Software Design Testability Factors: A New Perspective

Mohd Nazir
Jamia Millia Islamia (A Central University), New Delhi. E-Mail: nazir_alig@rediffmail.com
Dr. Raees A. Khan
Babasaheb Bhimrao Ambedkar University, Lucknow. E-Mail: khanraees@yahoo.com

ABSTRACT
Testability has always been an elusive concept. Its correct measurement or evaluation is a difficult exercise because of the potential factors affecting testability. It has been revealed from literature survey that researchers and practitioners have made some efforts to measure testability at source code level. Measuring testability at code level leads to late arrival of design flaws. Changes in design at a later stage costs more than the changes made at there itself. It is an inevitable fact that by estimating testability, a decision on such a change may be incorporated at design phase. Considerable efforts have been made to list out testability factors in order to find out measurable characteristics of software testability. It has been inferred from the literature on testability factors that there is a heavy need of proposing a commonly accepted set of the factors affecting software testability. This paper puts forth an effort in producing a set of testability factors in terms of direct measurable. Using the identified set of factors, software characteristics and metrics, testability may be quantified early in design phase.

KEYWORDS
Software Testing, Testability, Quality, Object Oriented Characteristics, Quality Factors, Testability Factors.

INTRODUCTION
As software grows more complex and begins increasingly to replace human decision-makers in every facet of our society, software quality and reliability requires careful attention. Moreover, when the life of human beings and huge fortunes are dependent on automated systems, software quality assurance can no longer be treated lightly. Many a time software can be as unreliable as human beings when it is employed for decision making because there is always a possibility of subtle errors or drifts that may culminate into a disaster [2]. It is well understood that delivering quality software is no longer an advantage but a necessary factor. Unfortunately, most of the industries not only fail to deliver a quality product to their customers, but also do not understand the relevant quality attributes [22]. Building reliable and quality end software is becoming more and more important because of the ceaselessly growing complexity, pervasiveness and criticality of software. The advancement in the software development process has been accelerated drastically in the last couple of decades. As a result, the complexity of applications and environments has been substantially increased and schedules have been pinched. Under these circumstances, software quality tends to suffer; ensuring the desired level of quality and dependability became more crucial, difficult and expensive [6]. Software testing is one of the most prominent ways of assuring quality of software system; the effectiveness of testing decides the quality of released software. However, testing has now become a tedious task and an expensive activity because the size and complexity of software is growing rapidly. A recent survey reveals that the cost incurred in testing often range from 40% to 80% of the entire cost involved in development [7]. Software testing is an economic problem closely intertwined with nearly all major technical issues in Software Engineering [4]. Testability is a quality factor; its measurement or evaluation can be used to predict the amount of effort required for testing and help allocating required resources. However, testability has always been an elusive concept and its correct measurement or evaluation a difficult exercise. Most of the studies measure testability or precisely the attributes that have impact on testability at the source code level [1]. Estimating testability at a later stage leads to the late arrival of desired information, leading to late decisions about changes in design. This simply increases cost and rework. Therefore, early evaluation of testability in the development process may enhance quality and reduce testing efforts and costs. Therefore, one pro-active strategy that organizations can adopt is to design their software products with testability as one of the key design criteria [11]. Object orientation is now in widespread use by the software industry, despite the fact that technology is not mature enough from the testing point of view [23]. While in an object-oriented context, new testing challenges arise due to complex interactions and couplings among classes, some classical testing problems, such as covering control flow graphs, may disappear. However, object orientation facilitates software design to be modeled at a higher level. As a result, any potential problem with the design can be fixed at this level at the right time. Rest of the paper is organized as follows: Section 2 briefly addresses software testability, software design and testability at design phase. Section 3 briefly summarizes the contributions in the area by various researchers. Section 4 presents motivational facts and the identified testability factors. Section 5 finally presents summary and conclusion. Section 6 points out future direction of research in the area.
SOFTWARE TESTABILITY
Software testability has always been a challenge before the researchers and practitioners working for quality improvement. It is evident from literature survey that testability is not a new term; rather it has been in discussion among the industry professionals at various forums. A critical look at the work reveals that practitioners are considering testability with different perspectives. It seems that, in reference to software testability, there are two major groups of experts advocating for testability estimation and improvement. From the decades back, these experts are strongly recommending to pin the testability bud at source code level. On the other hand, recent developments in software configuration management advocate integrating software testability in design phase to improve quality and security [14]. Both groups are strengthening the idea of estimating testability in order to enhance quality, but it has been observed that there is no common consensus among the researchers and practitioners about the aspects actually related to software testability [13].

Testability is one of the most important quality indicators since its measurement leads to the prospect of facilitating and improving a test process. The process of software engineering evolves with a unique issue of testability [17]. Software testability analysis has been an important research direction since 1990s and became more pervasive in 21st century [13]. Although, some achievements have been made on the measurement of testability, design for testability, testability to reliability estimation; these progresses have not been widely adopted in practice by industry. Designing, verifying and measuring highly testable software are becoming the important and challenging tasks for software developers. It has been recommended by the testability experts to integrate testability as early as possible in development life cycle. Measuring and evaluating testability at the early stage of software development process improves design well in advance before implementation and maintenance, where changes are highly expensive and error-prone [1]. Measures of testability can give an indicator as to the effectiveness and efficiency of testing. Thus, early evaluation of testability in the development process may greatly reduce testing efforts and costs.

According to IEEE standard the term of testability refers to the degree to which a system or component facilitates the establishment of test criteria. Further, it describes the performance of tests to determine whether those criteria have been met. It also defines the degree to which a requirement is stated in terms that permits the establishment of test criteria and performance of tests to determine whether those criteria have been met. Moreover, testability is a software quality characteristic that is of major relevance for testing cost and software dependability. Still, testability is not an explicit focus in today’s industrial software development projects [8]. An extensive survey of literature reveals that processes, guidelines, and tools related to software testability are missing [8], but advocated generally to be inevitable.

SOFTWARE DESIGN
Software design, in some ways, is a strange art [9]. At the first instance it may result in a model of the domain by formally capturing and representing the user’s requirements and hence paving the way for a conceptual relation. It may serve well as a communication medium between designer and the user on one end, and act as a basis for implementation on the other end. No doubt, it is the key to the successful development of quality software. It is also the step that will determine the overall structure, nature, and approach of the resulting software. This is evident that the software design is an important stage spanning the whole software lifecycle, not only for software development but also for re-developing legacy systems.

Quality of software design heavily affects the quality of the final products because design is the backbone or the skeleton of any software system. A defective skeleton may not allow harmonious growth and can not easily accommodate changes without cumbersome prosthesis with possibly introducing various kinds of side effects. Due to the fact that, most of the time requirement analysis is incomplete, software designs should be built in such a way so as to make them easily understandable, testable, alterable, and preferably stable. If software design fails to accurately cover requirement specification, it may result in a failure of the project. On the other hand if it is made too complicated, it may cause unnecessary problems during and after the implementation that may result in shooting up the cost of a project. Researchers and practitioners recommend that flaws are best found by pondering design issues that go far beyond code. Hence, controlling and improving software design quality has been one of the important issues in Software Engineering [2].

TESTABILITY AT DESIGN PHASE
Testable design is more specific than good design as it is explicitly intended to match a particular test context. Improving software testability is clearly an important objective in order to reduce the number of defects that result from poorly designed software [10]. Any technique that improves a software design at an early stage can have highly beneficial impact on the final testing cost and its efficiency. Although software testability is most obviously relevant during testing, but paying attention to testability early in the development process can potentially enhanced testing and significantly improves testing phase effectiveness. Indeed, what makes a design testable is the easiness to test the components in the design. Therefore, one pro-active strategy that organizations can adapt is to design their software products with testability as a key design criterion [11]. It is suggested by a renowned expert that testers must realize that testability requires them to understand the software design, which may provide the basis for suggestions and facilitates concrete discussions [20].

TESTABILITY FACTORS
Software testability is an external software attribute that evaluates the complexity and effort required for testing
Software Design Testability Factors: A New Perspective

Software. The insight provided by software testability is valuable during design, coding, testing, and quality assurance [12]. However, testability is an elusive concept, its correct measurement or evaluation a difficult exercise. Furthermore, it is difficult to get a clear view on all the potential factors that can affect testability and the dominant degree of these factors under different testing contexts [13]. Most of the mechanisms available for testability estimation of object-oriented software, may normally be used in later phases of system development life cycle and rely upon information extracted on the operationalization of software. Such methods provide an indication of testability and hence quality, but too late to improve the product, prior to its completion.

The characteristics of testable software like adequate complexity, low coupling and good separation of concerns make it easier for reviewers to understand the software artifacts under review [21]. Plenty of work has been carried out in describing the need and importance of incorporating software testability since early 90s. A number of methods of measuring testability have been proposed. Unfortunately, the significant achievements made by the researchers in the area have not been widely accepted and are not adopted in practice by industry [13]. It has been found that there is a conflict in considering the factors while estimating software testability in general and at design level exclusively. Following sections briefly presents some relevant efforts made by researchers in the direction of finding our testability factors:

Robert Binder, testability Guru, in [4] did a novel work in finding out the need and significance of considering testability while developing software system. He is the first to introduce controllability and observability as two main facets of testability. He considered six factors of software testability including representation characteristics, implementation characteristics, built-in-test capabilities, test suite, test support environment, development process. The proposed factors are only described at a high level of abstraction, which lead to no clear relationship with the metrics that are based on design artifacts and the implementation. It seems that there is no empirical evidence of correlation between the suggested metrics and testability. Binder proposed a testability fishbone model to represent all its facets.

In 1998, Bruce and Haifeng Shi [5] examined the factors that affect software testability in object-oriented software and proposed a preliminary framework for the evaluation of software testability metrics. They also formulated a set of guidelines in object-oriented design to improve software quality by increasing their testability. Characteristics of OO software systems contributing to software testability were categorize into three groups namely: structure factors, communication factors, and inheritance factors. James Bach in 1999 defined testability as anything that makes software easier to test by making it easier to design and executes tests. Bach presented heuristics of software testability. He described testability and stability. He strongly believed that these are the characteristics of program leading to testable software.

Jungmayer in [18] identified testability as a major factor determining time and effort needed to test a software system. He strongly recommended to deploy testability into the system continually right from the beginning, especially at design phase. Large number of software related factors that affect testability during design were identified. Jungmayer grouped them into nine main testability factors namely complexity, separation of concerns, coupling, fault locality, controllability, observability, automatability, built-in-test capability, and diagnostic capability.

Y. Wang in [11] demonstrated that software testability at class and system levels can be quantitatively modeled and analyzed. He defined testability as a product of test controllability and observability. A set of fundamental built-in-test capability mechanism oriented to the basic control structures of code was developed in order to improve the testability of software in terms of controllability, visibility, operability, simplicity, understandability, suitability.

In [3] Bruintink and van Deursen identified the factors, which are related to testability of object-oriented software and relate testability to the number of test cases needed to test a system and to the effort required to develop each test case. They define and evaluate a set of metrics for assessing the testability of classes of a Java system. The approach characterizes software testability for object-oriented systems using source code metrics. They categorize the source code factors as the factors that influence the number of test cases required to test the system, and factors that influence the effort required to develop each individual test case. They establish a correlation between class level metrics, test level metrics, and the number of lines of code per test class. But inheritance-related metrics and the proposed testability metrics are not correlated.

Samar Mouchawrab et al [1] advocated that testability analysis can yield the highest payoff if focused during analysis and design stages of object-oriented development. A decision to change the design in order to improve testability after coding may be very expensive and error-prone. They investigated on how to measure testability based on design artifacts and proposed a framework that may help to assess testability of design that is particularly modeled with the UML. A number of hypotheses are proposed for each attribute in order to explain its expected relationship with testability, but these hypotheses are not empirically validated. Furthermore, they concluded that existing work on the testability measurement either take a very specific viewpoint or remain at a very general level. They opined that the state of the art is scattered, practitioners evaluating and analyzing testability, their designs lack operational guidelines on how to proceed in a systematic and structured manner. It is very obvious that despite considerable efforts are made in the direction by the researchers; much work still remains to be done in the area of testability measurement and evaluation.

Jerry Gao, and Ming-Chih Shih [16] discussed the component testability in a quantifiable approach based on a component testability analysis model. Further, they proposed a pentagram
model for testability measurement based on the five factors discussed component testability model. In future direction of in one of their papers presented at 29th Annual International Computer Software and Applications Conference, in 2005, they proposed to carry out a detailed testability verification guideline for design.  

Ronny Kolb and Dirk Muthig, in [15], presented a multi-dimensional model that defines testability relative to the testing phase (i.e., component test, integration test, system test, and regression test) and an activity of the test process (test case creation, test case execution, test result analysis, and defect localization). Furthermore, testability is considered as one of the characteristics of the architecture, design, and implementation. Using this model of testability, it is possible to discuss in detail and analyze testability more explicitly. They advocate that testing of product lines can be made more efficient and effective by considering testability already during architectural design. Emmanuel Mulo presented a report strengthening the integration of testability throughout development process [6]. He considers testability from two main stances: as an inherent property in the systems under consideration or as a strategy in the development process to make testing easier. As an inherent property, testability is defined by the terms observability and controllability. He strongly recommended to apply certain tactics in practice that improve on a tester’s ability to better manipulate the software and to better observe and interpret the results from the execution of tests. Dino Esposito recently talked about testability and security as a key characteristic to determine the overall quality of any software system [14]. Being a key attribute to guarantee the quality of a software system, testability should be planned early in the design phase. Dino argued in his talk that software testability results from a number of characteristics of the code being tested that the development team should ideally guarantee viz visibility, control and simplicity. Through the foregoing description, there appears a trend and approach adopted by experts in a systematic manner. A close look at the same is concluded in the table 1.

<table>
<thead>
<tr>
<th>Testability Factor</th>
<th>Observability</th>
<th>Controllability</th>
<th>Built-in-test</th>
<th>Traceability</th>
<th>Understandability</th>
<th>Automatability</th>
<th>Diagnostic capability</th>
<th>Fault locality</th>
<th>Simplicity</th>
<th>Complexity</th>
<th>Test support environment</th>
<th>Test suite</th>
<th>Development process</th>
<th>Representation characteristics</th>
<th>Implementation characteristics</th>
<th>Separation of concerns</th>
<th>Coupling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jerry</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wang</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bach</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jungmayr</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binder</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mulo</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Testability Factors Consideration by Various Experts: A Critical Look

**SUMMARY AND CONCLUSION**

With growing complexity, pervasiveness and criticality of software, building reliable and quality end software is becoming more and more challenging. Moreover, the advancement in the software development process has been accelerated drastically in the last couple of decades. As a result, the complexity of applications and environments has been substantially increased and schedules have been pinched. Under these circumstances, software quality tends to suffer. In the face of intense competitive pressure, a comprehensive and rational strategy to achieve high testability will be a strategic advantage-not a bottleneck [21]. The foregoing analysis implies that testability results from good Software Engineering practice and an effective software process. Improving software testability has become an important objective in order to reduce the number of defects that result from poorly designed software. Undoubtedly, testability is a key factor to software quality and security, and plays an important role in delivering safe and quality software. It is an obvious fact that by estimating testability early, a decision
Software Design Testability Factors: A New Perspective

may be taken to incorporate changes at design phase. In order to fulfill the immense need of commonly accepted set of the factors affecting software testability, an effort has been made in this paper in the form of a set of testability factors early in design phase, depicted in figure 1.

**Figure 1: Software Testability Factors in Design Phase**

**FUTURE WORK**
An efficient and accurate measure of software testability early in design phase is highly recommended by various researchers and practitioners. There is a common agreement among industry professionals and academicians to integrate testability with the development life cycle in order to deliver secure, safe and reliable software within time and budget. Unfortunately, there is no standard methodology or guideline available to quantify software testability. In the absence of any framework or model to estimate software testability, it is worthwhile to make an effort to get a prescriptive framework to quantify software testability in early stage of development process. Therefore, such a roadmap, which can be followed by the industry personnel and researchers to quantify software testability in design phase, appears highly desirable.

**REFERENCE**

http://www.regdeveloper.co.uk/2007/10/29/design_for_testability/page2.html
http://www.ucalgary.ca/~ageras/wshop/abstracts/2003/design-for-testability.htm


