International

[20] Scheff B. H. ,Georgon T. , "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.

[21] 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.

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