Abstract: Component-based software development has become one of the preferred streams for developing large and complex systems by integrating prefabricated software components that not only facilitates the process of software development but is also changing the ways for software professionals to develop software applications. Component based software development process is used for making a rapid software product by using fewer resources. Different components are collected and integrated together to form new product, so that the quality of new software product depends upon these components. Since components are the major building blocks for component-based systems, developing high quality components is becoming critical day by day. For generating high quality components, we must pay attention to component testability in order to ensure that the reusable components can not only be tested by component vendors, but also can be easily validated by component users. To ensure quality of overall product, testing of each component is very essential. Therefore, component testability becomes very important research topic in testing component-based systems. This paper discusses the component testability as a quantifiable approach based on a component testability point of view. Some important strategies of component testability are also discussed in this paper. This paper also discusses the different challenges during testing of software components and in the end of paper after considering different testing issues, suggestions to improve component testing strategy is provided.

Keywords: Software components, component testability, quality components, component testing strategy.

1. Important Strategies of Component Testability

Component-based software development (CBSD) has become one of the preferred streams for developing large and complex systems by integrating prefabricated software components that not only facilitates the process of software development but is also changing the ways for software professionals to develop software applications [1]. In present era of component paradigm, software testability has become an important concept in design and testing of software components because a program with good testability could not only simplify software test operations but also helps in reducing test cost. In this section, we will discuss the testability of software components from the component engineering point of view. Software testability can be viewed from two different aspects:

(i) Component Observability
(ii) Component Controllability
(iii) Component Presentation
(iv) Component Test Suite & Component Test Support
(v) Configuration Management
(vi) Component Understandability

1. Component Observability
Component-based software development (CBSD) has become one of the preferred streams for developing large and complex systems by integrating prefabricated software components that not only facilitates the process of software development but is also changing the ways for software professionals to develop software applications [2]. Software observability indicates how easy it is to observe a program in terms of its operational behaviors, input parameters, and outputs [3]. In context of component engineering, we found that the component traceability is an important factor that affects the component observability. Component traceability refers to the extent of behavior tracking capability of components. Component Traceability of a software component refers to the extent of its build-in capability of tracking the status of component attributes and component behavior. It has two folds:
A. Behavior Traceability: Degree to which a component facilitates the tracking of its internal and external behaviors [4].
B. Trace Controllability: Extent of the control capability in a component to facilitate the customization of its tracking functions.

2. Component Controllability
Controllability of a program (or component) indicates how easy it is to control a program (or component) on its inputs/outputs, operations, and behaviors [4]. For component developers, "controllability" of a software component includes three folds:
A. Controllability of its behaviors and output data responding to its operations and input data.
B. Built-in capability of customization and configuration of its internal functional features.
C. Installation capability.

3. Component Presentation
Component presentation consists of four factors:
A. Presentation Of Components for Users: As per these documents, users learn how to use a component. E.g.
component user manual, user reference manual etc.

**B. Presentation Of Design And Analysis of Components:** Engineers use component analysis and design documents to understand a component in terms of its interfaces and constraints.

**C. Component Program:** It includes component source code and its supporting elements. E.g. installation code, test driver etc.

**D. Component Testing and Quality Information:** It includes component test plan and/or test suites, test reports etc.

4. Component Test Suite and Support

Consistency is the key behind making of component test suites in developing component-based software. The basic idea is to build tests inside components. This simplifies component testing and reduces test cost. Testing software components needs several types of supporting facilities like test generation tools, component test drivers and stubs etc. In order to support component testing effectively, we need standard formats for test information in test suites and a consistent access interface between components and test suites.

A. **Constructing component test drivers and stubs systematically:** In the component-based software construction process, in order to cope with diverse software components and their customizable functions, we need new systematic ways to construct test drivers and stubs for software components.

B. **Building a generic component test-bed:** The challenge in component development and testing is how to construct a generic test bed, which is reusable to components based on different languages and technologies.

5. Configuration Management

Typically, the development of component-based systems starts with a collection of existing components [5]. During the life cycle of component-based software, evolution of software products depends on modifications and upgrades through involved components. To reduce the evolution cost on regression testing, configuration engineers must add component test drivers and stubs into their priority list. To support the evolution of components, they must keep tracking of the versions and releases of component test suites and problem databases for each component.

6. Component Understandability

A component should be able to be developed, acquired and incorporated into the system and composed with other components independently in time and space [6]. Component understandability depends on how much component information is provided and how well it is presented. It depends on factors like:

A. **Presentation of component documentation:** For a component, there are two sets of documents:

a. Written for users: e.g. component user manual, user reference manual, and component application interface specifications.

b. Written for component engineers: e.g. component analysis and design specifications, component testing and maintenance documents.

B. **Presentation of component program resources** with component source code, installation code, and test drivers.

C. **Presentation of component quality information**, including component acceptance test plan and test suites, component test metrics and quality report.

II. CHALLENGES TO COMPONENT TESTABILITY

Testing involves operation of a system or application under controlled conditions and evaluating the results [7]. In the component engineering paradigm, one of the primary goals is to generate reusable software components as software products. The third-party engineers use the components as parts to build specific software systems according to the requirements given by customers. Therefore, the testability of a program highly depends on the testability of involved components and their integration. Therefore, several concerns exist while building and testing of software components like:

A. How To Reuse Component Tests?

Over time, due to increased product functionalities, software projects have become more and more complex and along with increasing work completion pressures, the software projects are required to be accomplished in lesser amount of time but with fewer people. So, the increasing complexity included with time and budget constraints, has declined the product quality [8]. The primary key behind reuse of component tests is to develop some systematic methods and tools to provide reusable component test suites to manage and store various component test resources. In the component engineering practice, software development teams use an ad-hoc approach to create component test suites through a test management tool. Since existing tools usually depend on different test information interfaces, it becomes difficult for the engineers to deal with diverse software components on a consistent test suite technology. This problem affects the reuse of component tests while performing component integration and component acceptance testing. In order to solve the above said problem, there are two alternatives:

a. Create a new test suite technology for components with plug-in-and-test techniques, so that engineers are able to construct a test suite for any component, and are also able to perform component tests using a plug-in-and-test technique.

b. Create component tests inside components, known as build-in tests. Unlike the first approach, this approach creates component tests inside components: so, it simplifies the process of component testing and reduces the component test cost.
B. How To Construct Testable Components?
An ideal testable software component is not only deployable and executable, but also testable with the support of standardized testing strategies. Testing involves operation of a system or application under controlled conditions and evaluating the results [8]. Unlike normal components, testable components have the following features:

- Testable components must be traceable: Traceable components are ones constructed with a built-in tracking mechanism for monitoring various component behaviors in a systematic manner[9].
- Testable components must have a set of built-in interfaces to interact with a set of well-defined testing facilities.
- Although testable components have their distinct functional features/data/interfaces, they must have a well-defined test architecture model with built-in test interfaces.
- Testable components with built-in tests must use a standardized mechanism to enclose the built-in tests to be used in a consistent way.

C. How To Construct Component Test Drivers/Stubs?
In the component engineering paradigm, components might be customized according to the given requirements. So, we need new systematic methods to construct customizable test drivers and stubs for diverse components.

III. SUGGESTIONS TO IMPROVISE COMPONENT TESTING
An improved testing method would make a component/product more usable and there would be less chances of failure. These are some of the suggestions to improve the component testing.

A. Testing automation can help to evaluate large number of test cases that would be more effective to detect faults.

B. More information sharing between consumers and producers in case of third party component usage in order to have lesser problems while testing components.

C. More work should be done to create generic test suites as it would be more helpful for testing components with the similar behavior.

D. In case of component change, the information should be provided to the consumer so that they have complete information about changes made in the components.

E. Complete documentation including user manuals and interface specification should be provided by the component producers along with the components.

F. Testing should be made on the reusable components under different scenarios so that reusable components have vast scope of usability under different conditions.

G. Testing results should be stored for future enhancements.

IV. CONCLUSION
In the component based software development different and already built components are used. To ensure the quality of such product component testing is very essential. Testing should be performed on every component and whole software product before delivering it. But problems may arise for the tester in component testing phase due to the limited access of the component. In this paper component based testing strategies are discussed and using a good strategy according to situation quality of the product can be improved. If best suitable practices are applied on new component based software it become more reliable and chances of failure become very less. Still lot of work needs to be done in component testing area. Some suggestions are also given which can lead to better testing to deliver a quality component based software product.

V. REFERENCES