Issue and Challenges in Component Testing in Component Based Software Engineering

Praveen Kumar
Research Scholar,
Dr. A. P. J. Abdul Kalam University, Indore, MP, India

Abstract: Software development is not an easy affair. When it's come to develop large and complex software there is a need to use reusable components by the means of Component based Software Engineering (CBSE) Framework. In Component Based Software Engineering, component is basically developed in two categories i.e. Commercial Component and In-House Component. Developing and using reusable components is not an easy task for the software industry. To develop high quality software, various testing methods can test but when it comes to test any software that is developed using a reusable component then there is a requirement to first test the component itself for its quality. In this article the main focus is on the issue like controllability, observability, understand ability and challenges like develop reusability test case, test drivers etc in component testing in component based software engineering.

Keywords: Component Testing, Software Testing, CBSE, Commercial Component, In-House Component.

Introduction

Software development is not an easy affair. When it's come to develop large and complex software there is a need to use reusable components by the means of Component based Software Engineering (CBSE) Framework. Component-Based Software Engineering (CBSE) is an best form of Software Engineering that offers the feature of reusability. The reuse of software components makes CBSE a specialized paradigm of software development. CBSE stands on the philosophy of "the buy, don't build". Gaedke [1] defined Component-Based Software Development (CBSD) as "Component-Based Software Development (CBSD) aims to develop large software systems from previously developed components (which in turn can be constructed from other components)". CBSD methodology promotes the development of software systems by picking suitable and apposite pre-existing, prebuilt and reusable (off-the-shelf) software work products called 'components' and integrating those components with a pre-defined architectural design. CBSD promotes the thought of integration of heterogeneous, context independent, and pre-designed components to develop software applications. Component Based Software applications are developed by the integration of reusable, pre-existing as well as new components which are integrated through error-free interfaces. The objectives of Component-Based Software Engineering are to develop extremely large and complex software systems by integrating 'Commercially Off-the-shelf Components (COTS)', Commercial Components, In-House Components, third-party contractual components, and newly developed components to minimize the development time, effort, and the cost.

Software testing is one of the important concepts in software development process to ensure quality of the software. When it comes to test any software that is developed using a reusable component then there is a requirement to first test the component itself for its quality. In Component Based Software engineering there are issues and challenges in component testing.

Issues in component testing

Lack of testing tools available for component testing, as of now there are no any software tools available to test reusable components.

Understandability of a component is not an easy task, it depends on how much information is provided by the developer of the component to understand it better in the form of documentation, user manual, application interface specifications, component analysis, design, testing, and other specification and also how much information is provided of component test suites, component acceptance test plan and test metrics, and quality report. Component developer hides most of the information which is essential while using the component in a software project.

Tracing component attributes and component behavior is not an easy task while using reusable components in software development. Integration engineers face difficulty in tracking and monitoring the internal and external behavior of a component.

Observing the behavior of the program in terms of operational input and output is not easy. Component interface for incoming and outgoing affecting its observation label before and after integration of the reusable component.

Controllability capability is not facilitated by the developer of commercial components and In-House components which makes integration difficult. It is useful and necessary for the complicated component to integrate with Component Based Software Engineering and may also affect the testing of the software.

Portability – In this component is concerned with both transferring codes between frameworks and running code inside a set of working conditions. Most of the developers are concerned over how much a component depends upon other software, hardware, and other context and degree of visibility.

Interoperability – Component can connect and exchange information with one another easily but the potential difficulties include in adopting a component is an architectural mismatch in this component not able to meet the architectural constraint, functional requirement not matching up to the mark, and quality of the software also compromised when developers integrated several components into one software.

Challenges in Component Testing

Construct component test suites – developers of commercial components do not provide any test suites to test for acceptance of the component and any quality report that will help to check the quality of the component in terms of input and output behavior which may vary from software to software in which this component to be integrated. So, it is difficult to develop any test suites. The component integrator has to spend lots of time understanding the component and preparing test suites for acceptance testing.

Reusability support for component testing – the success of the reusable component depends upon how efficiently the component test is written that can be used across different software projects. It is difficult to develop a reusable test plan where every project may have a different interface, programming language, database schema, technologies, and design pattern.

Develop Testable component – Success of reusable component not only depends on its executable and deployable capability but also depends upon component test facilities. Testable components need to include an interface for test script, test data, and test case that interact with the testing facilities. It also includes a test setup interface that can be interacted with the test suite automated system, configure tests for component testing, test execution tools, and test report generation tools that will test components and generate report of test conducted.

Develop software component test drivers and stubs in a well-structured manner – It generally based on specific module requirements and design specifications this leads to the developed test drives and stubs used in specific projects only. The traditional way of developing and using test drivers and stubs is ineffective and causes an increase in the cost of development for each project. This is one of the major challenges in the development of test drivers and stubs in a structured manner. The solution to this problem is that the test drivers and stubs must be customizable according to the given environments by the developer.

Develop generic and reusable test-beds – In this testing of function, class, or library, test execution, test result, or report in an isolated environment for a particular component or module. The challenges are to develop common component test beds for different languages and technologies.

There are other few challenges like

Different programming languages support – Component development in one language and used in another language is a major challenge in component based software engineering. This can be solved by interoperability features of the software development environment where components developed in one language can be used in other languages without much problem. For example .Net Framework based software development environment provides such support that developers can develop components in one language that can be used in other languages.

Component Functionality – A component may have various functionalities that might not be mentioned in documentation by the component vendor that they sell that need to be explored by the component integrator. In this case, proper testing of the component functionalities is not possible and may generate wrong test reports.

Code Inaccessibility – As source code is not provided by the component vendor this makes white box testing difficult to do. In white box testing generally, code level testing is done. The data flow of the component needs to know to determine the test requirements.

Minimum information provided by the component vendor – Information providing features by the component seller is difficult as the component may be integrated into different environments and there is no quality way to provide information for a component.

Conclusion

Component Based Software Engineering (CBSE) faces issues and challenges in testing components by the component integrator due to a lack of testing tools, traceability, and controllability features are also missing, portability and interoperability features are not supported by the different developing environments, challenges like developing reusable test suites, developing test component, developing test drivers and stubs, code inaccessibility and minimum information provided.
The solutions to these issues and challenges would be to increase the development of reusable components by the usage of portable and interoperable developing environments like Java, .Net Framework, etc., by maximizing the features of the testing tools, reusability support for component testing, and developing generic and reusable test-beds.

References

[1] M. Gaedke and J. Rehse, “Supporting compositional reuse in component-based web engineering”, In Proc. ACM symposium on Applied computing (SAC ’00), ACM Press, New York, NY, USA, pp. 927–933, (2000).

[2] Jerry Gao, and Youjin Zhu, "Tracking Software Components", Technical report (http://www.engr.sjsu.edu/gaojerry/techreport/) in San Jose State University, 1999.

[3] Gao, Jerry. Component testability and component testing challenges.2000.

[4] Roger S. Pressman, Software Engineering: A Practitioner's Approach (fourth edition), McGraw-Hill, 1997.

[5] Atkinson C, Bayer J, Bunse C, Kamsties E, Laitenberger O, Lacqua R, Muthig D, Paech B, Wust J, Zettle J. Component-Based Product Line Engineering with UML, Addison Wesley Series on Component Software, 2002.

[6] Bachmann F, Bass L, Buhman C, Comella DS, Long F, Robert J, Seacord R, Wallnau K. Volume II Technical Concept of Component-Based Software Engineering. SEI Technical Report NO.CMU/SEI2000-TR-008, 2000.

[7] Cmkovic I. Component Based Software Engineering: New Challenges in Software Developrnent, 2003.

[8] Gao J, Gupta K,. Gupta, Shim S. On Building Testable Software Components, in J. Dean, and A. Gravel Eds,, proceedings of ICCBSS 2000, LNCS 2255, 108-121, 2000.

[9] Gill NS, Tomar P. Impacts of Inadequate Testing and Testing Infrastructure in Component-Based Software Testing and Development. Proceedings of TIS C: International Conference on Trends in Information Sciences and Computing, Chennai, India, 687-690, 2007.

[10] Jerry G. Component Testability and Component Testing Challenges, San Jose University, CA, 2000.

[11] Sparling M. Lessons Learned-Through Six Years of Component-Based Development. Communications of the ACM Journal, 43(10), 2000.

[12] Tracz W. Confessions of a Used Program Salesman: Institutionalizing Software Reuse, Addison-Wesley Longman; 1st Edition, 1995.

[13] Won K. On Issues with Component-Based Software Engineering. Journal of Object Technology 4: 45-50,2005.

[14] Tomar P. Analysis, design and development of component-based reusable models and testing processes, Ph.d. thesis, Dept. of computer Sci. & applications Maharshidayanand University Rohtak, Haryana, India, 2010.