Software Template for Evaluating and Scoring Software Project Documentations

Vikas S. Chomal  
1Assistant Professor  
2Research Scholar  
1The Mandvi Education Society Institute of Computer Studies, Mandvi, Gujarat, India  
2Singhaniya University, Pacheri Bari, District – Jhunjhunu, Rajasthan

Jatinderkumar R. Saini, Ph.D.  
1Director I/C & Associate Professor  
2Research Supervisor  
1Narmada College of Computer Application, Bharuch, Gujarat, India  
2Singhaniya 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>
<thead>
<tr>
<th>Sr No.</th>
<th>Attribute</th>
<th>Sr No.</th>
<th>Attribute</th>
<th>Sr No.</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Acceptance Testing</td>
<td>36</td>
<td>Functional Requirement</td>
<td>71</td>
<td>Software Life Cycle</td>
</tr>
<tr>
<td>2</td>
<td>Activity Diagram</td>
<td>37</td>
<td>Functionality</td>
<td>72</td>
<td>Software Project</td>
</tr>
<tr>
<td>3</td>
<td>Adaptive Maintenance</td>
<td>38</td>
<td>Gantt Chart</td>
<td>73</td>
<td>Software Quality</td>
</tr>
<tr>
<td></td>
<td>Item</td>
<td></td>
<td>Item</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-------------------------------------</td>
<td>---</td>
<td>-------------------------------------</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Alpha Testing</td>
<td>39</td>
<td>Integration Testing</td>
<td>74</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Beta Testing</td>
<td>40</td>
<td>Levels of Testing</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Black Box Testing</td>
<td>41</td>
<td>Milestone</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Boundary Value Analysis</td>
<td>42</td>
<td>Non-Functional Requirement</td>
<td>77</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Branch Testing</td>
<td>43</td>
<td>Normal Requirement</td>
<td>78</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Bug</td>
<td>44</td>
<td>Object Oriented Analysis</td>
<td>79</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Class Diagram</td>
<td>45</td>
<td>Object Oriented Design</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Code</td>
<td>46</td>
<td>Operational Feasibility</td>
<td>81</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Control Flow Based Testing</td>
<td>47</td>
<td>Path Testing</td>
<td>82</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Corrective Maintenance</td>
<td>48</td>
<td>Perfective Maintenance</td>
<td>83</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Critical Path Method</td>
<td>49</td>
<td>Process model</td>
<td>84</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Data Dictionary</td>
<td>50</td>
<td>Process Specification</td>
<td>85</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Data Flow Diagram</td>
<td>51</td>
<td>Project Monitoring and Control</td>
<td>86</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Debugging</td>
<td>52</td>
<td>Project Planning</td>
<td>87</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Defect</td>
<td>53</td>
<td>Project Progress</td>
<td>89</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Defect Removal Efficiency</td>
<td>54</td>
<td>Project Scheduling</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Design</td>
<td>55</td>
<td>Project Tracking</td>
<td>91</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Design Constraints</td>
<td>56</td>
<td>Quality Function Deployment</td>
<td>92</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Document Structure</td>
<td>57</td>
<td>Regression Testing</td>
<td>93</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Economic Feasibility</td>
<td>58</td>
<td>Reliability</td>
<td>94</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Effort</td>
<td>59</td>
<td>Reports</td>
<td>95</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Entity Relationship Diagram</td>
<td>60</td>
<td>Requirement Analysis</td>
<td>96</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Equivalence Class Partitioning</td>
<td>61</td>
<td>Requirement Validation</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Errors</td>
<td>62</td>
<td>Requirements</td>
<td>98</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Excited Requirement</td>
<td>63</td>
<td>Risk Management</td>
<td>99</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Expected Requirement</td>
<td>64</td>
<td>Sequence Diagram</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>External Interface Requirement</td>
<td>65</td>
<td>Size Estimation</td>
<td>101</td>
<td></td>
</tr>
</tbody>
</table>

**International Journal of Computer Applications (0975 – 8887)**  
**Volume 116 – No. 1, April 2015**
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
c) Test Plan
dd) Testing Process
e) 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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>7</td>
<td>6</td>
<td>10</td>
<td>15</td>
<td>47</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>10</td>
<td>7</td>
<td>8</td>
<td>13</td>
<td>24</td>
<td>13</td>
<td>19</td>
<td>8</td>
<td>19</td>
<td>11</td>
<td>142</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td>9</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>3</td>
<td>47</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>14</td>
<td>15</td>
<td>53</td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td>12</td>
<td>10</td>
<td>7</td>
<td>26</td>
<td>31</td>
<td>15</td>
<td>14</td>
<td>21</td>
<td>62</td>
<td>43</td>
<td>257</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>9</td>
<td>15</td>
<td>11</td>
<td>22</td>
<td>47</td>
<td>45</td>
<td>69</td>
<td>65</td>
<td>46</td>
<td>45</td>
<td>384</td>
</tr>
</tbody>
</table>

19
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>
<thead>
<tr>
<th>Year</th>
<th>Most Frequent Error Category</th>
<th>Error Category No.</th>
<th>Second Most Frequent Error Category</th>
<th>Error Category No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2002</td>
<td>Form Design / User Interface</td>
<td>5</td>
<td>Data Flow Diagram</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Database Design</td>
<td>6</td>
</tr>
</tbody>
</table>
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 nth Software Engineer, with n ranging from 1 to 73.

**Table 4: Survey Result**

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

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**

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

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

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

Table 7: Formula for Scoring Weight to Each Quantifiable Attributes

<table>
<thead>
<tr>
<th>Sr No.</th>
<th>Quantifiable Attribute</th>
<th>Total Marks</th>
<th>Marks Obtained</th>
<th>Weight</th>
<th>Score (Marks Obtained * Weight)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Code</td>
<td>100</td>
<td>70</td>
<td>0.0907</td>
<td>6.3490</td>
</tr>
<tr>
<td>2</td>
<td>Feasibility Study</td>
<td>100</td>
<td>65</td>
<td>0.0910</td>
<td>5.9150</td>
</tr>
<tr>
<td>3</td>
<td>Requirement Analysis</td>
<td>100</td>
<td>75</td>
<td>0.1194</td>
<td>8.9550</td>
</tr>
</tbody>
</table>

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).
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.
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 documentation. 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


[16] Jazzer A., Scacchi W., “Understanding the requirements for information system documentation: an
empirical investigation”, COOCS ’95, Sheraton Silicon
Valley, California, USA, ACM Press, p268 – 279.

Software Documentation”, International Journal of
Computer Science, Vol. 10, Issue: 5, No.1, September,
2013, ISSN: 1694 - 0784

Quality Systems”, Technical Research Centre of
Finland (VTT) Computer Technology Laboratory,
ACM SIGSOFT SOFTWARE ENGINEERING
NOTES vol 17 no 4 Oct 1992 Page 32

Systems Development: A Significant Criterion for
Project Success” HICSS 2009 42nd Hawaii International

do software engineering or freeing software engineers
from the shackles of documentation”, p81 – 91,
SIGDOC ’88, Ann Arbor, Michigan, USA, ACM Press,

[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