ISSN No. 0976-5697

Volume 2, No. 2, Mar-Apr 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Introduction of Component Based Testing

Vikas Verma*
Department of Computer Science & Engineering
University Institute of Engineering &Technology
Kurukshetra University, Kurukshetra
Haryana, India
vik.ver86@gmail.com

Dr.Sona Malhotra
Department of Computer Science & Engineering
University Institute of Engineering &Technology
Kurukshetra University, Kurukshetra
Haryana, India

Gurbaj Singh
Department of Computer Science & Engineering
University Institute of Engineering &Technology
Kurukshetra University, Kurukshetra
Haryana, India
gurbajsingh86@gmail.com

Divyendu Kumar Mishra
Department of Computer Science & Engineering
University Institute of Engineering &Technology
Kurukshetra University, Kurukshetra
Haryana, India
ratan01mishra@gmail.com

Abstract: One of the most common reasons for adopting component-based approaches is reuse. This means to build software from existing components by assembling and replacing interoperable parts. This reduces development time and improved product quality makes this approach very attractive. This Report introduces basics of Component Based Testing, fundamental technical skills and the supporting skills needed by successful software tester. Software development styles have changed a lot of times over the past few decades catering to the needs of the era, which they represented. With increasing pressures on time and money, the concept of component based software development originated. In this method, the software project is outsourced to other development organizations and finally, the third party components are integrated to form a software system.

Keywords: Component, Reusability, Test components, Component Based Testing

I. INTRODUCTION (COMPONENT BASED SOFTWARE ENGINEERING)

Component Based Software Engineering (CBSE) emphasizes the separation of concerns in respect of the wide-ranging functionality available throughout given software system. Software component can be deployed independently and subject to composition by third parties.
Some of the component characteristics which are relevant during their testing-

A. Component Observability: The ease with which a component can be observed in terms of its operational behaviours, input parameters and outputs. The design and definition of a component interface thus plays a major role in determining the component's observability.
B. Component Traceability: It is the capacity of the component to track the status of its attributes and behaviour. The former is called behaviour traceability where the component facilitates the tracking of its internal and external behaviours and the latter is called Trace controllability which is the ability of the component to facilitate the customization of its tracking functions.
C. Component Controllability: This shows the controlling ease on a component's inputs/outputs, operations and behaviours.
D. Component Understandability: This shows how much component information is provided and how well it is presented.[1][2][7]

II. TESTING SOFTWARE COMPONENTS

When to test a component: One of the first issues in testing software components is whether all that effort is required in the first place or not. When is it ideal to test a component in a system? If it is seen that the results of the component not working is greater than the efforts to test it, then plans should be made to test such a malfunctioning component.

A. Which components to test: When risk classification of the use cases is mapped onto components, we find that not all components need to be tested to the same coverage level.
B. Reusable components: Components that are used for reuse should be tested over a wider range of values.
C. Domain components: Components represents significant domain concepts should be tested for correctness and for the faithfulness of the representation.
D. Commercial components: Components sold as individual products should be tested for reusable components and for potential sources of liability.[3][4][9]

III. PROBLEMS IN SOFTWARE TESTING COMPONENTS

The focus now shifts to the most important problem of component software technology i.e. the problem of coming up with efficiently testing strategies for component integrated software systems.

A. Building reusable component tests: Current software development teams use an ad-hoc approach to create component test suites. Also it is difficult to come up with a uniform and consistent test suite technology to cater to the different requirements like different information formats, repository technologies, database schema and test access interfaces of the test tools for testing such diverse software components. With increasing use of software components, the tests used for these components should also be reused. Development of systematic tools and methods are required to set up these reusable test suites and to organize, manage and store various component test resources like test data and test scripts.

B. Constructing testable components: The definition of an ideal software component says that the component is not only executable and deployable, but it is also testable using a standard set of component test facilities. Designing such components becomes difficult because such components should have specialized and well defined test architecture model and built-in test interfaces to support their interactions to the component test suites and test-beds.

C. Building a generic and reusable test bed: There is a lot of difficulty of developing a testing tool or a test bed technology that is capable to test the system, which has components that use more than one implementation languages and technologies.[5][6][10]

Component Based Testing is a new approach to Software Testing based on the idea of creating Test Cases/Scripts from highly reusable Test Components. The main concept underlying this approach is to design new Test Cases for the particular Application-Under-Testing by assembling and configuring pre-existing application-independent Test Components.

[a] A Test Component is a reusable and compose test unit, providing test services through its contract-based interfaces.

[b] Test Components are cohesive, building blocks which encapsulate and offer Test services. A Test Service is a compound of test functions (i.e. atomic test cases) to accomplish a well-defined test goal.

[c] Test Components are context-independent and highly-configurable, so that they can be reused in multiple test projects within multiple, also heterogeneous, applications and scenarios.

By this approach we can create a new Test Case/Script by selecting and gluing together the Test Components that, through their offered Test Services, concur to satisfy the test case's goal for the specific Application-Under-Testing. Once assembled, Test Components must be only configured with the specific application and test data, in order to move from an application-independent context to the "application-dependant" one. [7][8][3]

IV. MAIN LEVELS IN COMPONENT BASED TESTING

A. Unit testing.
B. Integration testing.
C. System testing.

V. COMPONENT BASED TESTING METHODS

A. Acceptance Testing.
B. Regression Testing.
C. White Box Testing.
D. Black Box Testing.
E. Grey Box Testing.

Testing Software Components-

[a] Involves
[i] Testing individual components.
[ii] Testing interaction among components.
[b] Necessity
[i] Inconsistent infrastructure and environment
[ii] Inconsistent interaction model
[c] Challenges
[i] Lack of source code availability
[ii] Test Adequacy criteria[9][10]

VI. CONCLUSION

Efficient testing strategies need to be made for testing domain specific component software and developed tests can be stored to be reused later. If metadata is considered to be a potential solution to the problem of component testing then Metadata standard creation will need a lot of cooperation and coordination among the various third party component producers around. Reliability of components can be improved by improving the languages used to implement them. Apart from automation of test cases, there is a need for sequencing and prioritization of test cases.

VII. REFERENCES

[1] Szyperski Clemens, Pree Wolfgang, Pomberger Gustav, Broy Manfred and Deimel Anton, What characterizes a (software) component? (1998), www.citeulike.org.

[2] Ghosh Sudipto and Mathur Aditya, Issues in testing Distributed Component based Systems, Software Engineering Research Centre, Department of Purdue University, March22,1999, portal.acm.org/citation.

[3] Szyperski, Component Software, ACM Press, Addison-Wesley (1999).

[4] Bhor Adrita, Software Component Srategies, June 2001, www.ajevans.com.

[5] Brucker D. Achim and Wolff Burkhart, Testing Disributed Component Based Systems using UML/OCl, www.brucker.ch.

[6] Gao Jerry Zeyu, Tsao H.-S.Jacob and Wu Ye, Testing and quality assurance for component based Software, books.google.co.in

[7] Jalote Pankaj, An Integrated Approach to Software Engineering.Narosa press, New Delhi (2004)page no.144.

[8] Patton Ron, Software testing.Sams press, Pearson Education (2004)page no. 71-86.

[9] Pressman Roger S., Software Engineering. A Practitioner's Approach, McGraw Hill Higher Education (2001)page no. 721-742.

[10] Von Vorgelegt, Tracing Crosscutting Requirements for Component-Based Systems, 22. June 2005, Berlin 2005, deposit.ddb.de/cgi-bin.