Journal Homepage: -www.journalijar.com
INTERNATIONAL JOURNAL OF ADVANCED RESEARCH (IJAR)
Article DOI:10.21474/IJAR01/7722
DOI URL: http://dx.doi.org/10.21474/IJAR01/7722
ISSN NO. 2320-5407

RESEARCH ARTICLE

STRATEGIES AND ROLE OF COMPONENTS TESTING IN COMPONENT BASED DEVELOPMENT ENVIRONMENT.

Ms. Jyoti sharma[1] and Dr. Arvind Kumar[2].

1. Assistant Professor, Department of Computer Science & Engineering,Maharaja Agrasen Institute of Technology, Rohini, Delhi, India.
2. Assistant Professor,Department of Computer Science & Engineering, SRM University, Delhi- NCR, Sonepat, Haryana, India.

Manuscript Info

Manuscript History
Received: 21 July 2018
Final Accepted: 27 August 2018
Published: September 2018

Keywords:-
Component,Testing,CBSD

Abstract

Testing is basically a way of finding out that error or bug is present or not in the given module or the component. There is various number of testing approaches or strategies which are present in various books but for the component based testing the various approaches must be applied in the different manner according to the type or behavior of the component or on the category of the component. In this paper we have discussed about the idea of applying the present schemes of testing on the component in component based development environment.

Introduction:-

The components used in the development of the software are basically from the different backgrounds and for the different purposes as well. After aggregation of components in the component repository the well suited components according to certain number of parameters are selected. A component is generally developed once and then it has to be reused by modifying it or adapting it in a new system with a new set of requirements. So due to this reason the testing of components before integrating in to system is very important and plays a significant role in the development process of software. Due to the evolving role of software the testing plays an important role in the overall process of software development. The testing of the software is very necessary because if a component is released for the development of software without testing then it may lead to less reliability, high complexity and bad performance of the system.

The rest of the paper is organized as follows. Section 2 describes the general testing strategy which we will follow for the component selection. Section 3 explains the applicability of the testing strategies on the various components. In the section 4,5 we have the experimentation results and also the comparison with the testing techniques and finally section 6 concludes and explains the future scope.

Role Of Component Testing In Software Reuse

The software reuse was introduced in 1968 but due to the managed complexity of software system it was considered as so much important at that time but when the complexity starts to cause a problem then it is considered as an important concept and various methods for managing the complexity of the software has been Introduced.

Corresponding Author:- Jyoti sharma.
Address:- Assistant Professor,Maharaja Agrasen Institute of Technology,Rohini,Delhi,India

If the components are well tested before integration in to the final system then that particular component will be more reusable as compared to other set of components. The good testing of the components will enhance the performance of the system and reduce the rest cost and helps to improve the software quality.

Parameters For The Reusability Of Component

The reusability of a component must be high enough in order to use the component in various development environments. The customization of component during the development of a CBS plays a significant role. The summarizations of a large number of factors are responsible for estimating the reusability of a component which can be stated as: -

1. Documentation of the component
2. Component Dependency
3. Previous runtime chart for component usage
4. Component Modifiability or component updation
5. Component Interface complexity

Documentation of component: -

The components which have to be reused again and again must be properly fit in to several architectural styles on the basis of various need of the user. The documentation of a specific component must include the detailed information about the structuring of a component and also about its various compatibility issues. The purpose of achieving the better reusability of the component must be enriched in its well defined documentation.

Component Dependency:-

The components are generally interacting through its interfaces. The good reusability of a component is also dependent on the factor of component dependency. The usage of component in various development situations must have a moderate component dependency on other components or in other words the component must be least dependent on other components for its proper functionality.

Previous runtime chart for component usage:-During the selection of components in CBSD, the previous runtime chart must be considered for the better development of application or software. It will help out in better selection of components as well as the recent or the previous history of the component will help out the developer in choosing the optimal set of components.

Component Modifiability or Component Updation:-

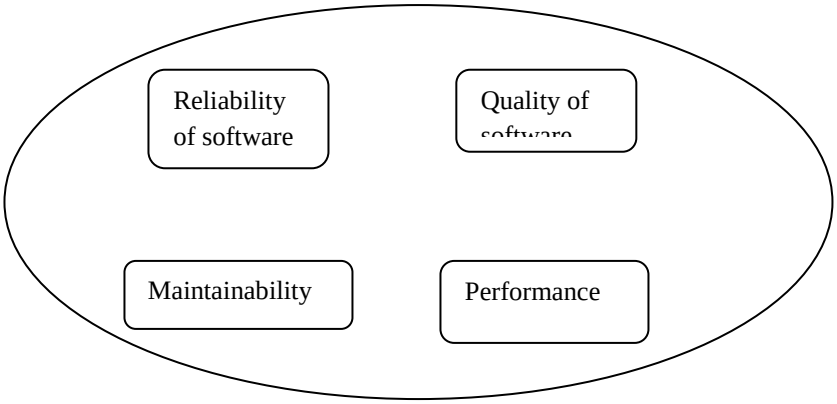
Modifiability of a component is defined as the ability of the component to go through the process of updation according to certain requirement specification. The better updation of a component will lead to a good software development. The more tendency of a component to go through updation or modifiable will lead good software development.

Component Interface Complexity:-

The COTS components are generally black box in nature but the in-house components are glass box in nature. The complexity of interface of component is least important in case of evaluation of COTS components but for the in-house components, it's better to use the less complex interface complexity component to reduce the complexity of whole application or software to be build.

Role Of Component Testing In Component Based Software Development

The overall development of software is heavily dependent on component testing. The components are generally of four types like in-house components, OSS components, outsource components, COTS components. The testing of the individual component is also dependent on the type of the component which is used. The factors which are affected on the basis of component testing are:-

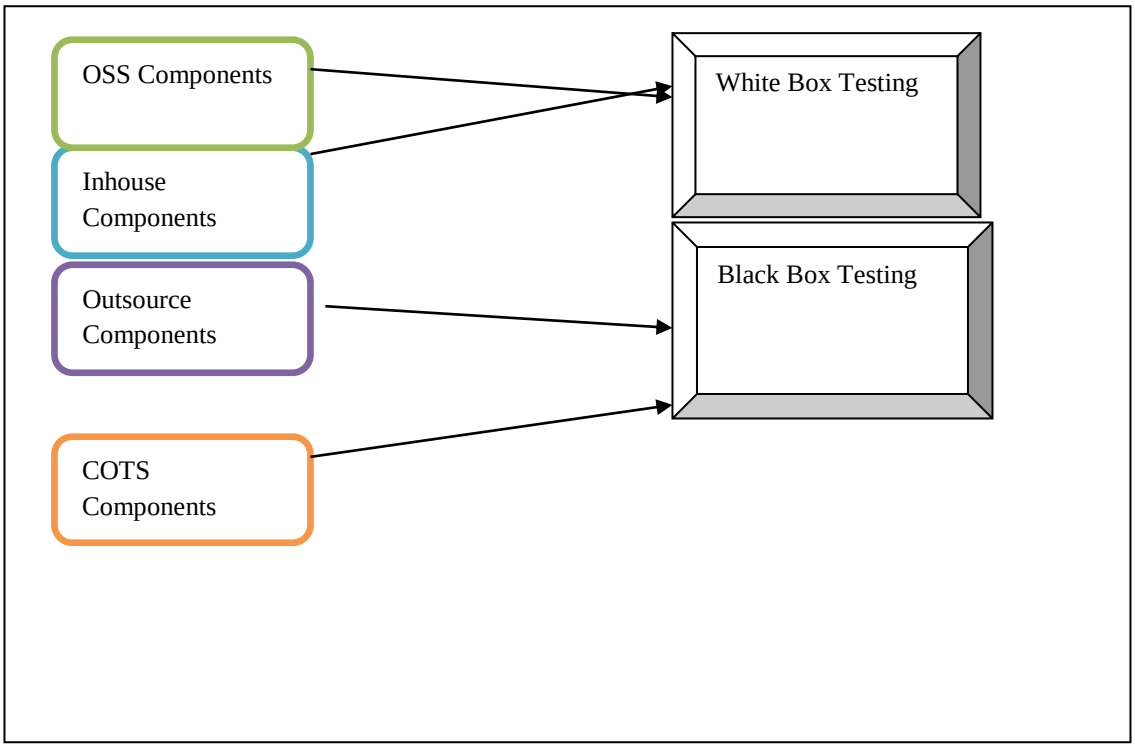


The testing of the components helps to improve the overall reliability of the software because the good testable components contribute to the reliability of the software. The reliability of the component is directly proportional to the quality of the software.

The good testable components help to reduce the errors in the software so it automatically improves the performance of overall system. Generally the products are divided in two categories like generic product and customised products. Component testing plays an important role in generic and customised software products.

Strategies Of Component And Software Testing

The software is generally built from a various number of operating procedures, documentation and a many number of programs. A various number of testing strategies are available for testing of software like black box testing and white box testing. Due to a variety of components available in the market these testing strategies can easily be applied for example the components where the source code is available, the white box testing plays an important role and the testing of components where the source code is not available, the black box testing helps out to tackle with the situation. The black box testing is generally applicable for COTS components because in this type of testing its not required to see the internal details of the component rather than to concentrate on the requirement specification for the components. Various types standard testing methods can be applied on to various types of components available.



In the OSS and In-house components, the white box testing can be easily applied because the source code is available. The source code is available with the fact that nobody is having permission to modify or to edit that source code. But due to availability of the source code the factors or the parameters related to the reliability and complexity of the software can be calculated so that they will provide more security and safety in selecting a component form the component repository. The advanced testing of components helps out in better development of the software.

There are various numbers of strategies which are basically employed in the testing of the software like black box testing, white box testing and grey box testing.

Consider the following code of a component in software for applying the standard strategies.

CheckCust.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;
import rail.ConnectionBean;
public class CheckCust extends GenericServlet
{
        public void service(ServletRequest request,ServletResponse response) throws ServletException, IOException
        {
 PrintWriter out = response.getWriter();
 //   RequestDispatcher rd=request.getRequestDispatcher("AdminHome.jsp");
                        try
                        {
          ConnectionBean CBean=new ConnectionBean();
     Connection con=CBean.getConnection();
          response.setContentType("text/html");
           String uid = request.getParameter("uid");
        String pwd = request.getParameter("pwd");
        System.out.println("user uid" +uid);
        System.out.println("pwd uid" +pwd);
        ResultSet custrs=CBean.executeQuery("select cid from customer_info where cid='" + uid + "' and pwd='" + pwd + "' ");
                        if(!custrs.next())
                        {
          out.println("Invalid Login Information");
                        }

            }
                catch(Exception e)
                {
                        out.println("Invalid Login Information"+e);
  }
```

The PC3M (Package cohesion component complexity metric) computed for the above code comes out to be 0.33[10].Similarly the values for the components whose source code is available can be computed and the appropriate approach of testing the component can be applied.

Black Box Testing For Component Based Development Environment:-

If we are applying the black box testing then there is no need of considering the code of the component. But in that case the complete set of components will be considered to calculate the complexity of the component. With the help of black box testing, the complexity can be calculated up to approximate level.

White Box Testing For Component Based Development Environment:-
White box testing considered the detailed investigation of the code for calculating the complexity of the software. This white box testing can be modified for usage in component based development environment by first calculating the Package cohesion component complexity metric [10]. As we know for the good development we required the low coupling and high cohesion therefore the components having the low coupling and high cohesion will get easily selected by PC3M and then the white box testing can easily be applied only to those components to get the result better. So the optimization of code only in selected components will be done in a proper way.

Grey Box Testing in Component Based Development Environment:-
In the grey box testing the tester has the ability to deal with the some of the internal code but not for each and every component.so after selecting the components having low coupling and high cohesion, using the grey box testing the components best suited for development of software will get selected.

Role Of PC3M Metric In Other Types Of Testing:-
Unit testing:-
Unit testing deals with the individual component testing so if the components with better quality can be find out using the PC3M metric then only on that component the unit testing can be applied to get the better results.

Integration Testing:-
This is a type of testing which can be applied after integration of various modules. The modules which could be integrated could be determined using the package cohesion component complexity metric. After calculating the value only the components with the better values could be integrated and the performance of the system will be increased.

System Testing:-
System Testing is that type of testing which could be applied to whole system. System testing would require less number of efforts if the components with the good quality are involved in the tested system.

Acceptance testing:-
Acceptance testing is related to the acceptance of end user. If the end user is not accepting the product then the components with approximate values of Coupling and cohesion can be taken for further development of software.

Load Testing:-
Load testing is done to run an application under heavy loads but if the components are selected with low coupling and high cohesion then the system may run under the heavy loads.

Performance Testing:-
Performance testing is that type of testing in which the performance of the system is checked and if the components are selected with the cow coupling values with the help of metric then the good performance cab be achieved by the system and performance testing will gives us the good result.

References:-

1. S. Kebir, I. Borne, and D. Mestali, "A Genetic algorithm based approach for automated refactoring of component based software", Information and Software Technology, vol. 88, pp. 17-36, 2017.
2. R. K. Bawa, and I. Kaur, "Algorithmic approach for efficient retrieval of components Repositories in component based Engineering", Indian Journal of Science and Technology, vol. 9, pp. 1-13, 2016.
3. T. Vale, I. Crnkovic, E. Almeida, Y. Cavalcanti, and S. Meira, "Twenty-eight years of component based software engineering", The Journal of Systems and Software, vol. 111, pp. 128-148, 2016.
4. M. O. Khan, A. Mateen, and A. R. Sattar, "Optimal performance model investigation in component based software engineering", American Journal of Software Engineering and Application, vol. 2, pp. 141-149, 2013.
5. A. Mandal, and S.C. Pal "Emergence of component based software engineering", International Journal of Advanced Research in Computer Science and Software Engineering, vol. 2, pp. 2-6, 2012.
6. S. R. Chidamber, and C. F. Kemerer, "A Metric suite for object oriented design", IEEE Transactions on Software Engineering, vol. 20, pp. 476-449, 1994.
7. H. Washizaki, H. Yamamoto, and Y. Fukazawa, "A Metric suite for measuring reusability of software components", Proceedings of the Ninth International Software Metrics Symposium, pp. 1-13, 2003.

8. S. ali, S. Ghafoor, and R. A. Paul, "Software engineering metrics for COTS based systems.", IEEE Computer Journal, vol. 34, pp. 44-50, 2001.
9. J. Chen, W. Yeap, and S. D. Bruda, "A Review of component coupling metrics for component based development", World Congress on Software Engineering, vol. 4, pp. 65-69, 2009.
10. P. Goswami and J. Sharma "Package Based Cohesion measurement in component based software development", International Journal of Engineering and Technology, vol. 9, pp. 3172-3182, 2017