ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645
Indian Journal of Science and Technology, *Vol 9(21), DOI: 10.17485/ijst/2016/v9i21/95147, June 2016*

An Efficient Object Oriented Design Model: By Measuring and Prioritizing the Design Metrics of UML Class Diagram with Preeminent Quality Attributes

Sudha Rajesh[1]* and A. Chandrasekar[2]

[1]Sathyabama University, OMR, Sholinganallur, Chennai – 600119, Tamil Nadu, India; sudharajesh2005@gmail.com
[2]Department of CSE, St. Joseph's College of Engineering, OMR, Sholinganallur, Chennai - 600119, Tamil Nadu, India; drchandrucse@gmail.com

Abstract

In Software Engineering, non-functional quality attributes plays a major part in the design phase. The conflicts among quality attributes will smash up the overall software quality. Due to many changes in the requirements, may affect the overall design quality. It is very important to maintain the quality in the former stages of SDLC. In order to overcome the conflicts among attributes and to design quality software, a work is proposed to create an Efficient Object Oriented Design Model (EOODM), by Measuring and Prioritizing the Quality Metrics of UML Class Diagram. The above model works in three steps as, 1) The conflicts quality attributes are managed by creating generic rules, 2) The measurement is done by quantify the Object Oriented Design metrics of a class diagram. 3) The priority will be given to the attributes by equivalence partitioning of the quality metrics. However, all the design is done by using UML diagrams, especially the Class Diagram, where they are the ruler for the developers. We are in a position to measure and estimate the software quality attributes along with the quality metrics. By measuring the Design Metrics of Class Diagram leads to Preeminent Quality Software. This model will help the designers to evaluate a better software system.The main aim is to enhance the software design by improving the design metrics of UML class diagram.

Keywords: Class Diagram, Design Metrics, Efficient Object Oriented Design Model, Quality Attributes

1. Introduction

In emerging world of Software Engineering, SDLC processes are very important in developing the software products. The developers' endeavour is to deliver the product with packed quality. *IEEE Definition* of software quality is "The degree to which a system, component, or process meets specified requirements". *Pressman's definition* of quality is "Conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software". Quality's perspective can be classified as Product-based quality, Value-based quality, User-based quality and Manufacturing -based quality shown in Figure 1. From these perspectives value-based quality is very important, since it defines the design quality of the software. To maintain the quality in former stages of SDLC is essential. Most of the developers propose the product design focusing on Functional Requirements.

Without considering the Non- Functional Requirements, there will be collapse in the overall system quality. The requirements of the product design will be listed out by the stakeholders. During the period of analyzing the requirements gathered, there will more conflicts among stakeholders' views. These conflicts may also lead to poor quality of software design.The developers use the Object Oriented Design methodologies to enlarge the

*Author for correspondence

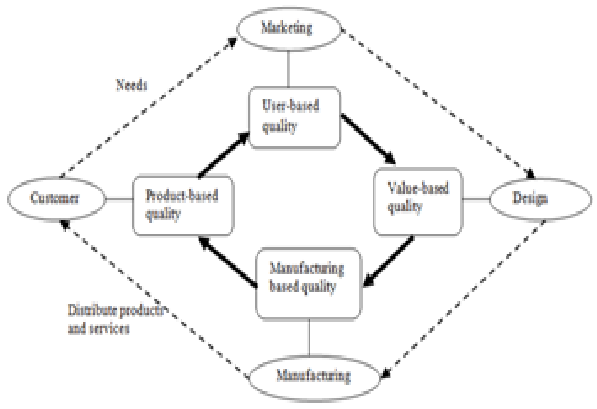

Figure 1. Quality Perspective.

software; chiefly they use the UML Class Diagram.

The metrics of class diagram may also be measured at the early stages of SDLC process. But the developers do not take care of the design part, which makes the system to be weak in quality. To avoid the above said problems with quality, can be reduced by developing an Efficient Object Oriented Design Model. During the development of the system architecture, it is important to authorize that the structural design has the necessary quality attributes; this is usually done using one or more architecture evaluations.A quality constraint is a prerequisite that is positioned on a software structure by a stakeholder; an excellence feature in the system really there on one occasion it have be employed. Quality attribute is the belongings of a software structure[1].

In [2] studied Non-Functional Attributes (NFA) and can be accomplished by applying guidelines while developing software systems. Such guidelines include best practices, design patterns and architectural styles. However, achieving multiple non-functional attributes at the same time has extra concerns. Guidelines have different effects on non-functional attributes of software, and guidelines have relationships among each other too. A guideline that has positive effect on a non-functional attribute may have negative effect on another attribute. In [3] suggests that non-functional quality is of little relevance for users and customers, and is instead primarily a concern for Software architects. The practitioners consider non -functional qualities as a late addition, slightly than as a major driver of structural design. Development teams underestimate the contribution of non-functional qualities to a system's success.According to[4] it is essential to recognizeall stakeholders, gather all requirements and make sure they know the inference of thesoftware. The space among stakeholders' idea of the anticipated software and the analysis's representationof that software is the reason of failing in analysis. If the necessities precise byanalysts can be experienced not in favor of stakeholders' hope, then this space might be pointed, and improved solutions might be given. They used ReqVerifier tool to check software requirements andprove them next to stakeholders' vision in order to enlarge a good software requirement for quality software. In[5] explained that the condition management to be the procedure of documenting, tracing, prioritizing and agreeing on requirements and then calculating modification and communicate to relevant stakeholders.

It is very important to analysis the requirement in the early stages of SDLC process, because they are useful to avoid the stakeholders' conflicts and satisfy the future expectations of them6. In7 projected 6 metrics-Weighted Methods per Class (WMC), Response sets for Class (RFC), Lack of Cohesion in methods (LCOM), Coupling Between Object Classes (CBO), Depth of Inheritance Tree (DIT), Number of Children of a class (NOC), with the help of various software quality attributes (e.g. efficiency, complexity, understandability, reusability, maintainability and testability) can be calculated. In8 referred a lot of software quality estimation techniques to construct software quality model andas well evaluatethe presentation of these technique. Some techniquesare Artificial Neural Network, Case-Base Rule, Regression Tree,Rule Based System, Multiple Linear Regression and Fuzzy System etc. Their outcomes expose that Fuzzy and Rule Based System techniquescan give a high-quality explanation to design a Software Quality Model. In9 acknowledged that choosing a SDLC models are very important, if the models are selected in a wrong way, it will lead the software development in some critical situations. They proposed a usable variety matrix fordecide best-fit SDLC models on special style ofSoftware Development Projects, wrap mutuallyconventional and lively methodologies. In10 proposed the object-oriented frameworks, which enables the reusable of the software. But reusing is not straightforward; most of them are reused in fewer quantities. They also suggests that reuse of software is the main objectives of Software Engineering. In11 suggested that complexity metrics of the class diagram is collection of three varieties of relationships: association, generalization and aggregation,which construct their overall structure. They suggested merging these three associations thatallow a complete and suitable determination of difficulty.

In12 suggested the basic software characteristics such as maintainability, portability, complexity, testability, reusability, and understandability can be measured using the design metrics. These characteristics help the designers to improve the quality of the system in better way. In13 developed a Generic Model which helps the developers to predict the quality in software system. They also calculated the quality factors with accuracy. The predictions are done by using the quality factors, Software development process and SDLC process. In14 provided a set of metrics to characterizes large object-oriented software systems. Their metrics distinguish the quality with respect to APIs of the modules. The intermodule dependencies are origin to inheritance, associational relationships, state access violations, fragile base-class design. In[15] developed a mathematical formalism which gives suitableconsideration to the things of the object-oriented system. Also shows that recent study in this region includes unstable treatment ofspecial yield and their possessions at dissimilar progress phase. Moreover outline thesequential progress of study in this region, and discover space that proposeschance for upcoming study.

1.1 Research Questions

The primary goal of the proposed work is to develop an Efficient Object Oriented Design Model - by Measuring and Prioritizing the Design Metrics of UML Class Diagram with Preeminent Quality Attributes. As a result, the following questions are to be answered:

1. How the conflicts of stakeholders are managed according to the system development?
2. What are the design metrics of a class diagram?
3. How the design metrics are to be measured?
4. How the design metrics are to be prioritized?
5. Will the metrics richness and rightnessdistinguish between the design output in former and later stages of Object Oriented Design?

1.2 Contributions

The major involvement of this work is to develop a model that should satisfy the above questions as follows:

1. To collect all the stakeholders' decisions in a single structural view that expose the centric-view decision in an architectural design.
2. The conflicts in stakeholders' views are managed by analyzing it, with finest consistency rules.
3. The design metrics are measured by the relationships and associations between them.
4. The metrics are prioritized based upon the quality attributes' contribution to develop the software.
5. It is possible to get quality software, if all the design metrics are measured in the former stages of SDLC process rather than in later stages.

2. Motivation behind the Proposed Work.

Following are the drawbacks identified: 1) There are many conflicts among stakeholders' views. Therefore it affects the overall systems' quality. 2) Developers are concentrating only on the functional attributes not on the non - functional attributes. It also affects the systems' quality. 3) Moreover the designs of a software system are done by using the Object Oriented Design Metrics and these metrics are not properly measured for the evaluation. 4) The UML Class Diagram is used for designing the software system, which also makes the design part to complex. Since there are many classes, it is difficult to identify the associations and relationships among classes. An Efficient Object Oriented Design Model (EOODM) is proposed to conquer the above mentioned drawbacks. This model first collects the stakeholders' views (attributes needed to develop software) and manages the conflicts by creating consistency rules. Then the design metrics for class diagram is measured by quality factors. Since Object Oriented Design Metrics plays an important role in designing phase of SDLC, some of the metrics are measured here. At last the priority will be given to the quality attributes based on their evaluation. Marcela Genero, Luis Jiménez, Mario Piattini[16], OO model, like class diagrams are the input object in the early hours of improvement, so centre of attention in their excellence must give to the quality of the OOIS which is eventually executed. William A.Ward, Jr.[17], Anas Bassam AL-Badareen, Mohd Hasan Selamat, Marzanah A. Jabar, Jamilah Din, and Sherzod Turaev[18], suggested various models to improve the quality of software.ShilpeeChamoli, Gil Tenne and Sanjay Bhatia[19], suggest an idea to identify the software metrics for various application, then analyzing which metrics affects the accurateness of software. This defect will make the software not to be in quality. In[20] suggested a method to evaluate the software projects and give them a rating, to identify the quality of each and every stages of the software development phases. The quality may vary due to many changes in the development process. In[21] proposed alatest algorithm with the aim of a designerto change a structural design to a complete model basis on dissimilar expansion of Petri Nets.

3. Proposed Model - Efficient Object Oriented Design Model (EOODM).

The following EOODM says that stakeholders' views are collected, managed, design metrics are specified, measured, given priority and validated. A sample voting system application has taken to implement the following steps.

3.1 Evolution of EOODM

Step1: Gathering the stakeholders' views for Voting System.

The following Table1 shows the various stakeholders' views for developing a voting system22. These views will create a problem during the design phase, because the developer will develop a model according to the views without the knowledge of different view in electoral dispute resolution bodies. Since all the views cannot be taken into account for designing practice, there will be some conflicts.

Step 2:Manage the conflicts in stakeholders' views

These conflicts can be managed by using the following rules[23], *Rule1:*Identifying stakeholders, *Rule2:*Recognize and Classify stakeholders'view,*Rule3:* Institutional Review, *Rule4*: Executive Development, *Rule5*:Conflict Analysis. By consideringthe above steps, the conflicts among stakeholders can beavoided. Once the stakeholders' conflicts are identified, their views are analyzed to manage the conflicts between them; they depend upon an approach for successful conflict management maximizes the integrative function of the two parties in conflict[20],

$$Max\, y = a + b_1X_1 + b_2X_2 = b_3X_3 + \ldots + b_nX_n + b_{n+1}X_1X_2 + b_{n+2}X_1X_3 + \ldots$$

Where x1 – utility to disputant 1 , x2 – utility to disputant 2 , x3 . . . xn – utility to third parties affected by the dispute between 1 and 2

Step 3:Specify the Design according to stakeholders' views (UML Class Diagram)

Step 4: Get the relationships between classes

The following Table2 shows the relationships among the classes from above figure 3. Among 18 classes, there were 17 relations which notify that the design was more complexity. If any one of the relationship of attributes or methods of classes are tried to modify, then it will affect in most of the places

Step 5: Specify Quality attributes with Design metrics

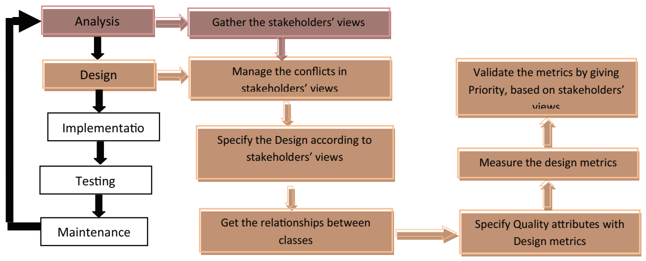
Analysis
Design
Implementatio
Testing
Maintenance
Gather the stakeholders' views
Manage the conflicts in stakeholders' views
Specify the Design according to stakeholders' views
Get the relationships between classes
Specify Quality attributes with Design metrics
Measure the design metrics
Validate the metrics by giving Priority, based on stakeholders' views

Figure 2. Efficient Object Oriented Design Model (EOODM)

Table 1. Stakeholders and their Views for Voting System

Stakeholders	Political Parties and Candidates	Election Board Staff	Administrative Branch of Government	Elected Government (Legislature)	Electoral argument -declaration Bodies	Official System
Stakeholders' views	Assurance	Safety Measures	Responsibility	Performance	Security	Accessible

Election Supervisor and Observers		The Media	The Voters	Social Society Association
Domestic	Domestic			
Effective		Truthful	Regular comm... & Responds (accurately)	Regular consultation with these stakeholders

Table 2. Relationships among classes

Classes/ Relationships	C1	C2	C3	C4	C5	C6	C7	C8	C9	C 10	C 11	C 12	C 13	C 14	C 15	C 16	C 17	C 18
C1	-	Gen	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
C2	-	-	-	-	Gen	-	-	-	-	-	-	-	-	-	-	-	-	-
C3	-	Gen	-	-	Gen	Dep	-	-	-	-	-	-	-	-	-	-	-	-
C4	-	-	-	-	Dep	-	-	-	-	-	-	-	-	-	-	-	-	-
C5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
C6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	Asso
C7	-	-	-	-	Agg	-	-	Gen	-	-	-	-	-	-	-	-	-	-
C8	-	-	-	-	-	Comp	-	-	-	-	-	-	-	-	-	-	-	-
C9	-	-	-	-	-	-	Comp	-	-	-	-	-	-	-	-	-	-	-
C10	-	-	-	-	-	-	-	-	Gen	-	-	-	-	-	-	-	-	-
C11	-	-	-	-	-	-	-	-	Gen	-	-	-	-	-	-	-	-	-
C12	-	-	-	-	-	-	Gen	-	-	-	-	-		-	-	-	-	-
C13	-	-	-	-	-	-	Gen	-	-	-	-	-	-	-	-	-	-	-
C14	-	-	-	-	-	-	-	-	-	Gen	-	-	-	-	-	-	-	-
C15	-	-	-	-	-	-	-	-	-	Gen	-	-	-	-	-	-	-	-
C16	-	-	-	-	-	-	-	-	-	-	Gen	-	-	-	-	-	-	-
C17	-	-	-	-	-	Comp	-	-	-	-	-	-	-	-	-	-	-	Agg
C18	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Asso → Association ,Gen → Generalization, Agg → Aggregation, Comp → Composition, Dep → Dependency																		

There are many quality attributes existing to define the design of a system. But the most important quality attribute for object oriented design is modifiability metrics of class diagram[24]. If there is one modification it replicate in many places. The modifiability metrics for voting system (Figure 2) and the weighted value for the metrics are shown in Table3.

Step 6: Measure the design metrics

The average modifiability of a system is calculated as

$$AM(S) = \frac{\sum_{i=1}^{n} M(c_i)}{n} \quad \text{(eq1)}$$

Modifiability of a class c:

M(c) = C(c) + MG(c) + MA(c) + MC(c) + MD(c) + MCAss(c) + MAssC(c) (eq 2)

Complexity of a class c:

C(c) = total no. of attributes + total no. of attributes operations in a given class c (eq3)

Modifiability Generalization of a class c:

The generalization of a class in voting system is defined as the product of generalization weight value and the summation of complexity of all sub classes as follows,

$$MG(c) = \frac{W_G C\left[ASub_G(c)\right]}{2} \quad (eq\ 4)$$

Modifiability **A**ggregation of a class c MA(c):

The aggregation of a class in voting system is defined as the product of aggregation weight value and sum of complexity of immediate super class of aggregation with the complexity of all sub classes of generalization in immediate super class of aggregation of a class is as follows,

$$MA(c) = \frac{W_A\left[C(ASub_G(ISup_A(c)) + C(Isup_A(c)))\right]}{2} \quad (eq5)$$

Table 3. Modifiability Metrics

Metrics	Weight values for metrics	Definition
C(c)	-	Complexity of a class c
M(c)	-	Modifiability of a class c
AM(S)	-	Average Modifiability of a system
MG(c)	4	Modifiability Generalization of a class c
MA(c)	5	Modifiability Aggregation of a class c
MC(c)	6	Modifiability Composition of a class c
MD(c)	3	Modifiability Dependency of a class c
MCAss(c)	2	Modifiability related to Common Association
MAssC(c)	1	Modifiability related to Association Class
ASup(c)	-	Total(All) no. of super classes for a given class c
ASub (c)	-	Total(All) no. of sub classes for a given class c
ISup(c)	-	Immediate super class for a given class c
ISub(c)	-	Immediate sub class for a given class c
n	-	Total no. of classes

Modifiability Composition of a class c MC(c):
The composition of a class in voting system is defined as the product of composition weight value and sum of complexity of immediate super class of composition with the complexity of all sub classes of generalization in immediate super class of composition of a class is as follows,

$$MC(c) = \frac{W_C\left[C(ASub_G\left(ISup_C(c)\right) + C(Isup_C(c))\right]}{2} \qquad (eq6)$$

Modifiability Dependency of a class c MD(c):

The dependency of a class in voting system is defined as the product of dependency weight value and sum of complexity of immediate sub class of dependency with the complexity of all sub classes of generalization in immediate sub class of dependency of a class is as follows,

$$MD(c) = \frac{W_D\left[C(ASub_G\left(ISub_D(c)\right) + C(Isub_D(c))\right]}{2} \qquad (eq7)$$

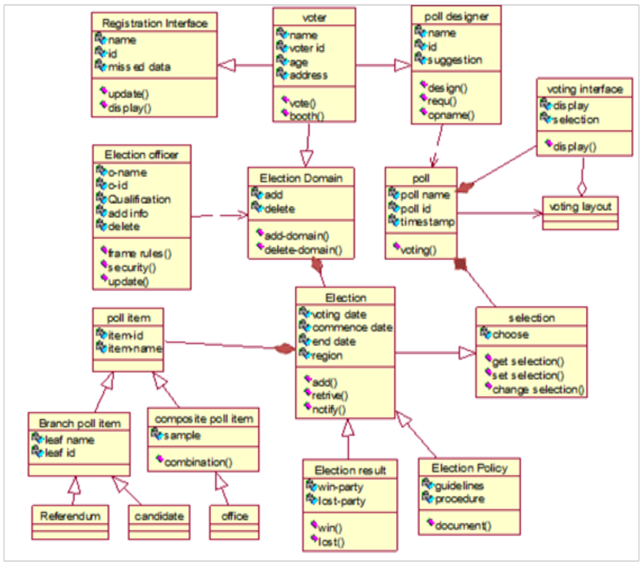

Figure 3. Design for Voting System according to the stakeholders' views.

Modifiability related to Common Association MCAss(c):

The common association of a class in voting system is defined as the product of common association weight value and sum of complexity of immediate sub class of common association with the complexity of all sub classes of generalization in immediate sub class of common association of a class is as follows,

$$MCAss(c) = \frac{W_{CAss}\left[C(ASub_G\left(ISub_{CAss}(c)\right) + C(Isub_{CAss}(c))\right]}{2} \qquad (eq8)$$

Modifiability related to Association Class MAssC(c):

The association of a class in voting system is defined as the product of association weight value and sum of complexity of immediate sub class of association with the complexity of all sub classes of generalization in immediate sub class of association of a class is as follows,

$$MAssC(c) = \frac{W_{AssC}\left[C(ASub_G\left(ISub_{AssC}(c)\right) + C(Isub_{AssC}(c))\right]}{2} \qquad (eq9)$$

Step 7: Validate the metrics by giving Priority, based on stakeholders' views

There are lots of quality attributes with metrics, but modifiability attribute with design metrics is precise, which is very significant during the designing phase of Object Oriented System Development process. The modifiability of voting system is measured with various metrics of class diagram is revealed in later section.

Table 4. Entire Super and Sub Classes for Voting System

CLASSES	CLASS NAME	ISup(c)	ISub(c)	ASup(c)	ASub (c)
C1	Reg. Interface	-	C2	-	C2
C2	Voter	C1,C3,C5	-	C1,C3,C5	-
C3	Poll Designer	-	C2	-	C2
C4	Election Officer	-	-	-	-
C5	Election Domain	-	C2	-	C2
C6	Poll	-	-	-	-
C7	Election	C8	C12,C13	C8	C12,C13
C8	Selection	-	C7	-	C7,C12,C13
C9	Poll item	-	C10,C11	-	C10,C11,C14,C15,C16
C10	Branch Poll Item	C9	C14,C15	C9	C14,c15
C11	Composite Poll Item	C9	C16	C9	C16
C12	Election Result	C7	-	C7,C8	-
C13	Election Policy	C7	-	C7,C8	-
C14	Referendum	C10	-	C9,C10	-
C15	Candidate	C10	-	C9,C10	-
C16	Office	C11	-	C9,C11	-
C17	Voting Interface	-	-	-	-
C18	Voting Layout	-	-	-	-

Table 5. Design Metrics of Voting System

COMPLEXITY																		
C(C)	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18
	5	6	6	8	4	4	7	4	2	2	2	4	3	0	0	0	3	0
GENERALIZATION																		
MG(C)	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18
	12	0	12	0	12	0	14	28	8	0	0	0	0	0	0	0	0	0
AGGREGATION																		
MA(C)	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7.5	0
COMPOSITION																		
MC(C)	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18
	0	0	0	0	0	0	30	12	41.5	0	0	0	0	0	0	0	12	0
DEPENDENCY																		
MD(C)	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18
	0	0	0	0	12	18	0	0	0	0	0	0	0	0	0	0	0	0
COMMON ASSOCIATION																		
MCAss(C)	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
ASSOCIATION CLASSES																		
MAssC(C)	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MODIFIABILITY OF CLASS																		
M(C)	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18
	12	0	12	0	24	18	44	40	49.5	0	0	0	0	0	0	0	19.5	4

Table 6. Threshold Value for AM(S)

Range of AM(S)	Modifiability level
AM(S) < 50	Easy to modify
> 50 AM(S) < 75	Moderate to modify
AM(S) > 75	Difficult to modify

4. Results

The entire super and sub classes are measured for voting system[25]. Then the complexity and various relationships among classes are considered in following Tables4 and Table5 respectively,

From the above results the Average Modifiability AM(S) (eq1) of Voting System is calculated as 12.39 %.The threshold value for AM(S) is set as Easy, Moderate & Difficult to modify, and the range for AM(S) is shown in Table 6. This establishes that if the stakeholders' views are analysed and the conflicts are managed, then a good quality software system can be developed. Even though the stakeholders' views are not modified after the designing phase, we were getting the Average Modifiability as 12.39%. It shows the attributes can be changed 50%, if it crosses the range, it will difficult to modify the design.

5. Conclusion

Software quality is the key element of software development life cycle. The quality can be maintained by the notification of the changes due to stakeholders' concerns. So a model is proposed to measure the metrics of changes in terms of modifiability.The proposed work is used to calculate the average modifiability of system and threshold value is defined as [AM(s) <50] AM(s)=12.39, so it is easy to do modification. Since all the stakeholders' views are considered and managed the conflicts the AM(s) is very less. If the modifications are less in the designing, it will help the designer to evaluate the preeminent software system.In future this average modification may be reduced by minimizing inconsistency among classes and their metrics. This work also brings out that,if any modification is done in any class it will affect the classes related to it.

6. References

1. Malik H, Hneif M.Guideline-based approach for improving the achievement of non-functional attributes of software. University of Malaya, Kuala Lumpur, 2010 Nov; 1–134.
2. Hneif M, Lee SP. Using Guidelines to Improve Quality in Software Nonfunctional Attributes.University of Malaya. IEEE Software | Published by The IEEE Computer Society. 2011 Nov.-Dec; 28(6):72 –7
3. Ameller D, Claudia P, Buschmann F, Ayala CJ, Franch X. Architecture Quality Revisited. IEEE Software published by the IEEE computer society. 2012 Jul-Aug; 29(4):22–4.
4. Zuhoor A, Salim A-K. Proposing a Systematic Approach to Verify Software Requirements. Journal of Software Engineering and Applications. 2014 Apr; 218–24.
5. Greene J, Stellman A,O'REILLY, Sebastopol. Applied Software Project Management. 2005.
6. Abran A, Moore J, John Wiley Sons, Hoboken. Chapter 2: Software Requirements. 2005.
7. Chidamber SR, Kemerer CF. A Metrics Suite for Object Oriented Design. IEEE Transactions on Software Engineering. 1994; 20(6):476–93.
8. Gupta D, Goyal VK, Mittal H. Comparative Study of Soft Computing Techniques for Software Quality Model. International Journal of Software Engineering Research and Practices. 2011 Jan; 1(1):1–5.
9. Khan PM, Sufyan Beg MM. Extended Decision Support Matrix for Selection of SDLC-Models on Traditional and Agile Software Development Projects. International Journal of Software Engineering Research and Practices. 2013Apr; 3(1):8–15.
10. Bosch J, Molin P, Mattsson M, PerOl of Bengtsson. Object-Oriented Frameworks - Problems & Experiences. Department of Computer Science and Business Administration. Printed by PsilanderGrafiska, Karlskrona. 1997; 1–20.
11. Sheldon FT, Daley KM. Measuring the Complexity of Class Diagrams in Reverse Engineering. Journal of Software Maintenance and Evolution, Research and Practice. 2005 Jul; 1–14.
12. Gaur A, Punde A. Software metrics evaluation using various lines of code and function point metrics. International Journal of Engineering Sciences and Research Technology. 2012 Oct.
13. Rana ZA, Shamail S, Awais MM. Towards a Generic Model for Software Quality Prediction. WoSQ'08. May 10 2008. Germany. 2008 May; 35–40 .
14. Sarkar S, Avinash CKAK, Rama GM. Metrics for Measuring the Quality of Modularization of Large-Scale Object-Oriented Software. IEEE Transactions on Software Engineering. 2008 Sep/Oct; 34(5).
15. Purao S, Vaishnavi V. Product Metrics for Object-Oriented Systems. ACM Computing Surveys. 2003 Jun; 35(2):191–221.
16. Genero M, Jimenez L, Piattini M. A Prediction Model for OO Information System Quality Based on Early Indicators. University of Castilla-La Mancha. 1–14.

17. Ward WA, Venkataraman B Jr. Some Observations on Software Quality. School of Computer and Information Sciences. University of South Alabama. 1999.
18. AL-Badareen AB, Selamat MH, AJabar M,Jamilah Din,SherzodTuraev , Software Quality Models: A Comparative Study. University Putra Malaysia. Springer-Verlag Berlin Hei delberg 2011Jun; 46–55 .
19. Chamoli S, Tenne G, Bhatia S. Analysing Software Metrics for Accurate Dynamic Defect Prediction Models. Indian Journal of Science and Technology. 2015 Feb; 8(S4):96–100.
20. Rashid E, Patnayak S, Bhattacherjee V. Estimation and Evaluation of Change in Software Quality at a Particular Stage of Software Development. Indian Journal of Science and Technology. 2013Oct; 6(10):1–10.
21. Emadi S, Shams F. A new executable model for software architecture based on Petri Net. Indian Journal of Science and Technology. 2009 Sep; 2(9).
22. International IDEA Handbook on Electoral Management Design-Stakeholder Relationships.2006; 201–21.
23. Moura H, Teixeira JC. Managing Stakeholders Conflicts. 2010; 286–316.
24. Bachmann F, Bass L, Nord R. Modifiability Tactics. Technical Report -Software Architecture Technology Initiative. 2007 Sep; 1–63.
25. Sudha Rajesh, Chandrasekar A. Software Quality Design Model: By Measuring the Modifiability Metrics of UML Class Diagram. TEQIP II sponsored International Conference on Computational Intelligence. Anna University. Triruchirapalli. 2015 Apr. p. 547–51.