Chapter -1
Introduction
CHAPTER –I
INTRODUCTION

Engineers modify and improve the object oriented software with the help of class diagrams; therefore, Class diagram is the main part in the design level of object-oriented software (Dallal, 2013). Class maintainability is the process of effortlessly changing a class. Previously discharging an object-oriented software framework, it was difficult to recognize the reason, time, cause and place of a class to bring some changes effortlessly (Kumar and Rath, 2016). During the phase, the probability is assessed utilizing the internal quality characteristics of a class; it incorporates cohesion, coupling, and size. For decreasing upcoming class maintenance endeavors price, engineers are urged to precisely investigate to achieve low maintainability classes by previously discharging the object-oriented framework. It was observationally considered that the correlation among internal class quality characteristics (size, cohesion, and coupling) as well as external quality characteristics (class maintainability). Utilizing statistical methods, models were developed depending upon selected interior characteristics of software to calculate class maintainability was observed (Dallal, 2013). Huge Software frameworks advance constantly to meet regularly changing demand for the user requirements. Change might be driven by business sector force, change to enactment or change requirements. Maintaining traceability joins amid consequent issues of a software framework is imperative to assess version deltas, to highlight effort/code delta irregularities and to survey the modification olden times. This can help arranging the upcoming phases of development and in assessing the dependability and price of modifications then the previous one.

Artificial Intelligence techniques are related to predict maintainability on two case studies, for example, Quality Evaluation System (QUES) and User Interface System (UIMS). This thesis also concentrates on the effectiveness of feature reduction techniques. For example, principal component analysis (PCA) is used for predicting maintainability (Kumar and Rath, 2014).
Clustering method to layout the object-oriented systems with help of K-means clustering procedures for maintenance of software is proposed (Data Mining) (Malviya et.al, 2012).

1.1 MOTIVATION

The evolution of huge software framework is a time and source-consuming action. Even by the growing computerization of software evolution action, assets are rare. So, necessity is to deliver precise data and rules to administrators, to assist them creating choices, design and plan actions and assign properties for the various software actions that proceed place throughout software evolution. Software metrics are, therefore essential to recognize anywhere the assets are required, it is essential for the conclusion. Maintainability of huge frameworks is an instance of a source- and laborious action. Relating identical challenging and confirmation determination to entirely quantities of a software framework has become cost-expensive. So, a slight change is required to classify change-proneness classes so that analysis/certification efforts can be focused on these classes. The obtainability of suitable product design metrics for characterization error-prone classes is, therefore, dynamic. Artifact-metrics are suggested and validated, for example, the quantity of lines of code, McCabe complexity metric, etc. The Point is that numerous companies have constructed their particular price, quality, and source forecast models constructed on artifact-metrics. TRW, the Software engineering Laboratory (SEL) and Helwett Packert are instances of software administrations that have been using artifact-metrics to construct their price, source, deficiency and efficiency models.

1.2 RESEARCH OBJECTIVES

1. To study data mining technique.
2. To analyze the algorithm, which is able to decide the cluster with goodness of fit among clusters using Chi-Square Test or others.
3. To propose clustering methodology to provide a general but illuminating view of a software system that may lead engineers to useful conclusions concerning its maintainability.
4. To extract useful information for the maintenance engineers through the clustering analysis and Classification Techniques.
5. To evaluate the maintainability by developed framework and validate it.

6. To develop a methodology based on the K-Means Clustering data mining Technique which has been implemented on UIMS and QUES class’s data with a set of selected metrics.

1.3 SUMMARY OF CONTRIBUTIONS

- The correlation among object-oriented metrics and maintainability were considered.

- After this, examined these five open source software written in Java language. The source code of the open source software is accessible at http://sourceforge.net. (Sourceforge Media, La Jolla, CA) (Malhotra and Jain, 2012).

- It was analyzed that the modification in software is categorized through modifications in particular classes of the specific software. Both the versions of specific software were examined. Changes in the current class’s were observed. Both (preceding, current) versions of every software were examined (Malhotra and Jain, 2011).

- The software which were considered are Database, Format, and Protocols, Game and Entertainment, Multimedia, Science and Engineering. It indicates the version, its issue date, Source Line of Code and quantity of classes for every of the software studied in this research (Malhotra and Jain, 2011).

- It was discovered that with the help of LOC tool LOC metric can be found and SCI understand tool from the sites https://scitools.com, with the help of these tools (Scientific Toolworks,Inc.St.George). It was discovered that the lines of code further generate SCI report.

- Through these tools, it was found out how many classes were added, deleted and changed. It was also discovered that how many changes were done in version 1 and version 2 as per the requirement of a user. It was observed that the capacities of the metrics were measured both independently as well as jointly, to appraise the class maintainability. Statistically made forecast models are built as well as approved.
The aim of this thesis is to study about classification techniques on huge datasets with the help of MATLAB and WEKA tool.

The verification of prototypes is evaluated by 10-fold cross-validation. During the examination of it, Chidamber, Kemerer metrics suite, in addition to it LOC metric is used for the development of a model by utilizing Naive Bayes classification methods.

The Open Source Java data with a total instance of 3518 data sets plus 8 characteristics (7 for input and 1 for output) are utilized to examined. The presentation of prototypes is assessed utilizing dual improved performance factors like accuracy plus F-Measure.

On the other hand, research aims at the uses of object-oriented metrics for forecasting change-proneness. Practically various software metrics are used as feedback for estimating change-proneness.

1.4 OUTLINE OF THE THESIS

In Chapter-3, properties of a technical estimation of software maintainability prediction and metrics. UML Class diagram’s attributes are professionally as well as precisely indicate the difficulty of Object-Oriented Software. This chapter has exposed that the method is related to person’s knowledge as well as can be beneficial to enhance software attribute (Mathur and Kaushik, 2016).

In Chapter-4, the forty programs on Visual Studio 2012 are compiled and are analyzed in code metrics. After the analysis of code metrics parameters like Maintainability Index, Depth in Inheritance, Line of Code, class coupling and cyclomatic complexity consequences are found. Prevailing method for maintainability are code metrics and the correlation between maintainability index, cyclomatic complexity, Depth in Inheritance, class coupling, Line of Code in the experiments. To reiterate, adhere to remark the source codes yet don't put a lot of trust in remarks to enhance maintainability.

In Chapter-5, the correlation between object-oriented metrics and maintainability were considered. Examination of five open source software transcribed in java was performed. The source code of open source software is
accessible at http://sourceforge.net (Sourceforge Media, La Jolla, CA). It had been analyzed later that the modification in software is measured through modifications in the classes of the specific software. Both versions of a specific software have been used. Changes in both classes are observed. Later on both (Previous, Current) versions of every software have been examined. The Database, Format, and Protocols, Game and Entertainment, Multimedia, Science and Engineering open source software are considered. It demonstrates the version, its issue date, Source Line of Code and quantity of classes for both the software examined in this research (Malhotra and Jain, 2012).

It was found that LOC metric with the help of LOC tool and SCI understand tool from the sites https://scitools.com, with the help of these tools it was discovered that the lines of codes were able to generate SCI report (Scientific Toolworks, Inc.St.George). Through these tools, It was retrieved that how many classes were added, deleted and changed. We also found how many changes done in version 1 and version 2 as per the requirement of a user. Statistically based prediction models are built as well as approved. The aim of the exploration is to research by taking a huge data utilizing MATLAB plus WEKA tool utilizing its Naive Bayes Classifier. The enactment of models is evaluated by 10-fold cross-validation. During the examination, Chidamber, Kemerer metrics suite, plus LOC metric is measured to formulate a model with the help of Naive Bayes classification methods. The open source Java data along with a complete number of 3518 data sets plus 8 attributes (7 for input and 1 for output) are utilized for investigation. Presentation of models is assessed with the help of various evaluation criteria like accuracy along with F-Measure. On the other hand, research aims towards the use of object-oriented metrics for estimating change-proneness. The assessment parameters depend on the results acquired from Kappa Statistic, Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and Relative Absolute Error. The outcomes show improved accuracy is attained through Naive Bayes Classifier estimation technique. Naïve Bayes classifiers consist of a total number of 3518 and 8 Attributes, such as WMC, DIT, NOC, CBO, RFC, LCOM, LOC, Change. Practically various software metrics are utilized as feedback for estimation of change-proneness.
Chapter-I: Introduction

In Chapter-5, Software Change and Prediction in Object-Oriented Systems utilising Statistical and Machine Learning Algorithms. During the software development process, software maintenance is an immature period throughout the life cycle. In the software development process, the maintenance process works smoothly in object-oriented software. When software is provided to consumers with their best features, then it becomes valuable. Maintenance of software engineering profession is a major part. Development of software environments requires equipment, exams, and many types of clients. The society demands daily maintenance of software. The software features are described through elements. Attributes are needed for invention. According to ISO 9126, functionality, reliability, efficiency, usability, maintainability and portability are six features, which are used to define the software quality. The unique rule of maintenance is the reliance of software metrics on the software's features. Maintainability has been characterized as "the easiness with which a software framework or factor can be changed to rectify errors, increase performance, or get used to an improved situation" (IEEE, 1990). During software maintenance, many properties are consumed. Object oriented software maintenance is expensive process in development period. At present, many techniques are used to reduce the cost of maintenance of software. The cost of software maintenance depends on the development period. It is found that association exist among maintainability object-oriented metrics, various studies have been performed previously. For maintenance exertion, metrics acts as predictors (Malviya and Badal, 2010). The investigators have utilized various methods, i.e., neural network and statistical techniques to create the correlation between software metrics and calculating maintenance effort. In the current scenario, for appropriate result in supporting software maintenance, is to apply data mining tools, as its capacity to work is determined by enormous data. Software maintainability is working continuously within software (Malviya and Singh, 2011). Software designer changes the software completely and reduces repair costs. Whereby its maintenance is estimated. To plan an object oriented structure, clustering based method is used. In the coming section, the K-means clustering method has been exposed by metric. Concepts, last attendance, and performance really happen only after the end result.
Chapter-1: Introduction

In Chapter-6, K-Means Clustering is a commonly used clustering process that attempts to reduce the median square distance between the concentrated in a similar group. Despite this fact, K-Means clustering does not ensure any accuracy. Its ease and speed are very attractive. Open source products focus on comparable applications that are fundamental these days. As a result, to design object-oriented software, it is necessary to identify maintenance. Different machine learning techniques have been implemented to find maintenance classes. In this chapter, the K-Means clustering method is used at the design level to identify the maintainable classes. Descriptive analysis is displayed in the performance metric as a confusion matrix. The issues of deciding the quantity of clusters as well as the clustering technique are unraveled at the same time by picking best model. In addition, the EM outcome gives a measure of vulnerability about the associated classification of every data point. We watched that that this approach can perform significantly better than standard method. It ignores the separation of those groups, which are either overlapping or different in different sizes. In this Chapter 6, the object-oriented software framework has been done through the k-means of maintenance-clustering. K-Means Clustering is a commonly used cluster process that is used to reduce the mean square distance between the focus on the same group. It does not ensure any accuracy; its ease of motion is extremely attractive. Open source products are fundamental to focus on comparable applications. Software maintenance is a very exaggerated and time-consuming process.

Chapter-7, Comparing Different Classification Techniques Using Data Mining Tools. This work has been done by Principal Component Analysis (PCA). The nature of the change-proneness of both products has been identified. PCA innovation improves the nature of the product in this section. Two commercial software are UIMS (User Interface System) and QUES (Quality Evaluation System), whose best source code metrics and maintenance are calculated.

Chapter-8, K-Means clustering and other classification techniques. The prediction model has been produced using the above mentioned machine learning strategies. Two professional dataset UIMS (User Interface Management System) and QUES (Quality Assessment System) have been used to perform its performance and
survey. The code for these two structures was made in classical Ada. The UIMS includes 39 classes and additionally QUES datasets involves 71 classes. In this chapter, examination, introduction of the specific proposed machine learning algorithms was appeared differently in relation to basic models for instance Decision Table, IBK, PART, Hyper Pipes, K-Star et cetera.

1.5 SOFTWARE METRICS

In previous research, it is seen that different software metrics suites is characterized by various variations in objectives, for example, exertion assessment, change-proneness, reusability, plus maintainability. Seven distinctive object-oriented metrics, for example, WMC, DIT, NOC, CBO, RFC, LCOM, and NOM are being measured used for forecasting maintainability of open source software. The points of interest of these metrics are portrayed in Chidamber and Kemerer (CK) object-oriented metrics (Kumar et.al, 2016).

Efficiency of metrics

When the maintenance information value is resolved, then the relationship between the software metrics continues with the maintenance. Maintenance 'change' has been described as "the number of lines changed per class " (Li and Henry, 1993). In our search, we have measured the change as a dependent variable, as well as all software metrics are independent variables during the creation of the connection. During the study, the dependent variable of the class of change is divided into two groups, which use a single cluster. Maintainability or change is hence a purpose of WMC, DIT, NOC, CBO, RFC, LCOM plus NOM (Kumar et.al, 2016).To examine efficiency of metrics utilized, different metrics are sorted into various clusters, for example,

i. Analysis 1- CK Metrics Suite is used to assess software maintenance. The maintainability functions is denoted by:

\[
\text{Maintainability} = \text{Change} = f(WMC, DIT, NOC, CBO, RFC, LCOM, NOM).
\]

ii. Analysis 2– Binary Univariate Logistic Regression (ULR) Exploration is considered to be effective in controlling the phase in each metric. Dual factors in the LR model have been measured to detect the level of each metric. These factors are as follows:
Chapter-I: Introduction

1. The Significance of regression coefficient: The coefficient of the metric indicates the amount of each metric in respect to maintenance.

2. P-value: p-value, demonstrates the importance of the relationship.

   The subset of the metrics that is huge, to evaluate the maintenance forecast is useful. The Maintainability function is described as follows:
   \[ \text{Maintainability} = \text{change} = f(\text{Reduced subset of metrics utilizing ULR analysis}) \]

iii. Analysis 3-During the investigation, cross-correlation analysis is implemented for inventing the subset in the metrics. It is necessary to measure change to evaluate maintainability, therefore purpose is signified in the following manner:
   \[ \text{Maintainability} = \text{change} = f(\text{Reduced subset of metrics utilizing correlation analysis}) \]

1.6 MAINTENANCE AND MAINTAINABILITY

The definition on behalf of software “maintenance” and “maintainability” are a lot of, but they are honestly dependable in choice and resolved. According to the IEEE standard meanings:

<table>
<thead>
<tr>
<th>Metrics Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weighted Methods per Class (WMC)</td>
<td>WMC is the method, to sum up the complexity of the class. WMC is the summation of entire complexities of techniques.</td>
</tr>
<tr>
<td>Depth of Inheritance Tree (DIT)</td>
<td>DIT of a class in an inheritance hierarchy. It is the maximum length from the class node to the root of the tree. DIT suggests for every class a degree of the inheritance levels from the object hierarchy topmost.</td>
</tr>
<tr>
<td>Number Of Children (NOC)</td>
<td>NOC totals the number of classes which inherit a specific class. NOC processes the number of immediate descendants of the class.</td>
</tr>
<tr>
<td>Coupling Between Object (CBO)</td>
<td>It is clear as whole quantity of different classes to which class is coupled. It deals the quantity of classes coupled with a particular class.</td>
</tr>
<tr>
<td>Response For Class (RFC)</td>
<td>RFC is the amount of the set of all techniques that can hypothetically be raised in response to entire techniques available inside the class hierarchy.</td>
</tr>
<tr>
<td>Lack of Cohesion in Methods (LCOM)</td>
<td>It is an amount of the number of techniques couples whenever resemblance is zero minus the count of method pairs where the resemblance is not zero.</td>
</tr>
<tr>
<td>Lines of Code (LOC)</td>
<td>It measures the sum of lines of code in a class.</td>
</tr>
</tbody>
</table>
**Maintenance:** The development of changing a software system or constituent subsequently distribution to precise errors, improves the performance or further attributes, or get used to an improved situation.

**Maintainability:** The simplicity by which a software system or constituent can be improved to precise errors, develop the performance or further attributes, or get used to an improved situation. Dependable with these characterizations, the maintenance method can be distributed into three parts of emphasis:

- **Corrective maintenance:** Maintenance achieved to reduce errors in hardware or software.
- **Adaptive maintenance:** Software maintenance is achieved through a better condition.
- **Perfective maintenance:** Software maintenance is done to increase performance, maintenance, or other properties of computer program.

Software maintenance is described as modifications in the maintenance period. To increase the utility of open source software such as Database, Format and Protocol, Game and Entertainment, Multimedia, Science and Engineering, different changes experience in the life cycle period. Otherwise the utility is included in the novel, or can expel the current utility, or by adjusting the structure on the occasion of need, through repair errors. Client necessities change according to period and situation. This might bring about different versions of open source software, as it needs a lot of assets required for software development. This is a direct result of disseminated and dynamic nature of Open source software. Open source software file is utilized for depicting the usefulness offered by an Open source software file. Maintainability of Open source software record is characterized by components which are influenced by the modifications (operation, Message, and so on) then the sort of modifications (addition, update, removed, move and so on.) incorporated. To obtain successful versions of open source software interface, the LoC and Understand SCI tools have been considered in this study. Maintenance is one of the modifications that the product structure is prepared in modified form during their maintenance phase. Software experiences several changes during one life-cycle to enhance
definite usefulness, or include new usefulness, or eliminate current usefulness, or repair the errors and adjustment the framework. It might bring about different versions of a software or open source software. This one is not at all simple assignment to modify rapidly a specific version of open source software, as per this one need a lot of assets required for software development. It is a direct result of appropriate updated open source software. In this research, maintainability of open source software has defined all sorts of changes (addition, move, removed, update and so on). LOC and SCI Tool have been considered in this study to look at succeeding versions of open source software (Kumar et. al, 2016).

1.7 PERFORMANCE EVALUATION PARAMETERS

The evaluation subsection provides the elementary definitions of the performance factors utilized for change proneness.

The confusion matrixes are categorized into four categories:

a). True positive (TP) are the quantity of classes correctly classified as change classes.

b). False positive (FP) denotes non-change classes incorrectly labeled as change classes.

c). True negatives (TN) relate to non-change classes correctly classified independently.

d). Lastly, false negative (FN) denote to change to change classes incorrectly classified as non-change classes.
These are the performance parameters utilized to measure the classification techniques.

- **Precision**
  It is utilized to number the degree of which the respond measurements in unchanged situations display the similar outcomes.
  \[
  \text{Precision} = \frac{TP}{FP+TP}
  \]  
  (1.1)

- **Recall**
  Recall represents number of significant objects that are to be recognized. They are represented as:
  \[
  \text{Recall} = \frac{TP}{FN+TP}
  \]  
  (1.2)

- **F-Measure**
  F-Measure combines the precision and recall numeric value to provide a single score, which is distinct as the harmonic means of the recall and precision. F-Measure is stated as:
  \[
  F - \text{Measure} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}
  \]  
  (1.3)

- **Specificity**
  Specificity focuses on how efficiently a classifier classifies the negative labels. It is characterized as:
  \[
  \text{Specificity} = \frac{TN}{FP+TN}
  \]  
  (1.4)

- **Accuracy**
  Accuracy measure is the fraction of estimated change- prone classes that are to be examined from modules/classes. It is characterized as:
  \[
  \text{Accuracy} = \frac{TN+TP}{TP+TN+FP+FN}
  \]  
  (1.5)