Test Case Minimization Strategies in CBSE Models: A Review

Zenith Pankaj, Dr. Sukhdip Singh

Abstract—**Software systems evolve into complex systems every day. Software engineers usually use regression testing over the models detect whether faults introduced into Component Based models. The testing technique works by to doing all existing test cases, but technique however require a lot of time and efforts, depending on the size and complexity of the component based system under test. This paper reviews some of the test case minimization strategies that software engineers use in CBSE Models**

Index Terms— **Test Cases, Component Based Software Engineering, Test case Minimization, Regression Testing.**

I. Introduction

A software component is a software building block that is confirms to a component model and it can be independently composed and deployed without modification according to the composition standard [1].
Component-based software is a collection of self-contained and loosely coupled components that allow plug and play [2]. The components may have been written in different programming languages and executed on different operational platforms, and distributed across geographic distances. Some components may be developed in-house, while others may be third-party or commercial off the shelf (COTS) components, with the source code unavailable.

Properties of Component-Based Software

A. Source code availability

When developing component-based software, developers prefer, rather than implementing the code, to adopt COTS components whenever suitable components are available [3, 4].

B. Distribution

With the development of Internet, more and more component-based software is distributed across networks. Different components will be designed, developed and integrated in distributed environment.

C. Reusability

One of the main objectives of component-based software engineering is to promote software reusability, so that it can improve the quality of future products and at the same time, reduce development costs.

***Zenith Pankaj**, Computer Science & Engineering, DeenbandhuChhotu Ram University of Science and Technology.Sonipat, India,9034222352*
***Dr. SukhdipSingh(Assistant Professor)**, Computer Science & Engineering, DeenbandhuChhotu Ram University of Science and Technology. Sonipat, India.*

II. Component Based Software Testing

To ensure the delivery of quality software, effective and efficient testing is a key part of software development. Test methodologies [5] are generally divided into two types: black box and white box. Black-box approach, such as random testing and functional testing, in which do not require the knowledge of implementation details, but when applied to CBS systems they may encounter a problems similar to those that found in testing of the traditional programs. In white-box approach, internal structure of the program is tested, so in this testing, firstly the program structure is examined and then various test cases are derived based on the program logic.

Difficulties in adequate testing and maintenance for component-based software

There are some major factors that affect our testing and maintenance activities.

A. Code availability

For COTS components, source code is very often not available, and the interaction among components will have to go through predefined component interfaces. The lack of source code causes a lot of problems.

B. Performance and reliability analysis

Performance is one of the quality features that are heavily affected by component-based software features. Reliability and usability also need to be reexamined. To analyze these quality features of component-based software, a key issue is how to reuse the results provided by the software components.

C. Adequacy

Test adequacy is one of the toughest issues in testing component-based software. On one hand, not all traditional test adequacy criteria can be used, especially for those source code–dependent criteria. On the other hand, even though some black-box based criteria can be adopted, the issue is: Do we still want to apply those criteria?

D. Maintenance

When a component in a component-based software is modified or upgraded, a maintenance activity occurs [6, 7]. Due to many of the characteristics of component-based software, difficulties can be encountered when traditional maintenance approaches are applied. The cost of maintenance phase for conventional software as two-thirds of the total cost and it can be still more for maintaining CBS.

III. Component-oriented software test tools

Recently, a number of software test tools have been developed to support test automation of software components. Table I shows some different Types of Test Tools and Tools Vendors[8]:

Table I: Different Test Tools and Their Vendors

Types of Test Tools	**Test Tool Vendors**	**Test Tools**
Problem management tools	Rational Inc.	ClearQust, ClearDDTS
	Microsoft Corp.	PVCS Tracker
Test information management tools	Rautional Inc.	TestManager
Test suite management tools	Rational Inc.	TestFactory
	SUN JavaTest,	JavaHarness
White-box test tools	McCabe & Associates	McCabe IQ2
	IBM	IBM ATC
Test execution tools	OC Systems	Aprob
	Rational Inc.	Visual Test
	Mercury Interactive	WinRunner
Code coverage analysis tools	Case Consult Corp.	Analyzer, Analyzer Java
	OC Systems	Aprob
Regression testing tools	IBM	Regression Testing Tool

IV. Regression Testing for Component-Based Software

Regression testing, which is aimed at ensuring that the modified software still meets the specifications validated before the modification activities, is critical to the success of a component-based system. Regression testing [2] involves many different issues, for instance, test-suite management and regression-testing automation.
Regression testing for component-based software is significantly different from regression testing for traditional software. This is mainly because traditional software maintenance is usually carried out by the same team; therefore, the team has a full knowledge of the software. For component-based software, maintenance is often carried out by component providers. After that, users have to maintain their systems according to the changes that the component providers have made. Component providers have full control of their components, and can therefore use traditional approaches to maintain their components[9,10].

Need of Component Based Regression Testing

A. Testing is a critical activity which occurs during the maintenance stage of the software lifecycle. However, it requires large amounts of test cases to assure the attainment of a certain degree of quality. As a result, test suite sizes may grow significantly.

To address this issue, Test Suite Minimization and Reduction techniques have been proposed. However, suite size reduction may lead to significant loss of fault detection. To deal with this problem, many Algorithms have been Suggested in the Literature for example Genetic Algorithms, Greedy Algorithms and Heuristic Based Approaches.

V. Future Scope

1) Component based Testing can be performed using Various tools available in Various Languages, For Example JUNIT for Java under Netbeans Environment.
2) The Main task of this work is to Test all the components in a selected System as a whole or in other words the goal is to test the Integration of the components in the system.
3) The first phase is to build Components of the system which can be testing via Tools such as Junit.
4) The main task in the second phase is to design and generate component test cases for the constructed test models. An automated tool/program can be used for generating of test cases.
5) Finally these components can be tested for Integration after applying Test Case minimization algorithms.
6) Comparison of the Previously Generated Test Cases and Minimized Integrated Test cases will be Performed for Detection of Fault Tolerance and hence generation of Optimal Test cases.

VI. Conclusion

From this survey report it has been concluded that component-based software integration testing has some challenges that is: i) the component distribution with no source code, ii) heterogeneity of technology and the specification, iii) data can be lost across an interface due to informal and incomplete interface specification, iv) complicatedness to find out dependencies between components, v) inconsistency between the modules for instance improper call, vi) integration between modules not give the desired output, vii) data types and their valid range mismatch between different modules.

In general there is a lack of tools, method and strategies that covers the integration testing problem as a whole: from defining the integration order to test case selection.

It can be concluded that if we use proper tool, methods and strategies then overcome the component-based software testing challenges. We use an approach to component and their interactions modeling using UML interaction diagram, component and their interface are specified by using UML diagram.

REFERENCES

[1] Pravin, Albert, and SubramaniamSrinivasan. "EFFECTIVE TEST CASE SELECTION AND PRIORITIZATION IN REGRESSION TESTING." Journal of Computer Science 9, no. 5 (2013): 654.

[2] GaoTao, Chuanqi, Bixin Li, and Jerry Gao. "A Systematic State-Based Approach to Regression Testing of Component Software." Journal of Software 8, no. 3 (2013): 560-571.

[3] Maragathavalli, Mrs P., and S. Kanmani. "Test Suite Minimization using Hybrid Algorithm for GA generated Test Cases." INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY 6, no. 1 (2013): 279-286.

[4] Parsa, Saeed, and AlirezaKhalilian. "On the Optimization Approach towards Test Suite Minimization." International Journal of Software Engineering and Its Applications (IJSEIA) 4 (2010): 15-28.

[5] Pressman, R., "Software Engineering: A Practitioner's Approach", New York: McGraw-Hill, 2001.

[6] Cheesman, J. and J. Daniels, "UML Components: A Simple Process for Specifying Component-Based Software", Reading, MA: Addison-Wesley, 2001.

[7] Sparling, M., "Lessons Learned Through Six Years of Component-Based Development", Communications of the ACM, Vol. 43, No. 10, pp. 47–53, October 2000.

[8] Weyuker, E. J., "Testing Component-Based Software: A Cautionary Tale", IEEE Software, Vol. 15, No. 5, pp. 54–59, September/October 1998.

[9] Cook, J. E., and J. A. Dage, "Highly Reliable Upgrading of Components", International Conference on Software Engineering, Los Angeles, CA, pp. 203–212, 1999.

[10] Orso, A., et al., "Using Component Metacontents to Support the Regression Testing of Component-Based Software", Proc. of IEEE International Conference on Software Maintenance (ICSM2001), pp. 716-725, 2001.

[11] Gallagher, Leonard, and Jeff Offutt. "Test sequence generation for integration testing of component software." The Computer Journal 52, no. 5 (2009): 514-529.

[12] Heineman, G. T. and W. T. Councill, "Component-Based Software Engineering: Putting the Pieces Together", Reading, MA: Addison-Wesley, 2001.

[13] Sun Micro, "Enterprise JavaBeans Technology", http://java.sun.com/products/ejb/, 2002.

[14] Microsoft, "COM+ Component Model", http://www.microsoft.com/com, 2002.

[15] Object Management Group, "CORBA Component Model Joint Revised Submission", 1999.