CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

In today's world software is playing a highly significant role in our life, and hence, it is being used in different ways than ever before. Software systems are pervasive in all aspects of society, from electronic voting machine to online shopping. An important part of our daily life is mediated by software. Software has a major impact over the telecommunication, transportation, industrial process, military, entertainment, offices, aircrafts or even wrist watches and home appliances. The nature and complexity of software have changed significantly in the last two decades. The software has seen many changes since its initiation. Software industry has also progressed at a rapid speed through the computer revolution and recently the network revolution has been triggered and accelerated by the explosive spread of the internet and most recently the World Wide Web. A major problem of software industry is its inability to develop bug free software. Software industry has been delivering exponential increase in price and performance but still the problems with software are not decreasing. Software still come late; exceed budget and is full off residual faults/errors.

Software development process normally focus on avoiding errors, identifying and correcting software faults that do occur, and predicting software quality and reliability after development. It is well understood fact that producing high quality software is not an advantage but is an essential requirement. It was discussed by

Arora et al. (2011), Jalote (2012), Pizzi et al. (2013) & Amid et al. (2013) that majority of the industries not only fails to produce high quality software for their customers but also do not recognize the appropriate quality attributes. The potential impact of software errors on business, human life and environment have grown. It increasingly breaks more and more critical functionalities within software products and business processes. Our society is highly dependent on software. Failures of software can contribute or cause to serious accidents that result in injury, death, major financial loss or significant environment damage. Such software accidents have already happened and without intervention. The increasingly pervasive use of software especially in areas such as transportation, health care, and broader infrastructure may possibly make them more frequent and more serious.

Software testing is one of the most famous ways of promising to produce error/bug free quality software; the helpfulness of testing decides the quality of developed software. On the other hand, testing has now become a tedious task and a costly activity because of the rapidly increasing size and complexity of software. A latest survey done by Athanasiou et al. (2014), Felderer et al. (2014), Zheng and Bundell (2008) & Mao et al. (2007) reveals that the cost incurred in testing often range from 40% to 50% of the entire cost involved in software development. Software testing is a financial problem strongly intertwined with nearly all major technical issues in development life cycle.

Objective of software engineering is to create high quality software in time and within budget. If a product is meeting its requirements, we may say it is a superior quality product. The whole thing is measured with respect to requirements and if it matches, product is a quality product. Quality has become more important with our

increasing dependence on software. In the past decades, the demand for quality in software products has been increasingly emphasized.

The next section describes software quality.

1.2 SOFTWARE QUALITY

The 1990s and beyond is regarded as the quality era. In this era of information age, quality has been brought to the center of software development life cycle, as exemplified in capability maturity model. Different people understand different meanings of quality like: ‘Conformance to requirements’, ‘Fitness for the purpose’, ‘Level of satisfaction’ etc. In the environment of software engineering, software quality measures how well software is designed (quality of design) and how well the software conforms to that design (quality of conformance).Even though there are several definitions, it is frequently described as the 'fitness for purpose' of a part of software. One of the challenges of software quality is that everybody feels that they understand it but may not be able to clearly express the same. Software quality may be defined as conformance to openly stated functional and performance requirements.

In modern software development industry, despite knowing the benefit as well necessity of delivering quality product in market, expected level of quality are becoming more challenging and crucial. It was discussed by Xie et al. (2014), Huang et al. (2013), Elish et al. (2011), Gupta et al. (2005) & Dromey (1995) that with the ever increasing size and complexity of software applications, software industries lack to deliver the quality product and even some times some quality attributes are neglected. In this highly competitive software industry, companies are habitually trying to meet the release dead line that usually reduces the testing time. As a result,

the software product may not be correctly checked for the possible defects. Therefore, we cannot take the quality assurance part of each software product carelessly and the fault prevention and fault detection have to be considered at every possible step of development life cycle.

Measuring software quality is not a new theme, but it has been investigated for years in software engineering discipline. Since there is no clear understanding of 'what aspects of software quality should be considered qualitative', it is not easy to find suitable ways to measure it and other related aspects. While there is a uniform agreement that we need quality software but the question of 'how, when, and where you measure and assure quality' are far from the settled issues. It was highlighted by Birolini et al. (2014), Singh et al. (2014) & Cinneide et al. (2011) that testing is the course of action of scrutiny of any software to make sure that it performs as per the specified requirements. Main intention will be making the testing process easy and detecting the defects in effective and confident way. Ease of testing is measured in terms on testability. Testability is a quality that refers to the capacity of a system to be tested; it measures how simple it is to test in order to troubleshoot a given piece of software. High testability increases the probability of revealing the faults and its early measurement leads to the prospect of controlling fault, to facilitate and improve test process.

The next section talks about software testability.

1.3 SOFTWARE TESTABILITY

Testability has always been an elusive concept and its correct measurement or evaluation is a difficult exercise because various potential factors have effect on software testability. Testability is one of the most important quality indicators. Its

measurement leads to the prospects of facilitating and improving a test process. The notion of 'software testability' discussed by the experts Zuhoor & Martin (2003), Ghosh (2002), Gross et al. (2001), Martin & Zuhoor (2000) has been a subject of different interpretations. Consequently, several definitions of testability have been published in the literature; the most general definition of software testability is, ease of performing testing. Software testability is an external software quality attribute that computes the software complexity and the effort required for testing. It facilitates the testing process and makes the creation of better quality software possible. Testability is also defined by Kansomkeat et al. (2008), Bach (1999) & Gelperin (1999) as anything that makes software easier to test by making it easier to design and execute tests.

The next Section shows the relationship between Testability and Quality.

1.4 TESTABILITY-A KEY FACTOR TO QUALITY

An accurate measure of software quality fully depends on testability measurement. Software testability is a non functional requirement significant to the testing team members and the end users, who are involved in user acceptance testing. However, non functional requirements are quality requirements and make the customer happy and satisfied. Software testability is one of the important concepts in system design, and testing of software components and program. Developing programs with high level testability constantly simplifies test process and reduces test cost, as well as increases software quality. Software testability analysis may be useful to produce the quality software. Testability is important as highlighted by Wang et al. (2009), Ma et al. (2007) & Kolb et.al (2006) for both organizations and adhoc developers with a high level of development process maturity. It reduces development cost in a

reliability driven process, and increases system reliability in resource limited processes. Testability refers to the inherent ability or extent of ease with which software undergoes through final testing.

As discussed by Khalid et al. (2010), Baudry & Traon (2005) how easily the faults will be removed, depends upon the testability of the system. Most of the studies calculate testability or more specifically the attributes that have impact on software testability but at the code level of the development process. On the other hand, testability measurement at the code level is a good indicator of effort estimation process. Code level estimation leads to the late arrival of information in the software development process. Measuring testability at later phase of development life cycle after coding has been started may be very expensive and error prone. But if testability is measured earlier in the development life cycle, before coding starts, exclusively at design phase it may greatly reduce the development cost and rework. As a result it can accelerate the development process and improve the software quality.

Since we are working to measure the testability of object oriented software, in next Section we will discuss about object oriented design and how testability can be measured for such software.

1.5 OBJECT ORIENTED DESIGN

Object oriented technology have become the most popular, familiar and most widely used concept in software industry. Most of the focus of the object oriented approach to software development has been on analysis and design phase. Object oriented technology focuses on objects as the primary agents involved in a computation. Each class of data and associated operations are collected into a single system entity. It requires much significant effort at the initial stage in the development life cycle to

recognize objects and classes, attributes and operations and the relationships between them. Object oriented programming is a basic technology as stated by Lee et.al (2014), Azam et.al (2014) & Chidamber et.al (1994) that supports quality goals. Only by knowing the syntax elements of language and/or the concepts involved in the object oriented technology is far from being sufficient to produce quality software.

1.5.1 Design Properties

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) & Venkatesan et al. (2013). There are several important themes of object orientation that are known to be the basis of internal quality of object oriented design and support in the context of testability measurement. These themes significantly include cohesion, coupling, encapsulation and inheritance. Cohesion refers to the internal consistency within the parts of the design. A class is cohesive when its parts are very much correlated. It should be difficult to separate a cohesive class. Coupling designates the relationship or interdependency among modules. Inheritance is the sharing of attributes and operations among classes based on a hierarchical relationship. It is a mechanism whereby one object acquires characteristics from one, or more other objects. Encapsulation is a method to support information hiding and data abstraction. It hides internal explanation of an object and show only external interface.

Practitioners and researchers frequently advocate that software testability should be planned at the design phase of development process. Therefore it is necessary to recognize object oriented design properties to quantify testability measures at design phase of software development process. During identification of design artifacts

which have direct impact on testability measurement, a realistic view should be considered. If we consider all artifacts and measures then they become high complicated, ineffective or time consuming. Therefore, there is a need to identify design artifacts and measures which affect the testability measurement process directly. In order to estimate testability, 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 testability.

The next section talks about testability factors.

1.6 TESTABILITY FACTORS

Researchers and practitioners have made significant amount of effort and contribution in the way of investigating testability factors in common and object oriented software in particular. It was discussed by the experts R.V. Binder (1994), W. N. Lo & Haifeng (1998), S. Jungmayr (2002), Mouchawrab et al. (2005), Zhao et al. (2006), Mulo (2007) & Bashir et al. (2012) that it is hard to get an understandable view on all the prospective factors that have an effect on testability and the dominant degree of these factors under different testing perspectives. It is conclusive from the existing literature review that there is a difference among researchers, quality controllers and practitioners in considering the factors while measuring testability in general and absolutely at design phase.

Despite the fact that, getting a universally accepted set of testability factors is only probable. Testability quality criteria are the characteristics which help to identify the testability factors. Criteria present a more complete, actual definition of factors as well as criteria common among factors assist to show the interrelationship between

factors. The criteria of the testability factor are the characteristics of the software product or development cycle by which the factor can be judged or recognized. An endeavor has been made to collect a set of testability factors that can affect software testability. However, without any loss of generality, it comes into view to include the factors namely, modifiability, simplicity, understandability, flexibility, traceability, complexity, self descriptiveness and modularity. Out of these factors, some of them have their direct impact in measuring testability of object oriented software design, while others have less or negligible impact. An effort has been made to recognize the testability factors that truly affect testability measurement at design phase. It was evident from literature survey that Modifiability and Flexibility are the key testability factors that truly affect software testability measurement and fulfill the quality criteria.

1.6.1 Testability Measurement of Object Oriented Software

Measuring testability of Object Oriented Software is a criterion of key significance to software designers, developers and the quality controllers. It is clear from existing literature that researchers considered different phases of development life cycle for testability measurement. Only a few research studies have been devoted to explore the concept of testability measurement at design phase. It is a well understood truth that a decision to modify the software design in order to improve testability index after coding process has been started may be very costly and error prone. At the same time measuring testability at design phase in the development life cycle may significantly reduce the overall development cost. Badri et al. (2010), Kumar et al. (2010), Dino Esposito (2008), John Hunt (2007) & Gao et al. (2005) argued that testability should be measured as a key attribute in order to promise the quality

software. Practitioners repeatedly advocate that testability must be measured at the design phase of development cycle. After the above discussion our conclusion is that testability is a quality factor and its measurement always support for delivering quality software.

1.6.2 Testability Measurement at Design Phase

Software design is the most creative and extremely important phase in software development life cycle. Software design can play the major role to control and improve the software quality. The quality of software design affects the overall quality of final product. Software testability is a design issue and needs to be addressed at the design phase. Measuring testability at a later phase in the development life cycle leads to the late arrival of desired information, leading to late decisions about changes in design and simply increases cost and rework. Therefore, early estimation of testability initially at design phase in the development process may improve quality and reduce testing efforts and rework. Our ultimate objective is that it is during the design phase that testability estimation can yield the highest payoff: design decisions can be made to increase testability before coding starts. When the design meets the testability requirements, it can be implemented. Paying attention to testability at design phase in the development process can potentially enhance testing and significantly improves testing phase effectiveness. Our main goal is to provide a comprehensive and complete framework and model to help in measuring software testability in a practical approach, with a focus on the design phase of object oriented development.

1.7 PROBLEM STATEMENT, ITS SOLUTION AND IMPACT OF PROPOSED RESEARCH

It is evident from the above discussion that software testability should be incorporated at design phase of development life cycle. Practitioners emphasized on the need of having an organized and efficient approach for testability measurement. Based on the explanation and conversation, there may be a vast set of research question that need to be addressed. Some of the relevant ones are recognized and stated as follows:

- What are the factors that straightforwardly affect software testability at design phase?
- What is the impact of each factor on testability measurement?
- Can we develop a testability measurement model to measure software testability at design phase of development life cycle?

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 'Measuring Testability of Object Oriented Design at Design Phase'. The problem is further subdivided into four sub problems enumerated as follows:

1) Development of the Testability Measurement Framework (TMF^{OOD}) for design phase. This framework gives a systematic way to develop testability measurement model. The framework comprises of seven steps namely Testability Factorization, Object Oriented Software Characterization,

Recognition of Metric, Correlation Establishment, Testability measurement and Finalization, along with an added common step of Design Review.

2) Modifiability Measurement Model (MMM^{OOD}) development: During literature survey it was identified that modifiability is a key factor to testability, and therefore this sub problem deals with developing a model to measure modifiability. For this sub problem we develop the modifiability measurement model with the help of object oriented design properties. This model shows a high correlation among Modifiability, design properties namely Encapsulation, Inheritance, Coupling and related metrics namely Number of Methods (NM), Maximum Depth of Inheritance Tree (MaxDIT) and Number of Association (NAssoc) respectively. Empirical validation validates the proposed model for better level of acceptability.
3) Flexibility Measurement Model (FMM^{OOD}) development: During literature survey it was also identified that flexibility is a key factor to testability, and therefore this sub problem deals with developing a model to estimate flexibility. For this sub problem we develop the flexibility measurement model with the help of object oriented design properties. This model shows a high correlation among flexibility, design properties namely Encapsulation, Coupling, Cohesion, Inheritance and related metrics Number of Methods (NM), Number of Association (NAssoc), Number of Attributes (NA) and Maximum Depth of Inheritance Tree (MaxDIT) respectively. Empirical validation validates the proposed model for better level of acceptability.
4) Testability Measurement Model (TMM^{OOD}) development: Modifiability and Flexibility measures are used to develop Testability Measurement Model that

works at design phase. In order to reinforce the claim of correlation between Testability with Modifiability and Flexibility, the proposed model has been tested and justified with the help of statistical measures. Finally, it incorporates the empirical validation of the testability measurement model.

1.8 IMPACT / SIGNIFICANCE OF PROPOSED RESEARCH WORK

The contributions made in the thesis bridges the gap between software industries personal understanding of testability and researches related to the topic. All of the contributions made are novel and are significant in the following manner:

- After applying developed Testability measurement framework and model, any external mechanism is not required to control and improve software testability rather it can be managed by the design constructs itself.
- Modifiability measurement model (MMM^{OOD}) provides a Modifiability Indexing (MI) benchmark for other researchers and designers.
- Flexibility measurement model (FMM^{OOD}) provides a Flexibility Indexing (FI) benchmark for other researchers and designers.
- For project ranking, Testability Indexing (TI) is possible using the Testability Measurement model (TMM^{OOD}). The developed model may be generalized and used by others researchers.
- Designers are facing difficulties to appraise the less testable parts of their design at an early stage. Early testability estimation may help to better understand both the design and architecture information of the system.
- It may help to discover the underlying errors in the software design at the early stage of software development life cycle, leading to avoidance of unnecessary overheads.

- It may help to evaluate the quality of software and facilitate the estimation, and planning of new activities, testing activities in particular.
- The chances of achieving customer satisfaction with testable products are much higher.

1.9 THESIS OUTLINE

The Thesis is organized into the following 7 Chapters.

CHAPTER 1: INTRODUCTION

This chapter provides introduction to the area, software quality, software testability measurement, testability related issues and its measurement at design phase, object oriented design, problem statement, its solution and impact of proposed research and thesis outlines.

CHAPTER 2: LITERATURE SURVEY

This chapter consists of a literature survey on relevant topics, prominently including testability models. It includes comprehensive report on software testability models and related issues, comparison of testability measurement models along with a critical examination of the same and contextual inferences and conclusions.

CHAPTER 3: TESTABILITY MEASUREMENT FRAMEWORK.

This chapter presents a Testability Measurement Framework for design phase of development life cycle. This framework provides a systematic way to develop testability measurement model.

CHAPTER 4: MODIFIABILITY A KEY FACTOR TO TESTABILITY

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

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

CHAPTER 5: FLEXIBILITY A KEY FACTOR TO TESTABILITY

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

CHAPTER 6: TESTABILITY MEASUREMENT MODEL

This chapter presents the Testability Measurement Model (TMM^{OOD}) in terms of Modifiability and Flexibility. Furthermore, the relationship of Testability 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 testability measurement model.

CHAPTER 7: SUMMARY AND CONCLUSION

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

1.10 SUMMARY

In this chapter we have introduced the area with the help of concepts like software quality, testability, testability of object oriented software etc. We illustrated testability factors and testability measurement in general and exclusively at design phase of development life cycle. Impact of testability measurement and its importance at design phase has been analyzed for producing high quality software. Subsequently problem statement, its solution and impact of proposed research is listed and finally the chapter describes outline of the thesis.

In the next chapter, we will talk about literature survey.