CHAPTER 1

“INTRODUCTION”
1.1 Introduction

In the current era, software plays a significant role in our day to day life. Software systems are inescapable in all sects of our society, from electronic voting machine to web based shopping. An imperative portion of our everyday life is intervened by software. Software has a noteworthy effect over the media transmission, transportation, mechanical process, military, and workplaces, air ships or even wrist watches and home appliances. The nature and complexity of software have changed altogether over the last few decades, and software has seen many changes since its evolution. Software industry is developing at a quick speed through the effect of Personal Computer and by the widely spread of the World Wide Web. A noteworthy issue of software industry is its failure to produce bug free software. Software industry has been conveying exponential increment in cost and execution yet at the same time the issues with software are not resolved. Software still come late; surpass spending plan and is full off lingering bugs and errors.

Software development process typically concentrate on dodging errors, distinguishing and revising software fault that do happen, and foreseeing software quality and unwavering quality after development. It is surely a known certainty that delivering high calibre software is not quality issue yet is a basic requirement. It was discussed by Arora et al. (2011), Jalote (2012), Pizzi et al. (2013) and Amid et al. (2013) that the greater part of the software industry not just neglects to deliver top notch software for their clients, and additionally do not recognize the proper quality
properties. There are potential effects of software errors on industry and human life. It progressively breaks an ever increasing number of basic functionalities within software and business structure. Decreasing software quality can contribute or cause to saviour outcome, which affects major budgetary failure or critical condition to society.

Software testing is the most popular and highly reliable methods that promise to deliver bug free quality software. The testing is important phase of development that assures the quality of created software. The problem of testing is a repetitive and an expensive process on account of the quickly expanding size and unpredictability of software. An overview done by Felderer et al. (2014), Zheng and Bundell (2008) and Mao et al. (2007) uncovers that the cost acquired in testing frequently run around 45% to 55% of the total cost engaged with software development. Software testing is a money related issue explicitly relates with almost all significant issues in development life cycle. Target of software designing is to make reliable software in time and inside spending plan. if a product is meeting its requirements, we may state it is an unrivalled quality product. Quality has turned out to be more critical with our increasing reliance on software.

1.2 Software Quality

The era of 90’s can be marked as the quality era. In this period, data quality has been conveyed to the main theme of software development life cycle, as illustrated in development process. Every person has its own definition of quality like:
'Conformance to requirements', 'Wellness for the purpose, 'Level of fulfilment' and much more definitions can be quoted. In the initial stage of software development, software quality measures how well software outlined (quality of plan) and how well the software fits within that plan (quality of conformance). Even there are a few definitions; it is often depicted as the fit for the purpose' of a piece of software. One of the difficulties of software quality is that everyone feels that they understand it however they will be unable to obviously express the same. Software quality may be characterized as conformance to transparently express effectiveness.

In present scenario, software industry knows the advantage and need of conveying quality product. It was discussed by Xie et al. (2014), Huang et al. (2013), Elish et al. (2011), Gupta et al. (2005) and Dromey (1995) that with the consistently expanding size and complex nature of software applications, software enterprises need to convey the quality products and even a sometime quality attributes are ignored. In this scenario software industries are continuously trying to meet the discharge dead line that minimizes the testing time. As a result, the software product may not be effectively checked for the conceivable imperfections. Along these lines, the quality confirmation part of every software product can't take heedlessly and the fault counteractive action and fault location must be considered at each conceivable stage of development life cycle.

Measuring software quality is not a new topic; however it has been researched for a considerable length of time in software development life cycle.
Since there is no certain comprehension of 'what parts of software quality should be viewed as subjective', it is difficult to discover appropriate approaches to quantify it and other related issues. While there is a uniform understanding that we require quality software but it is difficult to choose how, when, and where to measure the quality. It was mentioned by Birolini et al. (2014), Singh et al. (2014) and Cinneide et al. (2011) that testing is the plan of analysis of any software to ensure that it executes according to the predetermined requirements. Fundamental goal will influence the testing to process easy and distinguishing the defect in viable and certain way. Simplicity of testing is measured in terms of testability, which is a quality that indicates to the competence of a system to be tested; it measures that it is so easy to test keeping in mind to troubleshoot a given piece of Software. High testability expands the probability of revealing the defects and its initial estimation prompts the possibility of controlling issue, to promote and enhance test process.

1.3 Software Testability

Testability has dependably been a delicate idea and its accurate estimation or assessment is a tedious exercise on the grounds that different potential factors have impact on software testability. Testability is a standout among the most essential quality indicators. Its estimation prompts the possibilities of encouraging and enhancing a test procedure. The idea of 'software testability' examined by the researchers Ghosh (2002), Gross et al. (2001) has been a subject of various understandings. Therefore, a few meanings of testability have been mentioned in the
literature; the most accepted meaning of software testability is the simplicity of performing testing. Software testability is an external quality property that processes the software complexity and the efforts required for testing. It encourages the testing procedure and improves the formation of quality software. Testability is additionally characterized by Kansomkeat et al. (2008), Bach (1999) as something that decreases the demand of testing in software development by providing simpler design to execute tests.

### 1.4 Requirement Testability of Object Oriented Software

A precise measure of software quality completely relies upon testability estimation. Software testability is treated as non-functional requirement noteworthy to the testing team members and end users, who are engaged with the client acceptance testing. Non-functional requirements are quality requirements that satisfy the client. Software testability is one of the critical ideas during software development, and testing of system module and program. Generating software with high testability continually streamlines test process and decreases test cost, and in addition expands software quality. Testability analysis of any software might be helpful to deliver the overall quality. Testability is critical as mentioned by Wang et al. (2009), Mao et al. (2007) and Kolb et.al (2006) for the both organization and informal developers with high level of development process mellowness. It decreases development cost in an unwavering quality driven process. Testability alludes to the natural capacity or degree of simplicity with which software experiences through specific testing.
As discussed about by Khalid et al. (2010), Baudry and Traon (2005) how effortlessly the defects will be expelled, relies on the testability of the software. The vast majority of the research figure testability and other factors of the quality that have positive effect on software development yet at design or code level of the development procedure. So we need an effective framework or model to assess testability and its factors at requirement phase of development, measuring testability at later stages of development i.e. after designing or coding is a decent way of estimation process, but to enhance the quality of requirement is more effective. Designing or coding time estimation prompts the late entry of data in the development. Testability measurement at later period of software life cycle in the wake of designing coding has been begun might be exceptionally costly and error prone. In any case, if testability is measured in the development life cycle, before designing begins, solely at requirement stage it might significantly lessen the development and maintenance cost. Thus it can quicken the development procedure and enhance the software quality. Since we are attempting to quantify the testability of object-oriented software, in next section we will discuss about object oriented software and how testability can be measured for such software.

1.5 Object Oriented (OO) Software

Object oriented (OO) development have turned into the most prominent, well-known and most broadly utilized idea in the software industry. A large portion of the concentration of the object oriented way to deal with software development has been
on requirement analysis. Object oriented approach concentrates on objects that are essentially associated with computation. Each class of information and related operations are gathered into an individual attribute. It requires a lot of efforts at the initial stage of the development life cycle to perceive objects and classes, behaviour and operations and the relations between them. Object oriented characteristics of software programs is a fundamental approach as expressed by Lee et.al (2014), Azam et.al (2014) and Chidamber et.al (1994) that supports quality objectives.

1.5.1 Low Level Object Oriented Properties

Object oriented properties guide the developers what to add and what to avoid. Various methods have been characterized so far to evaluate object oriented (OO) software as discussed by Gupta et al. (2015), Chauhan et al. (2014) and Venkatesan et al. (2013). There are a few critical subjects of object oriented property that are known to be the premise of inward quality of object oriented software and support with regards to testability estimation. These subjects fundamentally incorporate cohesion, coupling, encapsulation and inheritance. Cohesion alludes to the interior consistency inside the modules of the requirement. A class is durable when its modules are especially associated. It seems to be hard to isolate a strong class. Coupling designates the relationship or interdependency among modules. Inheritance is the sharing of properties and operations among modules in light of a various levelled relationship. It is a component whereby one object acquires attributes from, at least one different object. Encapsulation is a strategy to help data covering up and
information expression. It covers up internal details of an object and show just external layout.

Experts and researchers strongly advocate that software testability should be efficient at the early stage of development process. Along these lines it is important to recognize object oriented properties to evaluate testability measures at requirement stage of object oriented development. During identification of requirement artefacts and low level of object oriented properties which have direct effect on testability estimation; a logical view should be considered. If we consider all factors and measure them they turn out to be high complicated, inadequate or tedious. Along these lines, there is a need to recognize requirement factors and measures which influence the testability estimation process specifically. With the goal to assess testability, its direct measures are to be perceived. Requirement level properties like correctness, completeness, modifiability, understandability, flexibility that are indirectly supported by low level properties such as inheritance, cohesion, coupling encapsulation, and others will be analyzed keeping in view their general effect on software testability.

1.6 Testability Factors

Researchers and professionals have attempted and suggested the method for examining testability factors like understandability, flexibility, modifiability. It was discussed by the experts R.V. Binder (1994), W. N. Lo and Haifeng (1998), S. Jungmayr (2002), Mouchawrab et al. (2005), Zhao et al. (2006) and Bashir et al.
(2012) that it is difficult to get an understandable view on all the considered factors that affect testability, and the major level of these factors under various testing points of view. It is convincing from the current writing survey that there is a distinction among analysts, quality controllers and professionals in considering the factors while measuring testability that is should be done at requirement stage. In spite of the way that, getting a commonly established quality criteria of testability factors is just plausible.

Quality criteria of requirement testability are the attributes which help to decide the factors to assess software testability. Quality Criteria introduce a more complete, genuine meaning of factors and additionally accepted criteria among factors helps to show the strong factors interrelationship. Criteria are the attributes of software product or development process by which commonly accepted factors can be judged. A model has been made to gather a set of requirement testability factors that can influence quality. Although, without loss of simplification, it comes into view to incorporate the factors to be specific, modifiability, backward or forward traceability, understandability, flexibility, electronically-stored, self-descriptiveness and modularity. Out of these variables, some of them have their direct impact in determining testability of OO software, while others have less or insignificant effect in some context. An effort has been made to perceive the testability factors that actually influence requirement testability estimation. It was obvious from script overview that to estimate testability of requirement is genuinely influenced by the
modifiability of requirement and understandability of requirement, and satisfies the quality principles.

1.6.1 Testability Measurement of Object Oriented Software

Measuring testability of Object Oriented Software is a criterion of key significance to software developers, engineers and the quality-controllers. It is clear from existing writing that researchers considered different phases of development life cycle for testability estimation. Just a couple of researches are committed to examine the idea of testability estimation at requirement stage. It is well known that a choice to adjust the software requirement to enhance testability index after designing process has begun, might be expensive and error inclined. In the meantime measuring requirement testability of the development process may fundamentally decrease the overall cost of development. Badri et al. (2010), Kumar et al. (2010), Dino Esposito (2008), John Hunt (2007) and Gao et al. (2005) contended that testability ought to be measured as a key recognition to guarantee the quality software. Professionals over and again advocate that testability must be measured at the requirement period of development cycle. After the above discussion our finding is that testability is a quality factor and its estimation dependably strengthen for conveying quality software.

1.6.2 Testability Measurement at Requirement Phase

Software Requirement is the most innovative and critical stage in software development life cycle. Software requirement can assume the key part to control and
enhance the software quality. The quality of software requirement influences the overall quality of end product. Software testability is an early stage issue namely requirement and should be addressed at the requirement stage. Measuring testability at later stage in the development life cycle prompts the late entry of required information, prompting late choices about changes in requirement is cost consuming and time taking process. In this way, early estimation of testability at first at requirement stage in the development procedure may enhance quality and lessen testing endeavours. Our ultimate objective is to estimate testability at requirement phase that can yield the most elevated result: requirement changes to increase testability before designing begins. At the point when the requirement meets the testability criteria, it can be executed. Focusing on testability at requirement stage in the development procedure can possibly upgrade testing and essentially enhances testing stage adequacy. Our principle aim was to give an extensive and complete framework and model to help in measuring software testability in a viable approach, with an attention on the requirement stage of object oriented development.

1.7 Problem Statement of Proposed Research

It is obvious from the above discussions that software testability ought to be consolidated at requirement stage of development life cycle. Experts stressed on the need of having an effective approach for testability estimation.
In view of the clarification and discussion, there might be a set of research question that should be considered. A portion of the important ones are perceived and expressed as acquire after:

- What are the factors that directly influence software testability at requirement stage?
- What is the effect of each factor on testability estimation?
- Can we build up a testability estimation model to measure software testability at requirement phase of software development?

In connection to the above query that are appropriate to the concerned theme of the research; the examination was designed to be a blend of subjective and quantitative in nature. With the goal to address the above research issues, the issue explanation that has been figured for the evaluation is distinguished as “A Perspective Framework for Industry Professional for Integrating Testability in Requirement Elicitation Process”. The issue is additionally subdivided into four sub issues listed as follows:

1) Development of the requirement based testability measurement framework (RTF\textsuperscript{OOS}). This framework gives an efficient approach to create testability estimation model. The system involves seven stages to be specific testability factorization, object oriented software characterization, recognition of metric, correlation establishment, testability estimation and finalization, alongside an additional basic advance of design review.
2) Requirement Modifiability Measurement Model (RMM\textsuperscript{OOS}) development: During literature study it was recognized that modifiability is a key factor to testability, and along these lines this sub issue develop a model to measure modifiability. For this sub issue we developed the modifiability estimation model with the help of object oriented properties. This model demonstrates a high relationship among modifiability and low level object oriented properties to be specific cohesion, inheritance, and coupling and related measurements in particular afferent coupling (CA), measure of functional abstraction (MFA), methods of class (CAM), separately. Exact approval approves the proposed model for better level of worthiness.

3) Requirement Understandability Measurement Model (RUM\textsuperscript{OOS}) development: During literature overview it was likewise recognized that understandability is a key factor to testability, and in this way this sub issue manages building up a model to appraise understandability. For this sub issue we develop the understandability estimation model with the assistance of object oriented properties. This model demonstrates a high relationship among understandability and low level object oriented properties specifically coupling, cohesion, inheritance and related measurements afferent coupling (CA), measure of functional abstraction (MFA), methods of class (CAM) separately. Observational approval approves the proposed model for better level of acceptance.
4) Requirement Testability Measurement Model (RTM<sup>OOS</sup>) development: Modifiability and understandability measures are utilized to create testability measurement model that works at requirement stage. Keeping in mind the end goal to fortify the claim of connection between's testability with modifiability and understandability, the proposed model has been tried and advocated with the assistance of factual measures. At last, it joins the exact approval of the testability estimation model.

1.8 Significance of Proposed Research Work

The commitment made in the proposal overcomes any issues between software businesses individual understanding of testability and studies related with the topic. All of the commitments made are novel and are huge in the following way:

- After applying created Testability estimation framework and model, any outer tool isn't required to control and enhance software testability rather it can be overseen by the requirement develops itself.
- Requirement Modifiability model gives a Requirement based Modifiability Indexing (MI) benchmark for different analysts.
- Requirement Understandability Model gives a Requirement based Understandability Indexing (UI) benchmark for different analysts.
- For venture positioning, Testability Indexing (TI) is conceivable utilizing the Requirement Testability Estimation model. The created model might be summed up and utilized by others analysts.
Designers are confronting challenges to evaluate the less testable parts of their requirement at a beginning time. Early testability estimation may better understand both the requirement and engineering information of the framework.

- It might find the fundamental blunders in the software requirement at the beginning time of software development life cycle, prompting evasion of superfluous overheads.

- It might assess the quality of software and encourage the estimation and setting up of new exercises, testing exercises specifically.

- The odds of accomplishing consumer loyalty with testable products are considerably higher.

1.9 Research Objectives

Keeping in view the open problem areas as security, we identified some of the software Quality factors (Testability, Understandability and Modifiability) where we can contribute to research and in turn to software industry. We identified the following objectives to be achieved as a part of the complete research work:

I  To establish a relation between object oriented approach (at an early stage) and identified testability factors that supports the quality criteria of the requirement.

II  To propose a methodology, to figure out method to make software requirement specification testable.
III To improve the software requirements at an early stage with the help of testability analysis.

IV The major objective of software testability measurement is to find out which requirements are poor in quality and stages where faults can hide from software testing.

v To produce guidelines for testable requirement specification

1.10 Thesis Outline

Rest of the Thesis is composed into the accompanying 5 Chapters.

Chapter 2: Literature Survey

This Chapter comprises of a writing overview on important subjects, conspicuously counting testability models. It incorporates thorough investigate software testability models and related issues, examination of testability estimation models alongside a basic examination of the same and relevant deductions and conclusions.

Chapter 3: Requirement Testability Measurement Framework

This part models a Testability Measurement Framework for requirement period of development life cycle. This structure gives an efficient approach to create testability estimation mode.
Chapter 4: Model Development of Proposed Framework

This Chapter talks about the proposed; Requirement Modifiability Model of Object Oriented Software (RMM\textsuperscript{OOS}), Requirement Understandability Model of Object Oriented Software (RUM\textsuperscript{OOS}), Requirement Testability Model of Object Oriented Software (RTM\textsuperscript{OOS}) and built up measurable relationship between's Modifiability and low level object oriented properties.

Chapter 5: Validation of the Framework

The chapter provides experiential validation of the Modifiability, Understandability and Testability measurement model of requirement on available data set with the help of statistical tools.

Chapter 6: Limitation and Future Work

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

1.11 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 requirement phase of development life cycle. Impact of testability assessment and its importance at requirement time 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.