CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

In the last two decades, there has been an exponential reduction in computer hardware costs. On the other hand due to the Internet revolution the network speeds have exponentially increased. These have helped broaden the scope of use of software in the field of education, entertainment, travel, transport, banking, payments, voting and telecommunications and many more.

Software is now being used extensively in all phases of our lives, ranging from routine tasks like setting up calendar and alarms to maintaining and controlling the national infrastructures. Due to huge influx of data from various digital sources, speed and newer ecosystems Software development has gained a global presence as discussed by researchers like Shull et.al (2016) .These factors make it imperative to have an effective framework for Software Engineering.

Software Engineering primarily deals with the design and development of software followed by its maintenance and retiring of software. The increased use of technology by the society emphasizes the importance and need for research, application and study of newer software engineering quality paradigms to generate cost effective and budget friendly software[s] for companies, vendors and consumers alike. In providing quality and functionality of software as per needed standards, time limits and budgets in

delivery is still an issue. Along with these issues there exist residual faults and errors in software that need to be taken care of even after delivery and installation. Bugs in software could be due to error in source code or design of software. These errors in software can have serious implications for businesses as well as consumers due to their widespread application in critical functionalities like healthcare and transportation and so on, as well as daily routine activities. In a society that is highly dependent on computerization and thus on software, failures of software could result into huge financial losses, security risks, frauds, injury or fatality and environment disasters.

Providing error free software, correcting any faults in the software, proper functionality and focus on quality and reliability of software after deliverance is therefore an important goal of any software development and deployment team. Diligence towards quality of software is thus not an option but a principle requirement as discussed by many researchers like Arora et al. (2011), Jalote (2012), Pizzi et al. (2013) and Amid et al. (2013). These authors further indicated that many industries were not able to deliver high quality software for their clients and even fewer recognized the importance of software having correct quality attributes that would result in their efficient maintainability and functionality.

In pursuit to provide quality software, it becomes imperative for these software systems to be easy to maintain as errors are an inevitable part of software development. Software maintainability is a quality attribute that provides a probability for repairing and restoring the software system or its components after failure or modifications occur within acceptable cost, ease and speed. It is the ease with which software systems changes can be deployed for repair of faults and increased

performance and adaptation of a changed environment. In the nineties, Pigoski (1997), Li and Henry (1993) and recently Zhou & Leung (2007) showed that cost was a growing concern in maintenance of software. Recent researchers like Ajmal et.al. (2002) discussed that maintainability is a major cost concern and software maintenance costs are increasing. Researchers like Chen et.al. (2017), Mehdi et. al. (2013) and others along with the industry reports clearly suggest that a major cost i.e. nearly 75% of software systems life cycle is increased due to maintenance requirement.

The primary goal of software engineering is to develop quality software within the constraints of time and budget. For developed software to be considered of good quality it must match users' expectations and requirements.

The first section of our chapter thus discusses software quality in brief.

1.2 SOFTWARE QUALITY

The software requirements can be clearly defined or indirectly suggested. Similarly the end-use expectations from software may be implicit or explicit. Therefore as per IEEE (1993) software quality is defined as the extent to which a software meets or conforms to these requirements and expectations. The attributes for assessing software quality are defined by ISO standards (2001). These standards specify Functionality, Reliability, Usability, Efficiency, Maintainability and Portability as the key quality attributes of a software system.

Quality of software has taken center stage in the software development life cycle in the 1990s and beyond. This has been exemplified by Capability Maturity Models

(CMM) that involves methods for development and improvement of an organization's software development process.

In the current scenario though the software industry clearly understands the benefits of delivering high quality products, it faces challenges in doing so primarily due to lack of time with strict delivery deadlines, constrained budgets, no definite ways to measure the quality attributes and poor planning in different stages of software development process. Several researchers like Huang et al. (2013), Elish and Alshayeb (2011) and Dromey (1995) have emphasized that with the rapid increase in complexity and size of software systems, the industry has gaps on deliverance of quality products. They further emphasized that the industry sometimes completely overlooks some of the prime quality attributes.

In the highly competitive industry of software ignoring software quality can result in unnecessary cost strain, therefore quality assurance of software should be a part of all steps of software development cycle.

Though ISO standards define the attributes of quality of software but they specify no ways to measure these. Hence, a number of methods to measure these have been investigated over the years. As there is no clear consensus as to what exactly is the best way to measure these quality attributes, the search for different permutations, combinations will keep on going for possible improvements in measuring these quality attributes. Some of the representative research includes work done by researchers for the last few decades viz. McCall et.al. (1977), Boehm et. al. (1978), Bowen et. al. (1985), Sneed and Mercy(1985), Grady and Mercy(1987), Sommerville(1992)], Chidamber and Kemerer(1994), Dromey(1995)], Li et. al.

(2000), Black (2001), ISO-9126(2004), Aggarwal and Singh (2007), Sastry and Saradhi (2010), Elish and Alshayeb (2011) and Lee (2014). These works additionally emphasize the importance of maintainability as a way forward to improve quality of software.

This leads to our next section which discusses the role of software maintainability in relation with software quality.

1.3 SOFTWARE MAINTAINABILITY AN ESSENTIAL FACTOR OF SOFTWARE QUALITY

For categorization of maintenance of software the key word coined is "maintainability". It is considered as the first key quality of well designed software by Sommerville (1992) in the starting of his book. The method of altering the software that has been already delivered is called software maintenance and the effortlessness with which this can be achieved is defined as software maintainability as discussed by McCall et.al. (1977). Research done by Boehm et. al. (1978), McCall et.al. (1977) ,Broy et. al. (2006), ISO/IEC (2001), IEEE (1993), Dagpinar et. al. (2003),Kitchenham and Pickard (1983), Ghezzi. et. al. (1991), Dromey (1995), Rizvi and Khan(2010) , Koten and Gray (2006) and Elish and Elish(2009) all clearly indicate that maintainability is a key attribute of software quality.

To facilitate the creation of better quality software, the maintenance process is supported by an improved software maintainability parameter. A precise evaluation of software quality fully depends on maintainability assessment. Bowen et. al.(1985) , Sneed and Mercy(1985) proposed in their work that a lack of maintainability always

contributes to a higher maintenance cost and effort. Grady and Mercy (1985) and Oman and Hagemeister (1992) discussed that the aim of increasing the maintainability of object oriented software is not just to be able to identify defects but to be able to identify these defects as and when they occur.

Software helps organizations in keeping pace with the rapid change in technology, business types, and competitions in the market place. A major part of the literature on software engineering is focused on software development, despite many statistics showing that maintenance of software takes bulk of the budget in SDLC. As shown by Koskinens 2009 survey, costs due to maintenance resulted in 75-90% of development and usage of business and command software along with 50-80% that of cyber-physical software system. Developing software with good maintainability may point to the parts of software that can be reused, which parts require redevelopment, which parts require maintenance and the amount of effort required for that. Numerous researches like Sommerville (1992) and Parikh et. al.(1983) also state that 50-70% of the total life cycle incurred is majorly is used up on software maintenance. Chen et. al.(2017) suggested that good measure of maintainability will further lead to saving a major part of the total ownership cost (TOC) of software.

Software systems can exist as Procedural or object 0riented systems. With time the application of procedural software systems has become limited. Object oriented software systems are a norm nowadays. The focus of our work is also only for object oriented software systems.

The next section hence discusses the Object Oriented technology and the major design attributes of it.

1.4 OBJECT ORIENTED TECHNOLOGY

Object oriented technology is used extensively in development in all fields of software systems. These systems may range from programming languages, databases, linking and embedding to operating and graphics systems. Object oriented technology primarily group data structures and operations to be performed on these in entities called objects. In the initial stages of development cycle a considerable amount of effort is required to identify objects and classes, attributes and operations and identify associations between them. Object oriented programming is a fundamental technology as stated by Lee et.al (2014) and Chidamber et.al (1994) that hold up quality goals.

1.4.1 Design Properties of Object Oriented systems

Object oriented design properties direct the designers what to support and what to keep away. A number of measures have been defined so far to estimate object oriented design discussed by Gupta et al. (2015), Chauhan et al. (2014) and Venkatesan et al. (2013)[83]. The basic properties of Object Oriented systems contribute to and support internal attributes which form the basis for external qualities like maintainability McCall et.al.(1977) , Genero et. al.(2003), Li & Henry(1993), Rizvi and Khan(2010) and many more as detailed in literature review. These properties notably include Encapsulation, Abstraction, and relationships, Coupling, Cohesion, Inheritance and Polymorphism. Encapsulation deals with information hiding. Data from the classes is not available directly but can only be accessed by the services provided by the classes. Abstraction is related to the user perspective of necessary methods and attributes to define essential characteristics of an object. Three types of relationships i.e. aggregation, association and generalization exists between classes of an object. Cohesion refers to intra dependency between classes. For a

quality object oriented software it is required that we maximize cohesion and minimize coupling. Coupling refers to the interdependency among modules. Inheritance is used to create sub-classes from the existing classes by acquiring some of their attributes and operations. Polymorphism provides the flexibility to use objects in different forms within a parent class.

Practitioners and researchers frequently advocate that software maintainability should be planned at the design phase of development process. Therefore it is necessary to recognize object oriented design properties to quantify maintainability measures at design phase of software development process. During identification of design artifacts which have direct impact on maintainability measurement, a realistic view should be considered. If we consider all artifacts and measures then they become highly complicated, ineffective or time consuming. Therefore, there is a need to identify design artifacts and measures which affect the maintainability measurement process directly. In order to estimate maintainability, its direct measures are to be recognized.

Design level properties like abstraction, inheritance, cohesion, coupling encapsulation, etc. will be examined keeping in view their overall impact on software maintainability.

The next section talks about maintainability factors.

1.5 MAINTAINABILTY FACTORS

Maintainability is defined by ISO-9126 (2001) as having sub attributes viz. Analyzability, Changeability, Stability and Testability.

Software changeability is a significant part of maintainability, particularly in circumstances where there are many changes in software requirements and expectations. High- level designs have an impact on maintainability which influences changeability as it is a sub factor of maintainability, as defined in ISO-9126 (2001). The importance of maintainability and changeability of software can no longer be ignored or underestimated. The intricacy and the need to conform can complicate incorporating changes in software, if not thought over and incorporated early during design phase itself. Its early estimate lays down the foundation of making possible as well as easing out a software maintenance procedure. Consequently, as stated by Ayalew & Mguni (2013) it is a quality of the software that requires close up development cooperation with software maintenance.

Stability factor of software is an important and desirable feature of any standard software design. Black (2001) in his work discussed stability is defined as the point to which the software module can avoid unpredicted effect from the modifications of the software. If this factor is not as per the desirable standard it largely increases the impact of any modifications that take place on i.e. intensification of changes is resulted throughout the design. The consequences of this is a higher actual cost and effort than earlier estimated which in turn impact software maintainability due to possibility of induction of new errors as discussed by Ebad and Ahmed (2015) and Yau and Collofello (1985). Although, stability is most noticeably applicable during maintenance but by emphasizing on the factor of stability in the initial stages of development cycle, the software maintenance usefulness and effectiveness may be improved. In view of above, scholars and industry personal always recommend an effective and correct assessment of software stability early at development life cycle.

Regardless of the fact that stability is dynamic and most noteworthy to the system development lifecycle, it is very poorly achieved.

Thus, we can say that the risks we face in software stability due to unexpected effects of modifications are a major concern and it affects the overall maintainability cost of the software.

Calculating maintainability at a later stage often results in delayed reception of crucial information therefore causing a holdup in response and implementation about changes in software design as discussed in Aggarwal et. al. (2005), Elish et.al.(2009) and Mishra (2005). This results in an increase in terms of cost and additional work. Consequently, early estimation of maintainability in the software development cycle may improve design quality and decrease maintenance efforts and cost as discussed by Chaumun et.al. (2002), kiewkanya et.al. (2004), Mishra(2005), Muthanna et.al. (2000), Rizvi and Khan(2009), Dallal (2013) and Kumar et. al.(2015).

For researchers, quality controllers and programmers planning and evaluation of maintainability at design phase of the software development life cycle is thus of inevitable importance.

Taking these facts into consideration our research work is thus focused on evaluation of maintainability at design stage to deliver quality oriented maintainable software. Also after relevant study the quality characteristic of maintainability has been refined into its important sub-characteristics that have significant contribution in maintainability evaluation at design phase of software development cycle. After detailed review of work done by McCall et.al. (1977), Kiewkanya et.al.(2004), ISO-9126(2001), Rickard et.al. (2002), Aylew et.al. (2013), Chuamun et.al. (2002),

Heitlager et. al. (2007), Dallal(2013), Genero et. al.(2003) and Yau and Collofello (1980), it has been concluded that Changeability and Stability are the two most significant factors affecting software maintainability evaluation.

The next section gives the objective of this research work and explains the Problem statement.

1.6 OBJECTIVE OF THE PROPOSED RESEARCH WORK AND PROBLEM STATEMENT

It is evident from the above discussion that software maintainability should be evaluated at design phase of development life cycle. Practitioners emphasized on the need of having an organized and efficient approach for maintainability measurement.

Based on the proposed criterion to measure maintainability, the objectives of the research are to:

1. To develop and draw attention to the need and significance of maintainability evaluation model.
2. Highlight the phase at which maintainability be evaluated in order to get maximum out of it.
3. To identify maintainability factors and design constructs.
4. Display a relationship among maintainability with object oriented construct.
5. To validate the proposed maintainability evaluation model for better level acceptability.

In relation to the above questions that are pertinent to the concerned topic of the research, the study was designed to be a mix of qualitative and quantitative in nature. In order to address the above research problems, the problem statement that has been formulated for the research is identified as '**A Model for Maintainability Evaluation of Object Oriented Software at Early Phase'**. The problem is further subdivided into three sub problems enumerated as follows:

1) **Changeability Measurement Model (CEM^{OOD}) development:** During literature survey it was identified that changeability is a key factor to maintainability, and therefore this sub problem deals with developing a model to measure changeability. For this sub problem we develop the changeability measurement model with the help of object oriented design properties. This model shows a high correlation among changeability and design properties namely Encapsulation, Inheritance, Coupling and polymorphism. Empirical validation is used to validate the proposed model for better level of acceptability.

2) **Stability Measurement Model (SEM^{OOD}) development:** During literature survey it was also identified that Stability is a key factor to maintainability, and therefore, this sub problem deals with developing a model to estimate Stability. For this sub problem we develop the Stability measurement model with the help of object oriented design properties. This model shows a high correlation among Stability and design properties namely Encapsulation, Coupling and Inheritance. Empirical validation is applied to validate the proposed model for better level of acceptability.

3) **Maintainability Measurement Model (MM^{OOD}) development:** Changeability and Stability measures are used to develop maintainability measurement model that

works at design phase. In order to reinforce the claim of correlation between maintainability with changeability and stability, the proposed model has been tested and justified with the help of statistical measures. Finally, it incorporates the empirical validation of the maintainability measurement model. Also we have compared our proposed maintainability model with two existing maintainability models.

1.7 MOTIVATION AND SIGNIFICANCE OF OUR WORK

As extensively detailed earlier in this chapter we can summarize the following:

a) Ensuring software quality is a necessity for any software to be efficient within time and cost constraints.

b) Software maintainability is a key attribute of software quality.

c) Cost of software maintainability is a huge slice, nearly 50-90% of the total cost of ownership of the software.

d) Even though software maintainability is an essential factor of good software, it is often poorly managed by software industry.

e) Many standard organizations clearly state the attributes of software quality but they specify no ways to accurately measure these quality attributes.

f) Many researchers have over the years attempted to give better and practical methods to effectively evaluate these attributes but there is always a need for better solutions as a perfect fit solution will always be elusive.

g) These factors motivate us to propose an effective evaluation model for measurement of software maintainability taking into consideration the less explored combination of sub attributes of maintainability viz. changeability and stability.

1.8 OUTLINE OF THE THESIS

The Thesis is organized into the following six chapters.

CHAPTER 1: INTRODUCTION

This chapter provides introduction to the area, software quality, software maintainability measurement, maintainability related issues and its measurement at design phase, object oriented design, problem statement, its solution, motivation and significance of proposed research followed by thesis outline and summary.

CHAPTER 2: LITERATURE SURVEY

This chapter consists of a literature survey on relevant topics, prominently including maintainability models. It includes comprehensive report on software maintainability models and related issues, comparison of maintainability measurement models along with a critical examination of the same and contextual inferences and conclusions. The relevance of sub factors of changeability and stability in relation with Object Oriented properties and maintainability is explored extensively.

CHAPTER 3: CHANGEABILITY EVALUATION MODEL (CEM^{OOD})

This chapter discusses the proposed Changeability Measurement Model (CEM^{OOD}) for object oriented design and established statistical correlation between changeability

and design properties. The chapter also provides empirical validation of the changeability measurement model.

CHAPTER 4: STABILITY EVALUATION MODEL (SEM^{OOD})

This chapter illustrates the Stability Measurement Model (SEM^{OOD}) for object oriented design. The chapter also provides empirical validation of the stability measurement model.

CHAPTER 5: MAINTAINABILITY EVALUATION MODEL (MM^{OOD})

This chapter presents the Maintainability Measurement Model (MM^{OOD}) in terms of changeability and stability. Furthermore, the relationship of maintainability with these factors has been tested and justified with the help of statistical measures and validated using experimental tryout; it incorporates the empirical validation of the maintainability measurement model. Further our maintainability evaluation model (MM^{OOD}) has been compared with two existing maintainability models.

CHAPTER 6: CONCLUSION AND FUTURE WORK

Finally, this chapter highlights the major contributions and future direction of research on the topic.

1.9 SUMMARY

In this chapter we have introduced the area with the help of concepts like software quality especially maintainability of object oriented software. We illustrated maintainability factors and maintainability measurement in general and exclusively at design phase of development life cycle. Significance of maintainability measurement

and its importance at design phase has been analyzed for producing high quality software. Subsequently, problem statement, its solution, motivation and impact of proposed research is listed and finally the outline of the thesis is given chapter wise.

In the next chapter, we discuss the literature survey in detail.

CHAPTER 2

LITERATURE SURVEY

2.1 INTRODUCTION

Maintainability is “the ability to identify and fix a fault within a software component” as given by ISO-9126. Software maintainability is defined by IEEE (1990) as “the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment”. Researchers such as Aggarwal et. al. (2005), Al Dallal (2013), Sommervillie (1992), Boehm et. al. (1978), McCall et. al.(1977), Lee (2014), Kiewkanya et. al. (2004), Grady and Mercy(1987) and many more have made efforts to evaluate and quantify maintainability of software for the last few decades. Finding a way to measure maintainability accurately is important as many researchers like Lientz et.al.(1978), Foster (1993) and Glass(2003) along with others have stressed upon the fact that major cost of SDLC is incurred due to maintenance efforts of software. Efforts have been made for this but actual implementation of these measures is still at a comparatively smaller scale in the industry as many times this evaluation is rather subjective. The importance and advantages of evaluation of maintainability in early stages of Software development life cycle has been advocated by many researchers such as Malhotra and Chugh (2016), Rizvi and Khan (2011), Aggarwal et. al.(2006), Ping(2010), Hincheernan et. al.(2012) and Boehm et. al.(1978).The constant need for better methods of evaluation of maintainability for object oriented software at early SDLC phase has encouraged us to undertake this research.

The milestone definition of maintainability is given by ISO 9126-1. It is an expansion of previous work done by researchers like McCall et.al.(1977), Boehm et. al. (1978), FURPS+ (1987) in defining a set of software quality characteristics. ISO-9126 (2001) has identified four major sub characteristics of maintainability:

1) Changeability i.e the amount of effort required to change a system.

2) Stability i.e the negative impact due to system changes.

3) Testability is the effort needed to verify (test) a system change.

4) Analyzability characterizes the ability to identify the root cause of a failure within the software.

We have in section 2.4 studied in details the various maintainability factors and from that we can infer that changeability and stability are key factors of maintainability. Testability is usually studied as a separate external quality attribute rather than as a sub factor of maintainability.

Since the 1970s, the software engineering community has been doing experimental research on software maintainability. Research in this area has become even more relevant due to increasing complexity of software systems in the 21st century and the huge costs incurred in their maintenance.

Before going into a detailed literature survey, a brief time line in the research on maintainability can be summarized as:

McCall et.al. (1977) and Boehm et.al. (1978) proposed quality models in which they referred to maintainability as an important attribute of a good quality software[s].These models were used as a base by ISO to define its quality model ISO-

9126(2001) in which they defined maintainability as a key quality attribute having four sub factors of Analyzability, Changeability, Stability and Testability. Berns(1984) specified that the level of difficulty in understanding software systems had an impact on the maintenance effort involved. Maintainability measure was estimated using complexity metrics by researchers Sneed and mercy (1985), Chidamber and Kremer (1991). Grady and mercy (1987) used supportability as a factor to evaluate maintainability. Oman and Hagemeister (1992), Coleman et.al. (1994) emphasized that the analysis of software maintainability can act as a guide for decision making in software. Li and Henry (1993) used regression technique to calculate maintainability. All these models were superseded with more efficient models using similar or newer techniques to evaluate maintainability. These models however laid a solid foundation for the future work in the area of development of quality models around the attribute of maintainability.

In the early 2000s the models proposed by authors like Muthana (2000) for maintainability mostly focused on procedural software. These were very effective for simple applications. With the introduction of large and complex systems based on Object Oriented approach and design attributes of polymorphism, encapsulation and inheritance, need for quality models for this approach became very evident. Despite this fact, not enough research works have been committed to explore the concepts of software maintainability in object oriented systems.

The Extensive literature survey done by us majorly focused on software maintainability in object oriented software at design phase but for completeness we have mentioned few papers on structured approach too. Further, we have arranged our broad literature survey in sections that review work done in software maintainability

at main three stages of SDLC viz. Analysis Phase, Design Phase and Coding Phase. Since this dissertation focuses on maintainability and its two sub factors changeability and stability, literature survey on these two important sub factors is provided in detail.

In this chapter we present the summary of the systematic literature review done to gather evidence on software maintainability measurement of object oriented design.

2.2 RELATED WORK ON MAINTAINABILITY

As discussed earlier the researchers have tried to evaluate maintainability at various phases. The critical review of the related work on the topic is mentioned in the following sub sections.

2.2.1 Maintainability at Analysis Phase:

Maintainability can be measured at different phases of software life cycle. Here we discuss few researchers work done on maintainability at analysis phase of SDLC.

David E. Percy (1981) used an evaluation process that followed four distinct stages of planning, calibration, assessment analysis and reporting by using questionnaires checklist by five independent evaluators during initial phases of development life cycle. They with effort arrived at a maintainability evaluation method that was cost effective and could to some reasonable degree be implemented. The authors used linear regression to evaluate a model for reliability. They further used reliability factor to evaluate maintainability at analysis phase for procedural modules of software.

Polo et al. (2001) studied code metrics for legacy programs and applied logistic regression to find correlation between these metrics and maintainability requirements. Their work provided a guideline of estimating maintainability of outsourced projects

in the initial stages when the maintenance contracts are signed and there is very little information about the software that has to be maintained. In spite of the limitations positive results were shown for maintainability estimation in these cases.

Polo et.al (2002) developed a methodology "MANTEMA" for prediction of maintainability as an extension of ISO/IEC 12207 standard. The well established European consultants Atos ODS applied this for software maintainability in collaboration with university. In their work the authors showed positive results on method that could help in arriving at service level agreements with the outsourcing organizations for maintenance of software on experimental basis even though details of information of the software to be maintained are not completely known.

Wensheng Hu et al. (2014) discussed that analysis of requirements is an important phase in SDLC. A large enough percentage of total faults that occur later and need maintenance are from requirement phase. Unambiguity in defining requirements specification leads to project success and consistency. Using Natural Language Processing methods and grey scale correlation, the authors in this paper presented a classification method in which firstly the keywords from various functional requirements were segregated and assigned a weight vector. Secondly using grey system, grey correlation coefficient was computed for these weight vectors and further used to construct a correlation matrix. Statistical tools were used to classify the specification statements of functional requirements. This work could provide guidance in improving development and maintenance of software.

2.2.2 Maintainability at Design Phase:

Many researchers worked on software maintainability at design phase with varying success and conclusions. Comprehensive studies of these are presented here.

Muthanna et al. (2000) proposed a model based on Polynomial Linear Regression for structured software at design phase of SDLC. The authors used software design metrics and statistically evaluated maintainability for software systems. Using these metrics they identified error prone modules which could assist the designers calculate better maintainability models .This model was not applicable to Object Oriented systems.

Subramanyam et al. (2003) used a subset of C&K metrics to help determine software defects in the early stages of SDLC. They used industry data from software developed in two Object Oriented languages viz. C++ and JAVA. The authors showed a significant association between these metrics and software defects irrespective of the size of the considered softwares.

Kiewkanya et al. (2004) used three methods to develop maintainability models and then presented a comparison between these three models. These maintainability models used two sub factors of maintainability namely modifiability and understandability. The first technique uses metrics discrimination analysis to correlate pattern between maintainability and design metrics of structural complexity. In the second technique weighted-score-levels of understandability and modifiability were converted into scores. The third method applies weighted sums which are a combination of levels of modifiability and understandability, obtained after the application of the two models of understandability and modifiability. A comparison

of maintainability models obtained from three techniques using methods of association, aggregation and classification were also done. The results showed that there was close accurateness of results of models obtained using the first two methods. There are however constraints for these models to give good results: Firstly for metrics determinant model, it is required to have an automated tool for the measurement its metrics. Secondly for weighted score model questionnaire technique is needed to get understandability and modifiability score for the relevant softwares, which is not a very accurate method of getting scores. Lucca(2004) developed a web application maintainability model specifically for web based applications using metrics like size, coupling and complexity metrics.

Genero et al. (2005) carried out trial analysis on relation between maintenance of UML (Unified Modeling Language) class attributes and a range of complexity metrics. From these trial analyses the authors found two metrics to be significantly affecting maintenance efforts viz. number of methods and number of associations. After laboratory experiments used to evaluate these two metrics, it was determined that these amounted for 28% of the maintenance efforts. The results indicated that in future research these factors could be explored in detail to predict maintenance estimations.

Koten and Gray (2006) used Bayesian Belief Network (BNN) on Li and Henry's datasets for predicting maintainability in object oriented systems. Systems. These data were collected from various Object Oriented systems. The author compared results from two systems namely UIMS and QUES by applying regression methods. He however did not give a generalized model for all object oriented systems for evaluating maintainability with accuracy. The author showed his model gave better

performance compared to the models developed using regression analysis till that time. Aggarwal and Singh (2007) used artificial neural networks ANN to predict maintainability for a Object Oriented software and they used the metrics LCOM, DIT, WMC, NOC, RFC, DAC, MPC, NOM for this. The results presented by them were adequate but not very high for prediction of maintainability. The performance of their model was largely dependent on the training data.

Genero (2007) stresses the relevance of having methods to measure the design properties like structural complexity based on associations and generalizations of Object Oriented systems. They further studied whether metrics of class diagrams could be used to predict the two sub factors of maintainability i.e. understandability and modifiability. The measures studied were more correlated with the subjective estimations about the complexity of class diagrams, hence these needed to be supported empirically with real life projects datasets and values from the industry which the authors proposed as limitations of their work and also a future direction of work. Breesam (2007) attempted to validate a set of metrics of class diagrams that can be used to measure quality of Object Oriented systems in terms of class inheritance based on generalizations and specializations. They used analytical and empirical methods to obtain results from these metrics. The data used to validate the studied metrics was limited by the experience of the students in Object Oriented Systems.

Zhou et.al.(2007) employed a new method called MARS i.e Multiple adaptive regression splines to predict maintainability of Object Oriented systems. They used the two databases given by Li and Henry (1993). They compared their results with four prediction models based on different techniques to measure maintainability. These four models were artificial neural networks models, support vector models,

multivariate regression models and regression tree models. The results fared better for their model as compared to the four models for one data set and almost equivalent if not better for the other dataset.

M.O. Elish et. al (2009) also used Li and Henry datasets to develop a model called TreeNets to predict maintainability for Object Oriented Systems. The TreeNet model is essentially based on data mining techniques and is an extension of CART i.e. Classification and tree regression method. Though a few maintainability prediction models as given by Lucia(2005) based on calculating adaptive maintainability prediction, Koten (2006) used BNN methods as discussed earlier, Misra (2005) employed corrective maintainability effort were available bur the prediction accuracy of these models were found to fare badly on criteria specified by Conte(1986)and MacDonnell (1997). TreeNet model was thus proposed o provide better maintainability prediction accuracy. The authors were further able to compare their results with the five models considered by Zhou et.al. The results showed that the TreeNet model achieved better prediction results. Further work was though required with other datasets to provide additional support to the results of this work as to realize the full implications and possible limitations that occurred due to these datasets.

S. Rizvi et al.(2010) selected metrics like number of classes, attributes, methods, associations, aggregations, dependencies, generalizations, aggregation hierarchies, generalization hierarchies , maximum depth of inheritance and so on from the data given by Genero(2007) , to express the internal quality attributes of modifiability and understandability of Object Oriented softwares using multiple linear regression. The authors further used these models of internal quality attributes to derive a model for

maintainability. They further validated their model using a set of real life projects. They had better results as compared to the TreeNet model proposed by Elish (2009). Though they arrived at satisfactory values of estimating maintainability but as future work proposed that more efficient models could be developed using other factors of quality from ISO-9126. Sastry and Saradhi (2010) tried to apply software metrics using GUI and scrutinized relationships between metrics and quality attributes.

Malhotra and Jain (2011) reviewed the metrics like coupling, cohesion, inheritance and so on that can be used to predict maintainability , fault proneness, reliability etc. The authors have studied numerous software metrics and varied literature that use different subsets of these metrics. They have also reviewed the different approaches like Support vector machines, naive bayes network, random forest, artificial neural network, decision tree, logistic regression etc. followed by researchers to analyze different datasets. The purpose of this study is to provide various researchers a platform for comparative analysis for identifying maintainability and fault proneness of Object Oriented systems. Gautama Kang et. al. (2011) used two internal quality sub actors viz. understandability, modifiability along with addition of two metrics scalability and class complexity to calculate maintainability. The authors inferred better correlation values of these four factors with that the compound MEMOOD model for maintainability.

Y. Dash (2012) proposed a maintainability model based on MLP(multi layer perception) and the authors calculated maintainability effort as a dependent variable with principal components of Object Oriented metrics as independent variables. These variables were like DIT, NOC, NOM, size and so on. They calculated the r-correlation coefficient and validated it to be superior to the WARD neural network

model proposed by Thwin (2005). Chug and Malhotra (2012) proposed a maintainability prediction model based on machine learning. They used three Machine learning algorithms namely: GMDH- Group Method of Data Handling, PNN- Probabilistic Neural Networks and GA- Genetic Algorithm. The two data sets used were from Li and Henry viz. QUES and UIMS. They showed that their prediction accuracy for GMHD models at pred(0.25) and pred(0.30) were much better than the values achieved for previous models that they have compared according to criterion set up by Conte(1986) and Mac Donell (1997).They claimed that their (GMDH) network model is good model for estimating maintainability of software. At the code level Hincheeranan et al (2012) have proposed models for two sub factors of external quality attribute: maintainability viz. flexibility and extendibility. They have suggested a tool for calculating maintainability based on four components viz. UML case tool , XML parser, Metric calculate and display metric results using the two sub factors of flexibility and extendibility. The proposed tool has not been developed or validated mathematically or empirically.

Al Dallal J. (2013) empirically studied the relationships between class quality attributes of size, cohesion and coupling with maintainability. The author developed a model based on the mentioned internal attributes to calculate maintainability. The results showed that classes with good class qualities of higher cohesion, lower coupling and smaller sizes were easier to maintain than those with poor quality values. This model thus helps identify the classes with low maintainability and assists in testing and documentation of the same for improving maintainability of softwares. Chug and Malhotra (2013) explored the meaning and measure of maintainability in the changed scenario where databases were heavily used by windows and web based

applications. For these data intensive applications they proposed a new set of metrics and using data from five real- world applications calculated maintainability using ANN(Artificial Neural Network) technique. The outcome from their study showed fair results for predicting maintainability for medium sized systems. As a future work they proposed calculating maintainability for larger Object Oriented systems and aspect oriented software development methods.

R. Malhotra and Chug (2013) have proposed a new metric suite, an extension of Chidamber and Kremer metric suite. They proposed to add two new metrics NODBC and SCCR .They have evaluated and analyzed this metric suite for their effectiveness for predicting maintainability of Object Oriented softwares. They have validated the model for data intensive softwares. These are implemented both at design and code phase of Software development life cycle.

Rajendra et. al (2015) estimated the maintainability of Object Oriented systems using the two sub factors of maintainability as flexibility and extendibility. They further established significant models for each of these sub factors internal attributes from the design properties attributes using multivariate linear regression technique. They calculated flexibility in terms of design properties coupling, cohesion and inheritance. They calculated the value of extendibility using design properties of coupling, cohesion and polymorphism. The authors further calculated the value for external quality attribute maintainability with flexibility and extendibility as independent variables. . The results they arrived were significant but using other factors of quality factors newer models for maintainability with improved results could be proposed.

Malhotra and Chug (2016), in their paper compiled a systematic review of studies on software maintainability between the years 1991 to 2015.The authors arranged and analyzed the work on maintainability using tangents of design metrics, tools and algorithms, data sources and so on. They summarized the following facts:

- Maintainability is still a major attribute of software quality.
- Maintainability still amounts to a major part of costs incurred in software development.
- Maintainability is more effective if measured in early phases of SDLC.
- Measure of Object Oriented design properties like coupling, cohesion, inheritance and so on gives best results for predictive maintainability.
- Design metrics remain the best method to obtain the characteristics of given software.

They compiled a total of 96 studies from various journals, conference proceedings and others. After an extensive review they also proposed that newer research can be done on open source datasets.

Chen et.al. (2017) in their paper stressed the huge level of cost saving in software by understanding the importance of software maintainability, and suggested answers to questions of decision regarding what parts of software to be reused, what parts to be redeveloped, the theoretical estimation of effort required to do so and thus giving indicators as how to reduce ownership costs.

2.2.3 Maintainability at Code Phase:

Hayes et al. (2003) in their new approach OMA (observe-Mine-Adopt) used maintainability product and perceived maintainability as two measures of maintainability as to improve software practices for better maintainability. The authors suggested that observations regarding which things work and which do not occur naturally during the process of software study. These observations when mined can be used to validate processes and practices which can be formalized as to be adopted by the team.

Hayes et al.(2004) used person hours as a measurement for estimating adaptive software maintainability model called AMEffMo. They also used COCOMO and regression analysis methods for providing managers and maintainers with useful information regarding adaptive maintenance efforts.

Prasanth (2008) proposed a method for estimating maintainability in terms of code complexity. The authors took samples of four and made assessments of complexity in absolute and relative terms. Code complexity is measured at testing phase. The authors used the fuzzy repertory table (FRT) technique for obtaining domain knowledge from testers for the software used for complexity analysis. They then used regression analysis to predict maintainability from the product's code complexity.

Jin and JA Liu (2010) proposed a SVM and clustering technique to predict the software maintenance effort. The probability value calculated by the authors showed a statistical significant correlation between the predicted and actual maintainability efforts. This predictor could be used to predict the inclusion of modules from

incremental releases of similar software for better maintainability. They carried out the maintainability analysis at source code stage of Software development life cycle. They studied the code written in C++ for HTML pages which could be part of software applications as standalone software or an embedded component.

2.2.4 Maintainability at Development Life Cycle:

M. Genero (2003) suggested in their work that using early metrics for analysis and design in Object Oriented software can greatly enhance decision making. The external quality attribute of maintainability can be better measured based on these metrics. They conducted a controlled experiment, gathered empirical data and showed positive results. The results showed that early metrics measuring internal attributes like UML class diagrams and structural complexity leads to a fair chance of obtaining good maintainability indicators based on which maintainability models can be developed. They also suggest further empirical studies especially based on industrial data for a more comprehensive outcome.

Prasanth et al (2009) used a method for predicting maintainability of software using code complexity at the testing phase. Absolute and relative complexity measurement was done from four sample products. Domain knowledge of testing experts is collected through FRT- fuzzy repertory table technique. Maintainability is predicted using regression analysis from samples code complexity.

A complete charting of the existing Maintainability Models proposed by Various Expert has been done in Table 2.1.

Table 2.1: A Systematic View of Maintainability Models Consider by Various researchers.

Year	Study/Author	Maintainability Evaluation Approach/Model	SDLC Phase	Validation
1984	G.M-Berns	Maintainability Analysis Tool for use with FORTRAN on a VAX	Not given	No Implementation
1985	T.P. Bowens	Average number of days to repair code.	Code level	No
1985	Sneed Mercy	Fuzzy Model	Code Level	No
1987	Kafura and Reddy	Cyclomatic complexity as well as six other software complexity metrics	Code Level	No
1987	Robert Grady (At HP)	FURPS Model	Code Level	Theoretical justification
1991	Geoffrey & kemere	Cyclomatic Complexity Density	Code Level	Yes

Continued on page……

Table 2.1 continued…

1992	Oman Hagemeister	Halstead's Effort (aveE), McCabe' Cyclomatic Complexity (G), LOC (Lines of Code)	Code Level	No
1993	Li Henry	Henry model based on coupling between classes	Code Level	Yes
1994	Coleman Oman	Oman model	Code Level	Yes
1995	Welker Oman	(Improved Oman Model) Cyclomatic Complexity V(g'),LOC (Lines of Code)	Code Level	No
1995	Dromey's "Quality Model"	Quality Model	Code Level	Theoretical justification
2000	Muthanna et al.	Model based on Polynomial Linear Regression	Design Phase	No
2003	Huffman Hayes et al.	Observe Mine Adopt (OMA) Based on Maintainability product	Code Level	No

Continued on page……

Table 2.1 Continued….

2004	Lucca Fasolino WAMM	Web Application Maintainability Model	Web based Approach	Web based Approach
2005	Hayes Zaho	(Main Pred Model) LOC (Lines of Code), TCR (True Comment Ratio)	Code level	No
2006	Koten Gray	Bayesian Network Maintainability Prediction Model	Design Phase	Yes
2008	Prasanth Ganesh & Dalton	With the help of FRT(Fuzzy Repertory Table)	Testing Phase	No
2009	MO. Elish & KO Elish	Produced Treenet model using stochastic gradient boosting	Design Phase	Yes
2010	C Jin & JA Liu	Based on Support vector machine	Code level	Based on vector machine
2010	S. Rizvi et al.	MEMOOD Model	Design Phase	Yes
2011	Gautama Kang	Compound Memood Model	Design Phase	No

Continued on page……

Table 2.1 continued…..				
2012	Alisara et al.	Maintainability Estimation Tool (MET)	Code level	No
2013	Al Dallal, J.	Object-oriented class maintainability prediction using internal quality attributes.	Design and code level	No
2014	R. & Chug A.	A Metric Suite for Predicting Software Maintainability in Data Intensive Applications.	Design Phase	Based on Metrics
2015	Singh et al.	Estimation of Maintainability in Object Oriented Design Phase: State of the art	Design phase	Theoretical Explanations
2015	Rajendra et.al	Model based on Object Oriented design properties	Design Phase	No

2.3 MAINTAINABILITY FACTORS

Various permutation and combinations of metrics and internal quality attributes have been suggested by several researchers like Genero (2007),Rizvi (2010), Rajendra (2015), Aggarwal (2006), for evaluating maintainability quality attribute of Object Oriented software for different phases of SDLC and at design phase. Table 2.2

provides a comprehensive view of the maintainability factors recognized by area experts of Table 2.1. It is also highlighted from the table that Changeability and Stability are the significant maintainability factors.

Table 2.2: Maintainability Factors Consider by Various Experts

Sub-characteristics ➔ / **Models ⬇**	**Analyzability**	**Changeability**	**Cohesiveness**	**Complexity**	**Flexibility**	**Modifiability**	**Modularity**	**Stability**	**Testability**	**Understandability**
J. A. McCall		**X**					**X**			
ISO 9126-1	**X**	**X**						**X**	**X**	
R. Land		**X**			**X**	**X**			**X**	
Dubey et.al		**X**								
A Chaumun et. al		**X**								
B. W. Boehm					**X**				**X**	**X**
Y. Ayalew et.al.	**X**	**X**								
I. Heitlager	**X**	**X**						**X**		
Al dallal et.al.		**X**								
D. Peercy							**X**			
H. Sneed *et al.*			**X**							
S. S. Yau *et al.*				**X**				**X**		

Continued on page…..

Table 2.2 continued……

IEEE Std.					**X**					**X**
G. R. Dromey						X				
Elish et.al.		X								
S. W. A. Rizvi et al.						X			X	**X**
Rajendra et.al.					**X**					
Genero et. al.	X					X		X		**X**
Godin et. al.		X						X		
M Alshayeb et. al.								X		
M. J. Kiewkanya et al.		X				X			X	**X**
Li et.al.		X								
Hagemeister et. al.								X		
Genero et. al.	X					X		X		**X**
Godin et. al.		X						X		
M. J. Kiewkanya et al.		X				X			X	**X**

2.3.1 Design Properties That Influences Maintainability

Object oriented design properties overcome the negative aspect of procedure oriented design. In order to design the software through an object oriented approach, the three essential properties are considerably being used i.e. encapsulation, inheritance and coupling. Object oriented design properties that have positive impact on maintainability evaluation has been identified and consolidated chart for the same is given in Table 2.3.

Table 2.3: Object oriented design properties contributing in maintainability evaluation: a critical look

Design Properties ➔ / Author/Study Changeability/Stability ⬇	Encapsulation	Coupling	Inheritance	Polymorphism
Li and Offutt (1996)	X		X	X
Godin et. al (2000)		X		
Arisholm et. al (2000)		X		
A Chaumun et. al (2002)	X			X
Heitlager et. al. (2007)		X		X
Riaz et.al (2009)	X	X		
Abidi (2009)		X		
Dubey et.al (2011)	X	X	X	X
A Hincheeranan (2012)	X	X	X	X
Y Aylew et. al (2013)		X		
Al Dallal et. al (2013)		X		
Malhotra et. al (2013)	X	X	X	
Ankita et. al. (2014)	X	X	X	X
Elish et al (2010)	X	X	X	
Ebad et al. (2015)		X		

2.4 LITERATURE SURVEY ON CHANGEABILITY

Researchers, practitioners and quality controllers emphasize on the need of having a systematic approach for changeability measurement. They argue that changeability can be measured at design phase by assessing the design level metrics of changeability. The contextual findings of related work on software changeability and the approaches available for its measurement may be summarized as follows:

In a study by H. Kabaili et al. (2001), the authors have discussed cohesion as a changeability indicator in SDLC phases. In this work, authors explored whether there exists a correlation between cohesion and changeability. Two cohesion metrics, LCC and LCOM were considered by author for estimating software changeability and a model of change impact was applied. To test the hypothesis that cohesion and changeability are correlated, researchers inspected the well known cohesion metrics, LCC but due to deficit of resources, were unable to examine the complete list of proposed sixty six changes of their impact model for object oriented language C++. Rather, researchers limited themselves to a subset of six changes which they preferred according to a set of four chosen criteria. They could not come up with a strong correlation between cohesion and changeability. They could not prove with conformity that defined cohesion metrics were good indicator of changeability.

Study done by M. Ajmal Chaumun (2002) discussed Changeability in terms of measuring change impact while considering correlation coefficient among two variables and a WMC metric. After deleting the outliers cases, the correlation coefficients was found to be weak and of the order of 0.55. Using Anova test they were though able to support that WMC metric and change impact is related. They

applied their study in application areas such as telecommunications. Using change Impact analysis for assessing changeability of software systems they were able to derive a limited model to implement successful system compilation after changes. A generalized model for performance of changeability was though not proposed by them.

Study done by M. K. Abdi et al. (2009) proposed a probabilistic method having Bayesian networks as opposed to earlier non-probabilistic approaches to help analyze change impact in object oriented systems. They primarily used coupling measurements like coupling between objects including classes used by target, coupling with no ancestors and so on to verify their approach. They used three scenarios in which a correlation hypothesis amid different metrics of coupling and the change impact that had been previously established in former works. In these three scenarios the change impact was found to be weak, of the order of 0.46, 0.48, 0. 54. In the fourth scenario the results in relationship proposed between these metrics and change impact contradicted to earlier results, leading them to search a hypothesis explaining factors like complexity and system size. This work suggests methods for improving maintenance of software in object oriented systems and focuses on change impact analysis in generic SDLC phases.

Yirsaw Ayalew et al. (2013) used cases on open source software and tried to explore impact of coupling and complexity metric in changeability and assess modularity of the system. The authors used three coupling metrics as indicators of changeability on open source software and showed that coupling metrics may be good indicators of changeability. In their work the authors provided theoretical approach for measuring changeability and extensibility of aspect oriented software. Moreover, no quantitative

changeability measurement model has been provided in this work. In the paper by Sun et al. (2012) an approach was developed to estimate a software system's changeability using two steps. The first method was using the formal theory analysis to do change impact analysis (CIA) that estimated the cascading effect of the proposed alteration .The author further proposed a new impact metric to demonstrate the capability in the system to absorb these changes. Study on three case application programs showed the usefulness of proposed approach of changeability evaluation. Depending on a questionnaires analysis, the study classified the change impact analysis (CIA) according to their impact on system changeability. Further based on outcomes, author proposed guidelines for making design decisions, and provides theoretical guidelines to improve system changeability. In this study the quantitative measure for improvement of changeability was not given and theoretical guidelines are not clear about the cause effect relation between given patterns.

The authors Malhotra et al. (2013) proposed a change proneness prediction model which predicted classes that showed change proneness by means of object oriented design metrics. The model proposed was fully based on open source software data sets. They analyzed and reused the produced estimate model of a chosen project and mapped it on a separate project thereby reducing to some extent dependency of training data on development of prediction model.

Measurement of change prone classes of software was done using non liner data fitting bi square method with robust results by Ankita et al.(2014). They would have extended their work by using probability density function to give better insight into the nature of mathematical relationship between the change-proneness and the factors/random variables that influence it. Moreover, this model was not empirically

validated and not applicable in the context. These outcomes though have not had a wide acceptance and thus, have not been used in practical by the practitioners. In addition, the model provided by the authors is not sufficient for both structural and behavioral architecture.

In the work done by Panjeta et al. (2014) authors highlighted changeability as one of the key characteristics of software maintainability. They theoretically and graphically tried solving this important subject by proposing a structure that facilitates to evaluate level of changeability by means of clustering methods (Machine Learning). To quantify changeability, authors proposed theoretical and graphical approach; but quantifying changeability through this technique showed high complexity. In this study, the authors have not given the quantitative measure of software changeability. However, they have only discussed theoretical approach for measuring software changeability.

Work done by Sen-Tarng Lai (2014) proposed a model for improving process of plan change in software project and mitigating development risk. A model called (WBSPM) i.e. WBS–based Plan changeability was proposed to increase change capability plan in WBS-based method taking into consideration changeability factors. This approach did not propose a generic model for changeability at design phase. This study is largely concerned with recognizing and assessing the factors of changeability in object oriented software and metrics correlated to the factors, which are been backed by the case studies. The authors used the source code analysis for characterizing the software changeability. In this research, authors identify possible relevant metrics to predict the class changeability and analyzed the approach in

theoretical manner only. Moreover, this approach has more emphasis on analysis phase; design phase has been considered only partially.

In the study done by Rongviriyapanish et al et al. (2016) java code changeability prediction model was proposed. Authors highlighted a value model for evaluating the levels of changeability in java program as significant in software development. The paper proposed a software changeability estimation model that took into account the metrics involving several appropriate object oriented attributes. The proposed method presented by using the multi layer perception, as a classifier arrangement and for training data of java classes from jEdit open source software project. An approximation of 74.07% was attained and the model could completely divide java classes with decent changeability level ranging from reduced or acceptable changeability levels. The proposed java code changeability prediction model measured changeability at source code level only. Study argued this model improved maintenance, debugging and hence improves software quality.

After a systematic literature review it comes into observation that there are numerous approaches available for measuring object oriented software changeability at analysis and coding phase. However at analysis phase, we only have the requirements and at design phase only, the complete structure of the software comes into picture. Therefore, changeability assessment at design phase is much more relevant as compared to analysis phase and also cannot be compensated during subsequent development life cycle. Panjeta et al. (2014) proposed theoretical and graphical approach for changeability assessment at design phase but quantifying changeability through this technique is very complex. Hence, there is a potential to develop a systematic solution for changeability evaluation at design phase in SDLC. Therefore,

a comprehensive outline and related model to evaluate changeability of object oriented software with the help of object oriented design properties at design phase seems highly needed and significant. At analysis phase, we have the functionality requirement and only at design phase, the complete structure of the software comes into picture. The lack of software changeability at design phase may be difficult to overcome throughout ensuing system development life cycle. Hence, there is a potential to develop a systematic solution for changeability evaluation which is implemented at design phase of different stages of SDLC. Therefore, a comprehensive outline and related model to evaluate changeability of object oriented software with the help of object- oriented design properties at design phase seems highly needed and significant.

2.5 LITERATURE SURVEY ON STABILITY

This section shows the result of a organized literature review, conducted to collect evidence on software stability evaluation of object oriented design and development. To evaluate the stability of software, a developer can follow one of the two approaches i.e measure stability in terms of single software and consider its stability values based on interrelationships, among software modules that project the degree of probability of ripple effects of changes on modules. Component level metrics are used to evaluate the design stability in this approach. The second approach uses software evolution history to measure stability across the different version of software. Architect level metric and program level metric are used to evaluate the differences among two version of developing software by Jazayeri (2002) and samadzeh (1994).

We are following the first approach and calculating stability at design phase. We can obtain designs stability measures at any level in the design procedure of a software thus facilitating solution for stability problems initially in the life cycle of software development. Black(2001), Bahsoon (2004)researched how software design stability assessment includes an added in depth study of the interface's of software components, and an account of the ripple effect as a importance of programs modification (stability of the program). The possible ripple effect is well-defined as the entire numbers of assumption complete by another component which invoke a component whose software stability is being evaluated share global data or file with component or are invoked by the component. Parts of the programs with lower stability can then be re-designed to improve the situation.

Though the emphasis of this paper, is on the above discussed first approach as in traditional software models but for completeness of literature survey this section also includes literature on evolution stability and architectural stability that cover different iterations of the software, and across various phases of the life-cycle of software development.

Chidamber & Kemerer (1994) theoretically suggested some software metrics as good pointers for the stability, of object oriented design and development, based on measurement theory and viewpoints of software engineers who were experienced and after evaluation with regards to standard criterion suggested that these metrics had desirable properties and could be used for good software design. They clearly indicated that future research should be done to ascertain this new approach for object-oriented software design approach in terms of required design features as against traditional approaches.

Li et al (2000) proposed three metrics for the objective of calculating object oriented software development, project progress and adjustment of project strategy in real time the researchers also did a study of software design instability that scrutinizes how the implementation of a particular class can distress its overall design. The work controls, that around few aspects of object-oriented design and development are fully autonomous of implementations although other parts are at the mercy of on implementation. They emphasized that metrics that are developed by Chidamber & Kemerer cannot quantity all parts of object oriented software design. Instances of these parts are the modification in the class member, class numbers and between class inheritance relation. Since this faultiness of C&K metrics, researchers developed these three metrics namely “system design instability” (SDI), “Class implementations instability” (CII) and last one “system implementations instability” (SII). The key objective that keen out in this work is to explain how the data that is collected from theses metrics can support developers and project manager to correct the project plan. These metrics were experimentally scrutinized compared to C&K metrics. Authors found out that SDI and CII assessment of Object Oriented features that are dissimilar from the features that are assessed by C&K metrics.

Fayad (2003) conversed the notions of, permanent business theme (EBT), business object (BO) and finally industrial object (IO) and just in what way they can be, used to shape a stable software design. EBT are those essential thoughts of the software that continue stable ended a time. BO is outwardly stable over time but might have interior variations. IO is exterior and unstable object of the software system. The author, claimed that software design structure fully depended on only these ideas would decrease re-engineering and therefore produce a stable software design.

Unfortunately, in their study the researchers providing only theoretic procedures for stable designs. Furthermore no measurable stability evaluation model has been providing in this study.

Work done by Elish et. al. (2010) did the future research on the object oriented design and development metrics developed by Chidamber and Kemerer, adopting the metrics suggested by them, as applicant pointers of the software stability of object oriented design and development. The purpose was to examine there are correlation amongst these metrics and the design stability for class. The investigational outputs specified that 'WMC', 'DIT', 'LCOM', 'RFC' and, 'CBO', metrics are destructively correlated with the stability of between software class. No correlations were found among "NOC" metrics and the design stability of between classes.

AlShayeb et.al. (2011) calculated the influence of system refactoring on software architecture stability and as a outcome suggested the software designers concerned to optimize their software design for architecture stability to evade via refactoring techniques that disturb the classes hierarchy. In its place they can usage those techniques that only affect method levels. Nevertheless, the researcher did not study the package as the basic part of software architectures in its place attentive on the classes level. Consequently, the work was incomplete to examine the outcome of refactoring approaches at acceptable/middle grain level that is fields method and classes.

Ebad et al. (2015) [6] the researchers present a novel software architecture stability metrics that quantity inter package call. This study hypothetically validated ASM via a number of protuberant mathematical properties. Researchers also authenticated the

metrics via 2 open source project, “JHot-Draw” and “abstract window” toolkits. Evaluations of the ASM were exposed to be dependable with the line of codes variations across the release in the 2 project. Beyond Compare is used to assess the variations at the source code levels for these 2 project. For instance, research scholars can use ASM to examine the correlations among module cohesion and module coupling on software architectural stability. The objective would be to usage such forecasters initially throughout the source code to forecast the stability of the software architecture. Regrettably we did not have access to information that would carry out such a study.

Mamdouh Alenezi (2016) paves the means for investigators to start examining way to evaluate software architectural quality attribute. Assessment of all these software qualities is important for this sub area of software engineering. This effort discovers “stability” and “understandability” of software architectures. The meaning and significance of software architectures were debated. In what way to assess these measurements were also broadly presented in this study. Software stability was given additional emphasis for their key importance and impact on software. Future guidelines contain devising new metrics to quantity both software architecture stability.

Our literature survey shows that though there are many approaches to quantifying stability in overall cycles of SDLC and in evolution software but the approach of design level object-oriented metrics, to quantify stability of class diagrams still needs to be quantified as a prerequisite for estimating stability model. It can be inferred from the extensive literature survey on stability that though many approaches on

defining stability have been given but there exists no complete model to measure stability of Object Oriented software at design phase of SDLC.

2.6 SIGNIFICANT CONCLUSIONS

The significant conclusions that can be drawn from the above detailed literature review is:

- Maintainability though an extremely important attribute of software quality, is desirable by all organizations but there is still no explicit focus in the industry on it in the development of projects. Appropriate processes, guidelines and tools needed for maintainability evaluation are still not available.
- Measuring maintainability in the early phase of SDLC like the design phase will help improve the design of software, which in turn will improve the quality of software. Maintainability is essentially a design issue and thus need to be handled at design phase.
- To get consistent and accurate measures of maintainability for Object Oriented software, it is essential to identify factors affecting maintainability. Defining a universally acceptable set of maintainability factors is not possible, but by extensive literature survey we have identified a key set of design attributes for maintainability measurement.
- Very few earlier approaches to measure software maintainability were validated empirically. A more organized understanding of maintainability measurement is yet to be evolved.

2.7 SUMMARY

The above literature survey shows that few models for measuring maintainability at design phase have been developed. All of these models show varying accuracy in measuring maintainability of Object Oriented software .Few of these models have been proposed but not validated. Out of these the model provided by Rizvi et. al.(2010) has shown better results than the rest and the authors have also validated their model. Rajendra et.al (2015) has also developed a model for measuring maintainability with good accuracy but they have not validated their model. A survey of the relevant literature also indicated that more work is done at measuring maintainability at the later stages of SDLC.

Having maintainability models at an early stage is thus still needed. To do so we have followed the systematic approach of identifying sub factors of maintainability and indentifying design properties and metrics of Object Oriented systems for estimating these sub factors. We in our work propose to develop and validate models for each of these sub factors and then maintainability measurement model for design phase for Object Oriented systems and also provide comparative study with relevant work.

In the next chapter we develop a model for the first internal attribute of maintainability i.e. changeability.

CHAPTER 3

CHANGEABILITY EVALUTION MODEL (CEM^{OOD})

3.1 INTRODUCTION

Changeability as given by ISO-9126 can be defined as the ability of the software to support a required or specific modification that needs to be implemented. Within the constraints of requirement specifications, it facilitates the designers with continuous evaluation and ratifications in the maintenance of software.

Chaumun et.al. (2002) in their work state that changeability is a significant key characteristic for maintainability analysis and evaluation for deliverance of good quality and cost effective maintainable software.

The constitution of ISO/IEC 9126 (2001) quality model proposed by software improvement group (SIG) used a three layered approach to indicate steps for developing quality models. Rongviriyapanish et.al. (2016) described that firstly quality sub attributes needs to be identified, secondly the system properties associated with them should be defined and lastly the source code measures were identified. In this context changeability, can be measured in terms of system properties like inheritance, encapsulation, coupling, polymorphism as given by Bagheri et. al. (2011), Kanaellopoulos et. al.(2008) and many more as described in the table 3.1 below:

Table 3.1 Changeability and related Object Oriented design factors

Design Properties ➔ **Author/Study ⬇ Changeability**	**Encapsulation**	**Coupling**	**Inheritance**	**Polymorphism**
Li &Offutt (1996)	**X**		**X**	**X**
Godin et. al (2000)		**X**		
Arisholm et. al (2000)		**X**		
A Chaumun et. al (2002)	**X**			**X**
Heitlager et. al. (2007)		**X**		**X**
Riaz et.al (2009)	**X**	**X**		
Dubey et.al (2011)	**X**	**X**	**X**	**X**
A Hincheeranan (2012)	**X**	**X**	**X**	**X**
Y Aylew et. al (2013)		**X**		
Al Dallal et. al (2013)		**X**		
Malhotra et. al (2013)	**X**	**X**	**X**	
Ankita et. al. (2014)	**X**	**X**	**X**	**X**

Few changeability models have been proposed based on different quality attributes and code metrics of Object Oriented systems as discussed in by Li and Offutt (1996), Godin et. al (2000), Arisholm et. al (2000), A Chaumun et. al (2002),Heitlager et. al. (2007),Riaz et.al (2009) ,Dubey et.al (2011),A Hincheeranan (2012),Y Aylew et. al (2013),Al Dallal et. al (2013) and Malhotra et. al (2013).

These models were proposed at different phases of SDLC. However, none of the models designed at the design phase have been empirically validated .This chapter

here highlights the importance of changeability broadly and also as an important contributor of software maintainability. In this chapter, a correlation between the major attributes of object oriented design viz. encapsulation, coupling, inheritance, polymorphism and changeability has been ascertained. A changeability evaluation model using multiple linear regression has been proposed for object oriented design. Finally, the validation of the proposed changeability evaluation model is made known by means of experimental runs using data from real life projects and the results show that the model is highly significant.

This rest of the chapter is structured into subsection as follows: Section 3.2 describes and establishes relationship between Changeability and Object Oriented Design Properties. In Section 3.3 we develop a model for evaluating changeability called (CEM^{OOD}). Section 3.4 shows the Statistical Significance between Changeability and Design Characteristics of Object Oriented Software. In Section 3.5 we have empirically validated our Changeability Evaluation Model (CEM^{OOD}) by comparing calculated values of changeability to actual values of changeability received from real life projects. Section 3.6 gives a Summary of the chapter.

Our next section establishes the relationship between the considered design properties of object oriented software and quality attribute changeability.

3.2 ESTABLISHING RELATIONSHIP BETWEEN CHANGEABILITY AND OBJECT ORIENTED DESIGN PROPERTIES

A wide-ranging review of object oriented design and development was done in Chapter 2 - Literature review and from studies done by McCall et.al. (1977), Land (2002), Dubey et.al(2012), Chaumun et. al. (2002), Ayalew and Mugni(2013),

Heitlager et. al. (2007), Dallal(2013), Elish and Elish(2009), Kiewkanya et. al. (2004), Genero et.al (2003) , Li and Ottfutt(1996) and many more, to develop a foundation to establish a relation between design properties and one of the quality attributes i.e. changeability. In view of this fact, a relation figure is proposed between the major properties of object oriented design and changeability as shown in Fig. 3.1. The mapping puts in place a relative impact correlation between changeability, object oriented design properties and the related design metrics. Out of the five major object oriented properties viz. encapsulation, coupling, inheritance, polymorphism and cohesion, we have used only four as some researchers like Kabaili et.al (2001) concluded that cohesion metrics is not a good indicator of changeability.

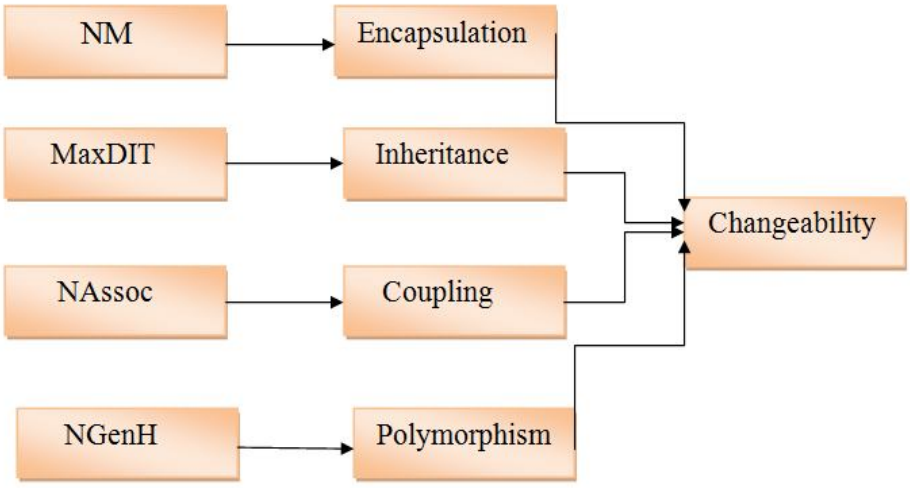

Fig. 3.1: Relation among Changeability, object oriented design properties and metrics.

3.3 CHANGEABILITY EVALUATION MODEL (CEM^{OOD})

In this section using the relationship established in Fig. 3.1, we propose a changeability evaluation model. We have implemented the method of multiple linear regression (MLR) to help us develop a model for Changeability.

Changeability=ß+ A1× Encapsulation + A2× Coupling + A3× Inheritance + A4× Polymorphism Eq. (3.1)

The datasets for developing and validating Changeability model is acquired from [Appendix I-Table I.1] that has been collected through the class diagrams. It includes a set of twenty (20) projects (indicated from P1 to P20) along with the value of metrics of each of these. Along with this, we have the actual mean values of different ratings by experts of Software Changeability for these projects. These are called 'Known Value' here in this chapter.

Table 3.2 shows the coefficients for Changeability evaluation model. We use the values we get from the unstandardized coefficients component of the table 3.2 to help develop the regression equation (3.2).

Table 3.2: Coefficients for Changeability Evaluation Model

Changeability Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std. Error	Beta		
1	(Constant)	8.477	2.906		2.917	.033
	Encapsulation	-.367	.452	-.140	-.813	.453
	Coupling	-1.530	.481	-.522	-3.182	.024
	Inheritance	-1.945	1.288	-.250	-1.510	.191
	Polymorphism	1.923	.512	.643	3.759	.013

Using SPSS to calculate the coefficients, the final changeability model that we derived is given below:

Changeability = 8.477 - 0.367× Encapsulation -1.53 × Coupling -1.945 × Inheritance + 1.923 ×Polymorphism Eq. (3.2)

The results of summarized model as shown in Table 3.3 are useful when calculating multiple regression. The coefficient determinant (R) with a value of 93.4% exhibits a very strong relation between the independent variables and the respective dependent variable. The value of this coefficient when squared i.e. R (square) from the table depicts the coefficient of determination. It refers to the ratio of total variance in changeability by all four independent variables. The value of both R^2 and adjusted R^2 is very encouraging. With the help of ANOVA analysis a significant regression equation was found **(F (4, 5) = 8.53, P<0.019) with R^2 of 0.934** from the model summary table.

Table 3.3: Changeability Evaluation Model Summary

Model(Summarized)				
Model	R	R Square	R Square (Adjusted)	Standard Error of the Estimate
1	.934a	.872	.770	.60873
a. Predictors: (Constant), Polymorphism, Coupling, Inheritance, Encapsulation				

The quality factor changeability increases with for each -0.367 units of encapsulation,-1.53 units of coupling,-1.945 units of inheritance and 1.923 units of

polymorphism. Hence encapsulation, inheritance, coupling and polymorphism are significant parameters of changeability.

The results of this trial experiment in assessment of changeability meet expectations and are very promising to attain maintainability index of object oriented design for small cost Software maintenance.

Our next section using a group of projects statistically establishes the correlation between design properties of Object Oriented software and Changeability.

3.4 STATISTICAL SIGNIFICANCE BETWEEN CHANGEABILITY AND DESIGN CHARACTERISTICS OF OBJECT ORIENTED SOFTWARE

The applications that are deployed in displaying the statistical significance among Changeability and object oriented design properties have been taken from [Appendix I-Table I.2]. We categorized the applications as: System G, System H and System I. All the systems are commercial software projects implemented in C++ with the number of classes and grouped as shown in Table 3.4. (Detail of the software Projects in each cluster is given in Appendix I- Table I.2)

Table 3.4: Group and Projects for proposed Evaluation model CEM^{OOD}

Group	Projects
System G	5
System H	5
System I	5

Table 3.5 gives the descriptive statistics for System G and Table 3.6 gives the correlation analysis for System G.

Table 3.5: Descriptive Statistics for System G

	Minimum	Maximum	Mean
Changeability	6.00	9.80	7.7000
Encapsulation	2.50	3.60	3.2800
Coupling	1.30	2.70	1.9400
Inheritance	.40	.80	.6000
Polymorphism	1.90	2.90	2.4000

Table 3.6: Correlation Analysis for System G

	Changeability	Encapsulation	Coupling	Inheritance	Polymorphism
Changeability	1	.906	.949	.868	.996
Encapsulation	.906	1	.927	.990	.882
Coupling	.949	.927	1	.887	.920
Inheritance	.868	.990	.887	1	.842
Polymorphism	.996	.882	.920	.842	1

Table 3.7: Descriptive Statistics for System H

	Minimum	Maximum	Mean
Changeability	5.90	9.80	8.1000
Encapsulation	2.50	3.80	3.2000
Coupling	1.30	2.40	1.7800
Inheritance	.40	.90	.6800
Polymorphism	1.90	2.90	2.4600

Table 3.8: Correlation Analysis for System H

	Changeability	Encapsulation	Coupling	Inheritance	Polymorphism
Changeability	1	.930	.929	.880	.975
Encapsulation	.930	1	.820	.890	.868
Coupling	.929	.820	1	.895	.870
Inheritance	.880	.890	.895	1	.753
Polymorphism	.975	.868	.870	.753	1

Table 3.9: Descriptive Statistics for System I

	Minimum	Maximum	Mean
Changeability	7.40	9.80	8.8000
Encapsulation	2.50	4.10	3.2400
Coupling	1.30	2.70	1.8200
Inheritance	.40	1.20	.8000
Polymorphism	1.80	2.90	2.3000

Table 3.10: Correlation Analysis for System I

	Changeability	Encapsulation	Coupling	Inheritance	Polymorphism
Changeability	1	.919	.907	.925	.950
Encapsulation	.919	1	.998	.869	.826
Coupling	.907	.998	1	.851	.806
Inheritance	.925	.869	.851	1	.982
Polymorphism	.950	.826	.806	.982	1

Table 3.11: Correlation Analysis Summary

	× Changeability Encapsulation	Changeability× Coupling	× Changeability Inheritance	× Changeability Polymorphism
System G	.906	.949	.868	.996
System H	.930	.929	.880	.975
System I	.919	.907	.925	.950

From the Table 3.11 after summary of the outcome of the correlation study it can be inferred that for Changeability evaluation model, there exists a high correlation between changeability and properties of Polymorphism, Coupling, Inheritance, Encapsulation for all the systems. The value of correlation 'r' ranges between ±1, positive value of 'r' in Table 3.11, indicates positive correlation between the two variables. The value of 'r' near to +1 specifies high measure of correlation between the two variables in above Table 3.11.

In the next section using a different subset of projects, empirical validation of Changeability Evaluation Model (CEM^{OOD}) is done.

3.5 EMPIRICAL VALIDATION OF CHANGEABILITY EVALUATION MODEL (CEM^{OOD})

The empirical validation is an important phase of proposed research. To verify our proposed model we empirically validate our model. This part of study focuses on the way the model proposed above is able to evaluate the Changeability calculated in object oriented software(s) at SDLC design stage. This experimental validation exists as a crucial step of proposed research to estimate Changeability Evaluation Model (CEM^{OOD}) for better and high level adaptability. Therefore, with this objective validation of the proposed Changeability Evaluation Model (CEM^{OOD}) is done using experimental tests.

In order to validate the developed Changeability Evaluation Model the projects viz. P1, P2, P3, P4, P5, P6, P7, P18, P19 and P20 were taken from Appendix I- Table I.1. The known Changeability value of the provided projects class diagram is shown in Table 3.12. Table 3.13 shows the corresponding known changeability ranks of the values of Table 3.12.

Table 3.12: Known Changeability Value

P1	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P18**	**P19**	**P20**
7.8	6.9	8.1	7.4	8.5	7.2	7	9.1	8.9	9.3

Table 3.13: Known Changeability Rank

P1	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P18**	**P19**	**P20**
5	1	6	4	7	3	2	9	8	10

Using the similar set of data for the given projects class diagram Changeability was calculated using proposed Changeability evaluation model and the results are shown in Table 3.14.

Table 3.14: Calculated Changeability Value Using Proposed Model CEM^{OOD}

P1	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P18**	**P19**	**P20**
2.5	2.2	3.8	3.4	3.7	1.4	0.4	4.4	6.7	5.8

Table 3.15: Calculate d Changeability Rank Using Proposed Model CEM^{OOD}

P1	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P18**	**P19**	**P20**
4	3	7	5	6	2	1	8	10	9

Charles Spearman's rank relation rs was used to test the significance of correlations calculated amidst Ranks of Changeability via proposed model and the ranks Known in it.

The 'r_s' was calculated using the formula given as under:

$$r_s = 1 - \frac{6\sum d^2}{n(n^2-1)} \qquad -1.0 \le r_s \le +1.0$$ Eq. (3.2)

'd' = difference that exists in Calculated Rank and Known Rank of Changeability.

'n' = total quantity of Projects taken in conducting tests.

Table 3.16: Computed Rank, Actual Rank and their Relation

Project(s)	P1	P2	P3	P4	P5	P6	P7	P18	P19	P20
Computed Ranks	4	3	7	5	6	2	1	8	10	9
Known Ranks	5	1	6	4	7	3	2	9	8	10
d^2	1	4	1	1	1	1	1	1	4	1
$\sum d^2$	16									
r_s Calculated	0.90303									
$r_s > \pm.781$	✓									

The correlation value among calculated Changeability ranks using proposed model CEM^{OOD} and known ranks is shown in Table 3.16 above. There appears to be a very strong positive correlation r_s value (+0.903) with a p=0.001 (99.9% statistical significance level) as shown by fig 3.2 scatter graph where data Set A represents the known changeability value and data set B are the calculated values by our changeability evaluation model CEM^{OOD}.

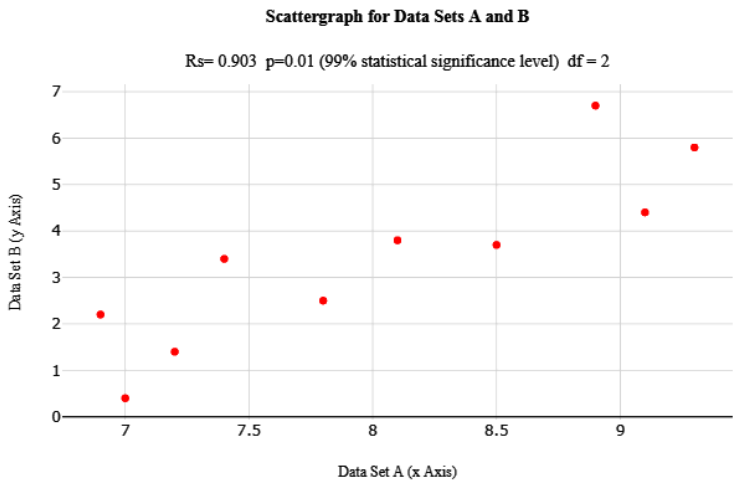

Fig 3.2. Graph showing Spearman's rank correlation between known and calculated values of changeability.

Correlation value r_s meets the expectations standard showing high confidence, i.e. of 99%.So we can say there is a strong correlation between the data set calculated from our changeability evaluation model (CEM^{OOD}) and actual values of changeability .

This study undoubtedly shows that the Changeability model is significant. In the end of this chapter we provide a comprehensive conclusion of our proposed model.

3.6 SUMMARY

This chapter displays the significance of Changeability and the correlation it has with the design properties of Object Oriented systems. These design properties are encapsulation, coupling, inheritance and polymorphism. The correlation was established using multiple linear regression formula and a Changeability Evaluation Model (CEM^{OOD}) is developed. The results obtained statistically confirm the significance and acceptability of the proposed model. The proposed model has been validated empirically via experimental test. The real-world validation of the Changeability model accomplishes that developed model is highly dependable, acceptable and significant. The chapter concludes that there is a high correlation between Changeability and design properties.

In the next chapter, we will discuss about Stability Evaluation Model.

CHAPTER 4

STABILITY EVALUATION MODEL SEM^{OOD}

4.1 INTRODUCTION

Stability can be defined as the responsiveness to change of a given system on the negative effects that may be triggered by these system changes [ISO-9126]. According to this quality assurance standard of ISO/IEC 9126, stability is well-defined as the point to which the software module, can avoid unpredicted effect from the modifications of the software. This is supported by the definition given by Black(2000). As discussed in literature review, Stability is an important key contributor for maintainability evaluation at design phase. Stability being significant software quality indicator, its correct assessment leads to improving the software maintenance process as defined by ISO-9126 (2001). Stability continually has a key impact in delivering maintainable and reliable software within an acceptable time and budget. Along with this, emphasizing on stability in early phase of software development cycle further simplifies maintenance process during maintenance phase and after implementation. The ability to only reengineer the required part in such a way that the rest of the software modules continue unchanged is what makes stability an important maintainability factor as discussed by Alshayeb and Oman (2011).

If the stability factor is not as per the desirable standard it cascades the impact of any modifications throughout the design, thereby increasing the possibility of generation of new errors and affecting software maintainability. This in turn results in increase in

costs and effort than the actual earlier estimated costs as discussed by Ebad and Ahmed (2015) and Raemaekers(2012).

The assessment of stability using design properties is more relevant and its justification indicates the valid influence of functional and structural information of object oriented design and development. L. Yu (2009) and Fayad (2002) discussed that regardless of the fact stability is vital and extremely noteworthy aspect for system development cycle it is poorly managed. Thus, we can say that the risks we face in software stability due to unexpected effects of modifications are a major concern and it affects the overall maintainability costs of the software.

In this chapter correlation between object oriented design properties and Stability has been established. A Stability Evaluation Model (SEM^{OOD}) has been proposed here for Object Oriented Design by using multiple linear regression. Consequently, the proposed model has been validated empirically using experimental runs.

The outline of this chapter is described as Section 4.2 describes the mapping between the quality attribute stability and object oriented design properties viz. Encapsulation, Coupling and inheritance. Section 4.3 explains the development of Stability Evaluation Model (SEM^{OOD}). Section 4.4 establishes the correlation and significance between different properties using statistical methods. Section 4.5 discusses empirical validation of our model. Section 4.6 summarizes the whole chapter.

4.2 MAPPING BETWEEN STABILITY AND DESIGN PROPERTIES

A broad analysis of object oriented design properties was discussed in Chapter 2 – Literature Review and from work done by researchers and standards like Heitleger

et.al. (2007), Yau and Chang(1984) , ISO (2001), Genero et. al. (2003), Alshayeb et. al (2011), Hagemeister and Oman (1992) and has been used to establish a relation between design properties and Stability. From this extensive research we were able to decide which metric or attribute is highly considerable to the development of our model. Along with the literature survey the data from software companies compiled by experts was also used as a directive for choosing these properties for the model. A correlation amongst object oriented design properties, design metrics and Stability as shown in Fig. 4.1.

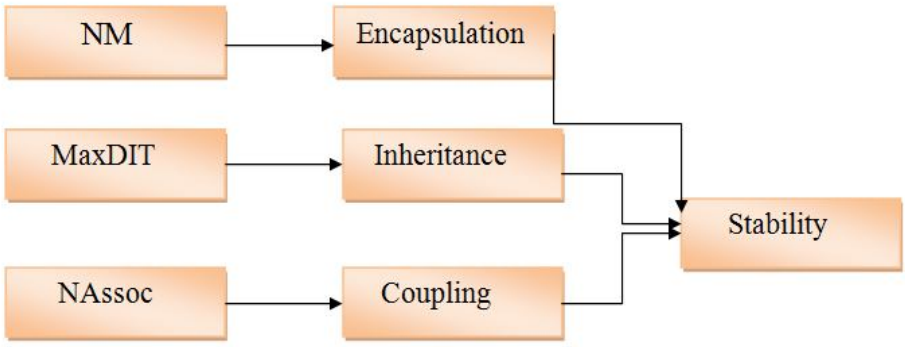

Fig. 4.1: Mapping among Stability, Object Oriented Design Properties

In the next section we discuss the proposed Stability Evaluation Model (SEM^{OOD}).

4.3 STABILITY EVALUATION MODEL (SEM^{OOD})

We have used the method of multiple linear regression to develop a measurement model for stability.

The data values for developing stability model and validating the developed model is acquired from Appendix I - Table I.1 that has been together from the class diagrams. It contains a group of twenty (20) class diagrams (designated from P1 up to P20)

laterally with the values of metrics of each of these. Along with this, we have the rating by expert of Software stability for these projects respectively. These are called "Known Values" here in this research work. For development of our model we have used values of projects P1 to P7 and P18 to P20.

$$\text{Stability} = ß + A1 \times \text{Encapsulation} + A2 \times \text{Coupling} + A3 \times \text{Inheritance} \quad \text{Eq. (4.1)}$$

The respective coefficients values are calculated via SPSS and a stability model is developed. Equation (4.1) signifies the relations amongst stability and the object-oriented design properties as evaluated.

Table 4.1 contains the coefficients for Stability Evaluation Model (SEM^{OOD}). The unstandardized coefficients from the table have been used as coefficients in the regression equation (4.2) to develop our model. In this table the Standardized Beta Coefficients specify the comparative measure of the contribution of each variable to the Stability model.

Table 4.1: Coefficients for Stability Evaluation Model

Coefficients[a]						
Model.		Unstandardized- Coefficients		Standardized- Coefficients	t	Sig.
		B	Std. Errors	Beta		
1	(Constant)	5.562	2.573		2.162	.119
	Encapsulation	-1.034	.319	-.750	-3.243	.048
	Coupling	.013	.306	.011	.041	.970
	Inheritance	1.006	.614	.442	1.638	.200
a. Dependent Variable: Stability						

Stability = 5.562 - 1.034 × Encapsulation + .013 × Coupling + 1.006 × Inheritance

Eq. (4.2)

The Model Summary Table 4.2 output is most valuable when performing multiple linear regressions. Capital R is the multiple correlation coefficients that tell us how powerfully the multiple independent variables are related to the dependent variable. R Square is also very high and gives us the coefficient of determination which further supports the correlation. A value of R close to 92% is obtained, thus from this table we can conclude that encapsulation, coupling and inheritance i.e. the independent variables are strongly correlated to the dependent variable viz. stability. With the help of ANOVA analysis a significant regression equation was found **(F (3, 6) = 4.826, P<0.049) with R^2 of 0.926** from the model summary table. The quality factor stability increases with for each -1.034 units of encapsulation, 0.013 units of coupling and 1.006 units of inheritance. Hence encapsulation, inheritance and coupling are significant parameters of stability.

Table 4.2: Stability Evaluation Model Summary

Model Summary				
Model.	R	R-Square	Adjusted-R Square	Standard Error of the Estimate
1	0.926a	0.857	0.713	0.51778
a. Predictors:- (Constant), Inheritance, Encapsulation, Coupling				

In the next section we establish the statistical significance between different considered variables.

4.4 STATISTICAL SIGNIFICANCE BETWEEN STABILITY AND OBJECT ORIENTED DESIGN PROPERTIES

To justify the correlation of Stability with object oriented design properties, statistical test are performed. The applications that are used to perform statistical test have been taken from Appendix I-Table I.1. We labeled the applications as: System D, System E and System F. Object oriented technology is applied in all of these commercial software[s] with the number of classes as shown in Table 4.3. (Detail of the Projects in each group is given in Appendix I-Table I.3)

Table 4.3: Group and Projects for Proposed SEM^{OOD}

Group	**Projects**
System D	4
System E	4
System F	4

Table 4.4: Detailed Statistics for System D

	Minimum	Maximum	Mean
Stability	6.80	9.30	7.8500
Encapsulation	2.90	3.60	3.3750
Coupling	1.30	2.70	2.1500
Inheritance	.50	.90	.6750

Table 4.5: Correlation Analysis for System D

	Stability	Encapsulation	Coupling	Inheritance
Stability	1	.979	.932	.973
Encapsulation	.979	1	.967	.926
Coupling	.932	.967	1	.917
Inheritance	.973	.926	.917	1

Table 4.6: Descriptive Statistics for System E

	Minimum	Maximum	Mean
Stability	7.00	8.60	7.8250
Encapsulation	2.50	3.50	3.2000
Coupling	1.30	2.20	1.7500
Inheritance	.40	.80	.6250

Table 4.7: Correlation Analysis for System E

	Stability	Encapsulation	Coupling	Inheritance
Stability	1	.850	.878	.881
Encapsulation	.850	1	.868	.700
Coupling	.878	.868	1	.947
Inheritance	.881	.700	.947	1

Table 4.8: Descriptive Statistics for System F

	Minimum	Maximum	Mean
Stability	6.90	8.50	7.6750
Encapsulation	3.50	4.10	3.7250
Coupling	1.30	2.70	1.9750
Inheritance	.60	1.00	.7500

Table 4.9: Correlation Analysis for System F

	Stability	Encapsulation	Coupling	Inheritance
Stability	1	.868	.917	.954
Encapsulation	.868	1	.931	.904
Coupling	.917	.931	1	.992
Inheritance	.954	.904	.992	1

Table 4.10: Correlation Analysis summary

	Stability x Encapsulation	**Stability x Coupling**	**Stability x Inheritance**
System D	.979	.932	.973
System E	.850	.878	.881
System F	.868	.917	.954

Table 4.10 displays the results obtained by applying correlation analysis on stability Evaluation Model (SEM^{OOD}). For the considered three systems viz. D, E and F, the properties of encapsulation, coupling and inheritance are significantly well correlated with stability. The values of r in this table are close to +1 which signifies that there exists a positive and high degree of correlation between the two variables.

No study or development of a model is complete unless we apply some methods of validating our model. The next section in details validates our model using experimental tryouts.

4.5 EMPIRICAL VALIDATION OF STABILITY EVALUATION MODEL

No study or development of a model is complete without proper validation of the proposed model. This requirement is for all engineering disciplines including software and computer engineering.

In this section we analyze how well our proposed model (SEM^{OOD}) estimates the stability values as compared to the known of each metric collected from the ten projects viz. (P8 to P17). The mean values of different rating by experts of software stability for these projects are used for comparison using Charles spearman's rank relation i.e in order to validate the, proposed stability evaluation Model the projects viz. P8 to P17 were taken from Appendix I- Table I.1.

An experimental validation of the developed Stability evaluation model SEM^{OOD} (equation 4.2) has been carried out with the Appendix I-Table I.1.

The known Stability value and rank for the given projects class diagram is publicized in Tables 4.11 and 4.12.

Table 4.11. Known Stability Value

P8	P9	P10	P11	P12	P13	P14	P15	P16	P17
6.7	7.2	7.9	7.0	8.3	6.8	8.6	7.4	9.3	6.9

Table 4.12. Known Stability Rank

P8	P9	P10	P11	P12	P13	P14	P15	P16	P17
1	5	7	4	8	2	9	6	10	3

Using the similar group of data for the given project class diagram Stability was calculated using developed Stability evaluation model and the results are publicized in Table 4.13.

Table 4.13 Calculated Stability Value Using Proposed Model SEM^{OOD}

P8	P9	P10	P11	P12	P13	P14	P15	P16	P17
2.1534	2.6637	3.2988	2.5635	2.6758	2.3777	3.8065	2.573	3.5	2.3591

Table 4.14 Calculated Stability Rank Using Proposed Model SEM^{OOD}

P8	P9	P10	P11	P12	P13	P14	P15	P16	P17
1	6	8	4	7	3	10	5	9	2

Table 4.15: Computed Rank, Actual Rank and their Relation

Projects➔ Stability Ranking⇩	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17
Computed Ranks	1	6	8	4	7	3	10	5	9	2
Known Ranks	1	5	7	4	8	2	9	6	10	3
d^2	0	1	1	0	1	1	1	1	1	1
$\sum d^2$	**8**									
$r_{s\ Calculated}$	**0.951515**									
$r_s > \pm .781$	✓									

(Charles Spearman's) Rank relations "r_s" was used to assess the importance of correlations between "calculated Ranks of Stability" via proposed model and its "Known Ranks".

The 'r_s' was calculated via the formulation specified as under:

Spearman's- Coefficient of Correlation (rs) –

$$r_s = 1 - \frac{6\Sigma d^2}{n\,(n^2-1)} \qquad -1.0 \leq r_s \leq +1.0 \qquad \text{Eq. (4.3)}$$

'd'=difference amongst "Calculated Rank" and "Known Rank" of Stability and 'n' = Total Projects used in the research.

The correlation value amongst calculated "Stability ranks" using developed model SEM^{OOD} and "known ranks" are shown in Table 4.15. Correlation value r_s unquestionably show that the Stability model is very important and highly significant with a confidence level of 99%. There appears to be a very strong positive correlation r_s value (+0.9515) with a p=0.001 (99.9% statistical significance level) as shown by the graph in fig 4.2 where data set A is known values of stability and data set b represents calculated values of stability using SEM^{OOD}.

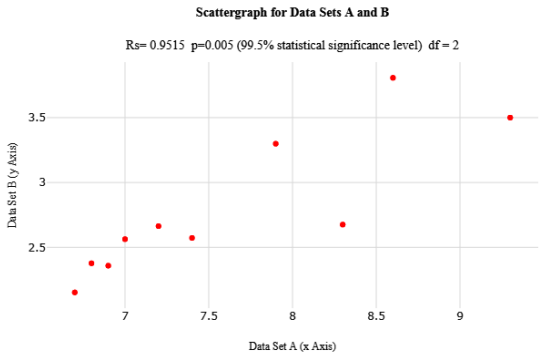

Fig 4.2. Graph showing Spearman's rank correlation between known and calculated values of stability.

Thus we can infer that there is a strong correlation between the data set calculated from our stability evaluation model (SEM^{OOD}) and actual values of stability.

It can thus be inferred that with no loss of generality our proposed Stability Evaluation Model (SEM^{OOD}) is reliable and relevant in the current perspective.

The next section gives a summary of the above sections.

4.6 SUMMARY

This chapter shows the significance of Stability and its relationship with object oriented design properties viz. encapsulation, inheritance and coupling. Stability is one of the most noteworthy factors for evaluating maintainability of object oriented design. This chapter proved the significances of Stability and its relationship with various object oriented design properties. Further, study developed a Stability evaluation model with correlation establishment among Stability and object oriented design properties. Subsequently, developed model was validated empirically by means of investigational tryout. The applied authentication on the stability model accomplishes that the stability model is most highly significant. The work concludes that there is a high correlation between Stability and design properties.

In the next chapter, we will discuss about Maintainability.

CHAPTER 5

MAINTAINABILITY EVALUATION MODEL MM^{OOD}

5.1 INTRODUCTION

According to IEEE standard maintainability can be defined in terms of how easy it is for software to be repaired, provide improved performance and adapt to changing environments. As given in early research by Somerville et. al. (1992), maintainability is a key attribute for well designed software. Hayes and Zaho (2005), Aggarwal et. al. (2005), Misra (2005) and Zhou and Leung(2007) in their study have shown that software maintainability can be enhanced by controlling Object Oriented design properties like coupling, cohesion, encapsulation, polymorphism and inheritance.

Evaluating maintainability provides guidelines that help in significant decrease in terms of cost and time in the various stages of software development and components, quality control and quality assurance as discussed in the research work of Aggarwal et.al. (2005), Dallal and Jehad (2013), Sommerville (1992), Boehm et. al. (1978), Kiewkanya et. al. (2004), Lee and Chang (2014), McCall et. al. (1977) and Grady et. al. (1987). The calculation of maintainability at a later stage of SDLC as discussed by DiLucca et.al. (2004), Li et. al. (2006) and Elish (2009) often results in delayed reception of crucial information therefore causing a holdup in response and implementation about changes in software design. This results in an increase in terms of cost and additional work. In their study, Singh et.al. (2014) and Dubey et.al. (2012) showed that a preference to transform the design so as to recover maintainability after

the coding may turn out to be more costly and prone to errors. Consequently, early estimation of maintainability in the software development cycle may improve design quality and decrease maintenance efforts and cost as reviewed by Hincheeranan and Rivepiboon (2012), Ping(2010), Boehm et. al. (1978), Aggarwal et. al. (2006), Malhotra and Chug(2016) .

Many models for maintainability have been proposed from time to time for measuring maintainability as shown in Table 2.1 in chapter2. These models have been defined at various phases of SDLC. Approaches used for defining the measures for these maintainability models along with their limitations and shortcomings are also discussed in detail in Chapter 2 here. Therefore we can conclude for researchers, quality controllers and programmers continuous effort for planning and evaluation of maintainability in design phase of the software development life cycle is thus of inevitable importance.

Taking these facts into consideration our research work is thus focused on evaluation of maintainability at design stage to deliver quality oriented maintainable software. Also after relevant study the quality characteristic of maintainability has been refined into its important sub-characteristics that have significant contribution in maintainability evaluation at design phase of software development cycle. From the analysis of study done by Hordijk et. al. (2005), Khan and Mustafa(2004), Genero et.al. (2003)Rizvi and Khan(2010) Maurya and Shankar(2012) and all others mentioned in Table 2.2 of literature survey ,it can be concluded that Changeability and Stability are the two most significant factors affecting software maintainability evaluation.

This chapter proposes a Maintainability Evaluation Model called MM^{OOD} that works at design phase of system development life cycle using multiple linear regression method. Furthermore, statistical test is performed to justify the correlation of Maintainability with its key contributors Changeability and Stability. The developed model has been validated using empirical tryout. In conclusion, it includes comparative analysis between our proposed maintainability model MM^{OOD} and related existing model and summary.

5.2 MAINTAINABILITY EVALUATION MODEL DEVELOPMENT

The steps to develop Maintainability Evaluation Model MM^{OOD} are as explained below.

- Identification of key factors of object oriented software that have significant and positive influence on maintainability evaluation at design phase of software development life cycle as discussed in Chapter 2 here. The factors were identified as Changeability and Stability.
- Identification of Object oriented design properties related to Changeability Viz. Polymorphism Encapsulation, Inheritance and Coupling. Stability viz. Encapsulation , Inheritance and Coupling were identified as discussed in chapter 2 here.
- Development of changeability evaluation model (CEM^{OOD}) in terms of Object Oriented properties as shown in chapter 3 here.
- Development of stability evaluation model (SEM^{OOD}) in terms of Object Oriented properties as shown in Chapter 4 here.
- Development of maintainability evaluation model (MM^{OOD}) in terms of changeability and stability is presented in this paper.

Taking into consideration the association between the maintainability factors and design properties of Object oriented software, comparative importance of individual factors that have major influence on software maintainability at design phase is adjusted proportionally (Fig. 5.1).

Fig 5.1 shows the relationship between maintainability, design properties viz. Changeability and Stability and design metrics.

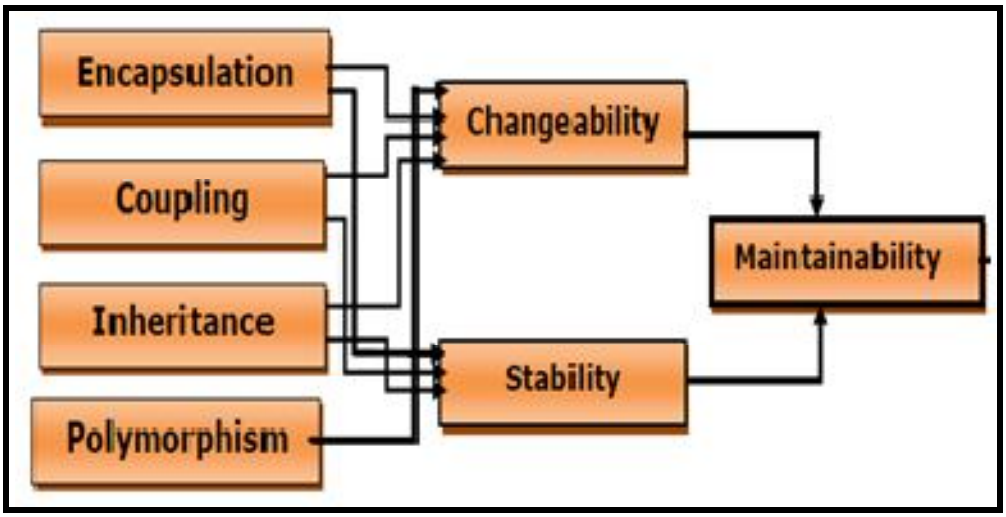

Fig 5.1: Relating Design Properties with Key Factors of Maintainability

In order to develop a model for Maintainability Evaluation, a Multiple Linear Regression Technique has been used to get the coefficients as explained by Gupta (1983). This system gives the association among dependent variable and multiple independent variables. Multivariate linear

equation is given below, in Eq. (5.1) which is as follows.

$$Y=a0+a1X1+a2X2+a3X3+\text{-- -- -- -+an Xn} \qquad \text{Eq. (5.1)}$$

Where,

• Y: Dependent Variable.

• X1, X2, X3--------Xn: Independent Variables.

• a1, a2, a3--------an.: Respective Coefficients.

• a0: Intercept.

The following Multiple Linear Regression equation has been established:

Maintainability = $\alpha_0 + ß_1 \times$ Changeability + $ß_2 \times$ Stability Eq. (5.2)

To develop and validate this model the data related to 20 projects was collected from the Industry. The projects were numbered P1 to P20. This data contains the evaluated maintainability value through ten Industry experts named as Evaluators. For maintainability estimation model and to determine the coefficients of Eq. (5.2), the data (P1, P2, P3, P4, P5, P6, P7, P8, P9, and P10) as shown in Appendix II-Table II.1, from industry was used and for this we considered the maintainability value given by Evaluator 1. Using SPSS, correlation coefficients are calculated and proposed model MM^{OOD} for Maintainability Evaluation is accordingly formulated as specified below in Eq. (5.3).

Maintainability = 4.467+ .190× Changeability -.112× Stability

Eq. (5.3)

Table 5.1 displays the coefficients value for Maintainability Evaluation Model MM^{OOD}. The un-standardized coefficients part of the result gives us the values that we want in order to write the Eq. (5.3). The Standardized Beta Coefficients give a measure of the influence of each variable to Maintainability.

Table 5.1: Coefficients values for Proposed Maintainability Evaluation Model

Model	Unstandardized Coefficients	Standardized Coefficients	t	Sig.

		B	Std. Error	Beta		
1	(Constant)	4.467	.513		8.700	.000
	Changeability	.190	.042	.783	4.532	.003
	Stability	-.112	.091	-.213	-1.232	.258
a. Dependent Variable: Maintainability						

The Maintainability Evaluation Model summary results as shown in Table 5.2 are highly significant when performing multiple regression. Capital R is the correlation coefficient that shows correlation between the multiple independent variables and the dependent variable. The obtained value of R = 0.949 shows a strong correlation between the considered independent variables, changeability and stability with maintainability. R Square provides the coefficient of determination. It refers to the ratio of total variance in changeability by all four independent variables. The value of both R^2 and adjusted R^2 are also very encouraging. With the help of Anova analysis a significant regression equation was found **(F (2, 7) = 31.550, P<0.000) with R^2 of 0.949** from the model summary table. The quality factor maintainability increases with for each .190 units of changeability and -.112 units of stability. Hence changeability and stability are significant parameters of maintainability.

Table 5.2: Proposed Maintainability Evaluation Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.949[a]	.900	.872	.23408
a. Predictors: (Constant), Stability, Changeability				

The following section establishes a statistical correlation between our two quality sub-attributes viz. changeability and stability with our quality attribute maintainability.

5.3 STATISTICAL SIGNIFICANCE BETWEEN MAINTAINABILITY, CHANGEABILITY AND STABILITY

To justify the correlation of dependent variable maintainability with independent variables changeability and stability, statistical tests are accomplished. The commercial applications that are used to complete statistical assessment are presented in Appendix II-Table II.2. We grouped the applications as: System A (with-3 projects), System B (with-3 projects) and System C (with-3 projects). All the systems are commercial software applications, implemented using object oriented technology.

Table 5.3 provides the descriptive statistics for System-A and Table 5.4 provides the correlation analysis for System-A.

Table 5.3: Descriptive Statistics for System A

	Minimum	Maximum	Mean
Maintainability	4.90	6.90	5.7666
Changeability	6.00	9.80	8.2000
Stability	6.80	8.30	7.5000

Table5.4: Correlation Analysis for System A

	Maintainability	Changeability	Stability
Maintainability	1	.983	.993
Changeability	.983	1	.954
Stability	.993	.954	1

Table 5.5 provides the descriptive statistics for System-B and Table 5.6 provides the correlation analysis for System-B.

Table 5.5: Descriptive Statistics for System B

	Minimum	Maximum	Mean
Maintainability	5.10	7.30	5.9000
Changeability	7.40	8.80	8.2.666
Stability	6.90	8.60	7.5000

Table 5.6: Correlation Analysis for System B

	Maintainability	Changeability	Stability
Maintainability	1	.933	.999
Changeability	.933	1	.951
Stability	.999	.951	1

Table 5.7 provides the descriptive statistics for System-C and Table 5.8 provides the correlation analysis for System-C.

Table 5.7: Descriptive Statistics for System C

	Minimum	Maximum	Mean
Maintainability	4.20	4.90	4.4666
Changeability	5.90	9.10	7.9666
Stability	8.50	9.60	9.1333

Table 5.8: Correlation Analysis for System C

	Maintainability	Changeability	Stability
Maintainability	1	.971	.935
Changeability	.971	1	.822
Stability	.935	.822	1

Table 5.9 summarizes the outcome of the correlations analysis for Maintainability evaluation model, which shows that for all the systems, both Changeability and

Stability are highly associated with Maintainability. The positive value of correlation analysis shows positive correlation between the maintainability, changeability and stability. The values close to 1 specify high degree of correlation between the dependent and independent variables Table 5.9 summarizes the outcome of the correlation analysis for Maintainability evaluation model, which displays that for all the systems, both Changeability and Stability are highly correlated with Maintainability. The positive value of correlation shows positive correlation between the variables. The values close to 1 specify high degree of correlation between the maintainability, changeability and stability.

Table 5.9: Correlation Analysis Summary

	Maintainability × Changeability	**Maintainability × Stability**
System A	.983	.993
System B	.933	.999
System C	.971	.935

The following section shows empirical validation for our proposed model MM^{OOD}.

5.4 EMPIRICAL VALIDATION OF MAINTAINABILITY EVALUATION MODEL MM^{OOD}

A crucial stage in every research is to empirically validate it. Based on this requirement we here provide a realistic validation of our proposed maintainability evaluation model using the sample runs. In order to validate proposed maintainability

evaluation model, the projects P11,P12, P13, P14, P15, P16, P17, P18, P19 and P20 as shown in Appendix II-Table II.2 are used to perform statistical test. To validate the model the maintainability values given by evaluators 1 is considered. During tryouts, maintainability value of the projects has been evaluated using the developed model MM^{OOD}. After this the maintainability ranks have been calculated and compared with the known ranks using Charles Spearman's Coefficient of Correlation. The known Maintainability values and ranks for the given projects class diagram is shown in Table 5.10 and Table 5.11.

The known Maintainability values for the given projects with 10-20 class diagram in each project is shown in Table 5.10.

Table 5.10: Known Maintainability Values

P11	P12	P13	P14	P15	P16	P17	P18	P19	P20
5.3	4.8	5.0	5.1	5.5	4.9	5.4	4.2	4.3	5.3

Table 5.11: Known Maintainability Ranks

P11	P12	P13	P14	P15	P16	P17	P18	P19	P20
8	3	5	6	10	4	9	1	2	7

Using the similar group of data for the given projects maintainability values was calculated using proposed maintainability evaluation model and the results are shown in Table 5.12.

Table 5.12: Calculated Maintainability Values by Proposed Model MM^{OOD}

P11	P12	P13	P14	P15	P16	P17	P18	P19	P20
5.64	5.33	5.50	5.65	6.01	5.18	5.78	5.03	5.30	5.44

Table 5.13: Calculated Maintainability Ranks by Proposed Model MM^{OOD}

P11	P12	P13	P14	P15	P16	P17	P18	P19	P20
7	4	6	8	10	2	9	1	3	5

Table 5.14: Calculated Ranks, Known Ranks and their Relations

Projects ➔	P1	P1	P1	P1	P1	P1	P1	P1	P1	P20
CalculatedRanks	7	4	6	8	10	2	9	1	3	5
Known Ranks	8	3	5	6	10	4	9	1	2	7
$\mathbf{d^2}$	1	1	1	4	0	4	0	0	1	4
$\sum d^2$	16									
$\boldsymbol{r_s}$	**0.90303**									
$\boldsymbol{r_s > \pm .781}$	✓									

Charles Spearman's Coefficient of Correlation (rank relation) r_s was used to check the significance of correlation between calculated ranks of Maintainability using the proposed model and its known ranks.

The correlation values between rank through the proposed model and known rank are shown in Table 5.14. Correlation value $\boldsymbol{r_s}$ of 0.903 clearly show that the model is significant. The correlation is up to standard with high degree of confidence, i.e. up to 99.8% as shown in fig.5.2 where data set A represents known values of maintainability and data set B represents calculated values of maintainability using our maintainability evaluation model MM^{OOD} model. Therefore, we can conclude without any loss of generality that Maintainability Evaluation Model is highly reliable and significant.

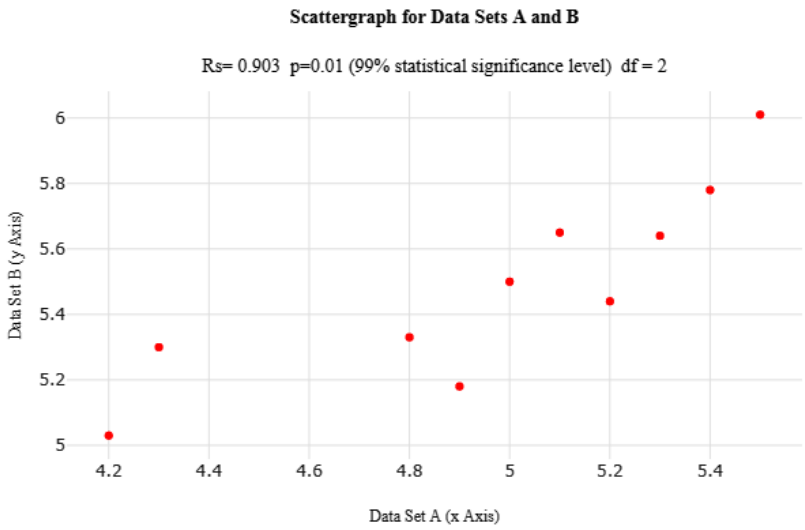

Fig. 5.2 Graph showing Spearman's rank correlation between known and calculated values of maintainability.

In our next section we compare our proposed model MM^{OOD} with two relevant models.

5.5 COMPARATIVE ANALYSIS BETWEEN MM^{OOD} AND RELATED EXISTING MODEL

To perform comparative study between proposed model (MM^{OOD}) and related existing model, the projects P11, P12, P13, P14, P15, P16, P17, P18, P19 and P20 are used as shown in Appendix II-Table II.1. The data contains the maintainability values given by ten individual experts (here termed as evaluators). Therefore, the Charles spearman's coefficient value has been calculated in comparison with ten different evaluators Table 5.15: Rank Correlation Comparison between: Proposed Model MMOOD to models proposed by Rajendra Singh et al. and MEM^{OOD}.

Table 5.15: Rank Correlation Comparison between Proposed Model MM^{OOD} and Rajendra et al.

Evaluators	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
$\sum d^2$ with Proposed Model-MM^{OOD}	**16**	**18**	**14**	**20**	**24**	**28**	**20**	**24**	**26**	**28**
$\sum d^2$ with model proposed by Rajendra et al.	76	74	54	54	94	78	62	130	90	60
$\sum d^2$ with model MEM^{OOD}	88	106	86	78	144	142	68	114	94	124
r_s with Proposed Model-MM^{OOD}	**0.903**	**0.891**	**0.915**	**0.879**	**0.854**	**0.830**	**0.879**	**0.854**	**0.842**	**0.830**
r_s with model proposed by Rajendra et al.	0.539	0.552	0.673	0.673	0.430	0.527	0.624	0.212	0.455	0.636
r_s with model MEM^{OOD}	0.467	0.357	0.479	0.527	0.127	0.139	0.589	0.309	0.430	0.248

It is obvious from Table 5.15 that r_s values with the assistance of developed Maintainability Evaluation Model MM^{OOD} are bigger than both related existing model in above Table 5.15. This specifies that the proposed model MM^{OOD} has an improved correlation with the maintainability ranks given by the experts and is able to evaluate maintainability more correctly and appropriately. Therefore, it is clear and evident from the empirical validation and comparative study that the developed model is more significant and better than the both related existing model.

5.6 SUMMARY

In this chapter, software maintainability key contributors are identified and their influence on maintainability evaluation and enhancement at design phase has been investigated. 'Changeability and Stability, two of the main contributors affecting object oriented design and development have been taken into consideration for model development. Considering both the major factors, a maintainability evaluation model for object oriented design has been developed (MM^{OOD}), and the statistical inferences are validated for high level better acceptability. Afterward comparative study is doing between proposed MM^{OOD} and other related existing models. Comparative study outcome specifies that the proposed model MM^{OOD} has a better relationship with the maintainability ranks given by the experts and is able to evaluate maintainability more correctly. Therefore, proposed maintainability evaluation model for object oriented software design is very trustworthy and associated with object oriented design properties. Maintainability evaluation model has been validated empirically using experimental tryout. The practical validation on the maintainability model accomplishes that developed model is extremely reliable, acceptable and consistent.

The next chapter describes the conclusion and future direction.

CHAPTER 6

CONCLUSION AND FUTURE WORK

This final chapter summarizes the contributions of the dissertation and suggests the directions for extension of our work.

From the detailed discussion of our motivation for this work it can now be well understood that effectively measuring maintainability improves quality of software positively on the scale of cost benefit analysis. Quantitatively assessing the maintainability to improve software i.e. in the design phase of SDLC on Object Oriented design characteristics provides us huge benefits in present scenario of Object Oriented software development. Organizations can thus use the proactive strategy to design their software products with maintainability as key design criteria. We now summarize the contributions of this work.

6.1 CONCLUSION

We have focused on developing and designing an efficient method for assessment of maintainability at an early stage of different phases of software development life cycle for Object Oriented systems. To develop this model the work proposes maintainability as a function of two internal attributes given by ISO-9126 as sub factors of maintainability viz. changeability and stability.

We have made mainly three contributions during the present course of study in

addition to many macro level direct or indirect findings. The contributions are

the three models for changeability , stability, and maintainability namely CEM^{OOD}, SEM^{OOD} and $MM^{OOD.}$ These are explained below.

6.1.1 Maintainability Factor Identification: We have used the literature review for maintainability factor identification and development of maintainability measurement model as explained below.

In fact, there are many factors affecting software maintainability. We identified two key factors viz. changeability and stability having a significant contribution in measuring software maintainability at design phase.

6.1.2 First Contribution: Changeability Evaluation Model (CEM^{OOD})

We have developed a Changeability Evaluation Model (CEM^{OOD}) for object

Oriented design and established the statistical correlation between changeability and design properties viz. encapsulation, inheritance, coupling and polymorphism with the help of multiple linear regressions. Finally, empirical validation of the changeability evaluation model was performed using commercial software applications.

6.1.3 Second Contribution: Stability Evaluation Model (SEM^{OOD})

We have developed Stability Evaluation Model (SEM^{OOD}) for object oriented design and established the statistical correlation between stability and design properties viz. encapsulation, coupling and inheritance with the help of multiple linear regression. An experiential corroboration of the Stability Evaluation Model is also done using commercial software applications.

6.1.4 Third Contribution: Maintainability Measurement Model (MM^{OOD})

In order to strengthen the claim of correlation of Maintainability with Changeability and Stability, statistical analysis was performed. Being highly correlated, Changeability and Stability measures are used to develop Maintainability Measurement Model (MM^{OOD}) as a third contribution of the thesis. Subsequently, an empirical validation of the maintainability measurement model was carried out using commercial software applications. The experimental result shows, the developed model (MM^{OOD}) is highly significant and better than the existing models (MM^{OOD}) and Rajendra model.

6.2 FUTURE WORK

The model developed to measure software maintainability of object oriented systems is extremely significant and correlated with object oriented design properties.

Subsequently, we have validated the model using commercial software applications.

However, there is still some scope for future work that is listed below.

1. The models though have been analyzed and validated for separate data sets in the thesis, an analysis on larger data set may further help fine tune the value of coefficients.

2. The scope of this thesis is limited to establish the effect of stability and changeability on maintainability. The relationship between maintainability and other quality sub-factors or parameters may be analyzed as a direction for future work.

3. A generic guideline may be produced in the form of developer's manual for designing class hierarchy based on the results of the model.