ISSN:2229-6093

Saurabh Rawat et al, International Journal of Computer Technology & Applications,Vol 8(3),334-337

A Contemporary Approach to Component-Based Software Testing

Saurabh Rawat, Rajesh Kumar

Graphic Era University, Dehradun, India

E-mail:- rawatsaurabh.777@gmail.com

Abstract

Testing is a vital action encouraging productivity, not negligible, in the sphere when Software Development community is enveloped by Component Based Software Development. Testing constitutes more than fifty percent of the expense of Software Development, resulting in increased cost.This paper describes the issues and challenges of component-based systems. This paper also suggests the requirement of a novel approach to testing of component-based software. It also proposes a novel approach to testing of component-based software while considering some important factors like component study, component test case design, component test execution, and component test analysis and component test documentation.

Keywords - *Component-based development, Software Component Testing, Issues and Challenges in Component-Based Software Testing*

1. Introduction

Successful Software Development bank on software testing, whether software is positioned on any traditional or condition oriented access. Quality and authenticity, non functional traits, of a software product is ameliorated by Testing. Software component testing is an approach to finding mistakes, diminishing expanse, elaborating reliability, and strengthening the quality of software components [1, 2].

Testing is the most important activity of the software development process for finding the maximum errors; therefore without proper testing of the software product, all the efforts will be in vain [3, 4, 5]. Component-based testing is a highly complex activity because component-based development involves not only the reusable entities called components but also their interactions and integration overheads. In the conventional approach to software development, the testing process is applied at the later stages of the development; usually testing is treated as a single-phase activity [7]. But in component-based development, testing is applied not only to the individual components but also to the whole integrated software system. This process not only fulfils the aim of finding errors but also improves the software quality [6].

I. SOFTWARE-COMPONENT TESTING

The Component-Based Software Development (CBSD) approach aims at delivering quality in large software system by selecting and integrating already existing reusable and testable component. Rather than development from scratch it gathers the existing components which results in low development and maintenance cost. Component-Based Software Testing (CBST) of component is maturing as a new sub-discipline of Component-Based Software Engineering (CBSE). “CBST is a process used to identify the correctness, completeness, quality and documentation of developed component. So, moving towards this direction this study proposes two component-based test processes [174]. First is modified component-based test process documentation and second is a new process to construct testable component [175].

Quality assurance and productivity can be achieved through software testing process. With the advancement of technology and increase in appetite to get more, the complexity of software development is increased. This advancement has made distributed development of applications an important concern for the focus of the developers. Quality is one of the most important concerns during development of an application. To achieve the quality in distributed applications, developed using components, a highly efficient and resourceful component–based testing is required. Different aspects of the components like reusability, security, reliability, truthfulness, and safety etc. are required to be tested effectively. Study demonstrates that more than half of the development cost is a result of testing performed. This cost increases with the increase in complexity of the software. In CBS, development process follows component-based approach that mainly consists of encapsulated data and functions within the modules as unit of testable components. These components are configured at run

time through parameter passing [176,177]. Our first main concern in this chapter is to improve component-based test processes documentation while considering two factors viz. requirement in CBST and test case process documentation; and second is testable component construction through a new and novel process.

Software-component testing represents a group of activities, which involves component study, quality test design and generation, test execution, fault detection, and finally testing evaluation.

Some widely-known definitions of software component testing follow.

- According to Sommerville [7], "Component (or unit) testing is the activity in which individual components are tested to ensure that they operate correctly. Each component is tested independently and correctly, without other system components gaps and errors."
- According to IEEE Standard Glossary [8] "Component testing is the testing of individual software components or groups of related components."
- According to Gao et al. [9], "Software component testing refers to testing activities that uncover software errors and validate the quality of software components at the unit level."

A. *Issues and Challenges in Component-Based Testing*

After analysis of the above-mentioned definitions, we have identified some challenges, issues and limitations in software component testing, which are as follows.

- Component based software process of authentication and acceptance contradicts from traditionally developed systems [10].
- Component-based software testing is defined as an activity of an individual component as well as of a component-based development project.
- In component-based development all the predefined components are tested by both vendors and users. A component vendor performs testing activities during early phases of component development whereas a user performs testing activities during application engineering [11, 12].
- Component-based software engineering develops a software product that exhibits higher quality and reduced development cost and time. Still the development of component-based systems has been flooded with problems surrounding the integration and composition of third-party components [13, 14].

B. *Requirement of a Contemporary Approach to Testing of Component-Based Software*

The main concept of component-based development is to develop a new software system by using pre-defined components. These components are context-free, independent entities that are developed with the intent of reusability. Developers can reuse components by making minor or major modifications to make these components compatible with other existing or newly-developed components instead of developing each and every part from scratch [15, 16, 17]. Because of this reusability, it is expected that the cost of delivering a software product by using component-based development will be lower than conventional software development [18].

When the already-developed components are tested along with the fault or error free components and are integrated with other software components, then at times they may not be successful in providing the expected results. Sometimes it happens that some off-the-shelf components do not match with the new components since they are developed to fulfill a different aim [19, 20]. Although software-testing techniques defined in literature [3, 4, 11] verify the compatibility, interaction and context of interconnected components, they do not provide a discrete layer of software-component testing. Therefore, the software development community is facing testing problems in component-based development.

Though conventional software testing can help in software component testing at the unit level, it can not guarantee error-free software as various components are integrated to achieve a software system [21, 22]. Thus there is a requirement of defining a clear and well-defined approach to software-component testing.

II. PROPOSED NOVEL APPROACH TO COMPONENT-BASED SOFTWARE TESTING

The proposed modified approach of component-based testing of a software development process is an integration of various test case designs. It mainly aims at providing effective, error-prone software thus results in delivering a quality product. A well-defined sequence of steps is a road map that led towards software-component testing, a flexible approach that provides and promotes customization. On a parallel

track this modified approach focuses on encouraging reasonable planning and management tracking with the progress in the project. A documentation is followed to define all the testing activities to be performed with the list of possible inputs and outputs. This study presents a sequence of different phases towards the successful completion of testing process and is shown in Figure 6.1. With the rise of CBSE, component testing is also gaining popularity and contributes towards reliable and reusable software system. A comparative analysis is performed between the resulting values and the expected values to enables development of strong and efficient tools and techniques of CBST. Component test specification, execution, and recording may however, be carried out for a subset of the test cases. Component-based documentation is one important step that helps practitioners and researchers to make efficient reusable and testable component.

Following are the steps of testing process.

1. Component Consideration
2. Component Test Case Design
3. Component Test Execution
4. Component Test Analysis
5. Component Test Documentation

1. Component-Consideration

This stride yields a total sketch portraying the fine points of accessible components. The pattern of a test along with different test conditions and familiar outputs is defined

The nitty-gritty of components, preferred from storehouse is also covered.

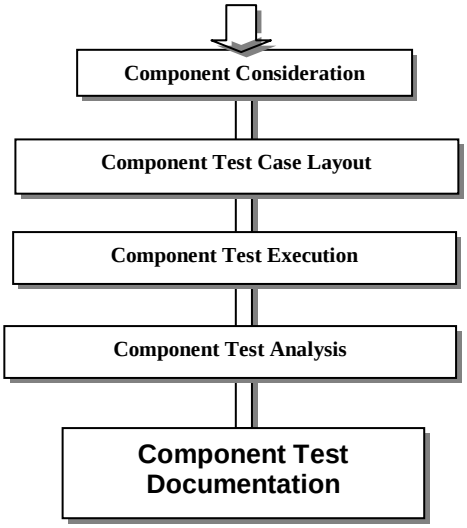

Figure 1. - Phases of novel approach to component-based software testing

The principal objective of component review is to consider the necessary attitude of the component-testing policy, resource fulfillment along with risks and preferences. This layout also interprets the number of test cases included, their periods as well as test completion precedent, risk areas and assigned holdings.

2. Component Test Case Layout

The test cases are properly layout after an intensive consideration of available components. These test cases include a predefined group of conditions and the expected outputs based on these conditions the actual results is evaluated to determine whether a software system is functioning as per the requirements or not. It is also considered as an approach to evaluating the success or failure of a software product.

Conceptually a test case signifies the pre-state, test environment, test inputs, test execution conditions and expected output for the fulfillment of a given test objective. The test-design process is very broad; it involves the criteria for test case selection and the development of a test procedure.

3. Component Test Case Execution

This step is all about executing the test cases in the target testing environment. Each and every test case shall be executed one by one for finding the maximum number of errors. In other words, test execution is the procedure of implementing all the available test cases and then carefully observing the results.

Various roles are included in executing test cases at different testing levels; which are as follows.

- At the unit level, the developers execute the test cases.
- At the integration level, both the developers and testers are involved in the testing process.
- At the system level, though testers are involved, developers and end users may also take part.
- At the acceptance level most of the test cases are performed by end-users along with the partial involvement of the testers.

4. Component Test Analysis

This step is all about examining each test case so that component functions and behaviors can be thoroughly analyzed by checking their expected output and actual input. In this step the test case execution results are compared with the expected output. If execution results and expected outputs are same then it will be

considered a successful test and if otherwise, then developers will perform the follow-up activity to locate the actual faults to be fixed.

5. Component Test Documentation

Test documentation is the process of creating documents. The main purpose of documentation is to help in the estimation of required testing efforts, test coverage and tracking as well as tracing of requirements. After successful completion of each test, a final documentation is prepared.

III. CONCLUSION

The proposed novel approach of component-based software testing focuses on components, which are the smallest units of software design. The conclusion of this paper is that if software community seeks to develop testable components by using the proposed approach then it is easy to produce complex software in less time and cost. The scope of our research is to practically implement the proposed approach on a set of components and to present the analytical results.

References

[1] Bertrand Meyer, Christine Mingins, and Heinz Schmidt, "Providing Trusted Components to the Industry," IEEE Computer, vol. 31, no. 5, pp. 104–105, May 1998.

[2] Bertrand Meyer, "The Grand Challenge of Trusted Components," Proc. 25th Intl Conf on Software Engineering (ICSE 2003), Portland, Oregon, USA, 03–10 May 2003. IEEE Computer Society Press, 2003, pp. 660–667.

[3] Myers, G., The Art of Software Testing, Wiley, 1979.

[4] Nasib S. Gill and Pradeep Tomar, CBS Testing Requirements and Test Case Process Documentation Revisited, ACM SIGSOFT, Software Engineering Notes, March 2007, Volume 32, Number 2.

[5] Szyperski C., (1998). Component Software, Beyond Object-Oriented Programming, ACM Press, Addison-Wesley, NJ.

[6] M. Sitaraman and B. W. Weide, "Special Feature Component-Based Software Using RESOLVE", ACM SIGSOFT Software Engineering Notes 19, No. 4, 21-67, October 1994.

[7] Ian Sommerville, Software Engineering. 9th Edition, Addison-Wesley, March 2010.

[8] IEEE Std 610.12–1990, IEEE Standard Glossary of Software Engineering Terminology, IEEE Standards Board, 28 Sept 1990.

[9] Jerry Zeyu Gao, H.-S. Jacob Tsao, and Ye Wu, Testing and Quality Assurance for Component-Based Software. Artech House Inc., September 2003.

[10] Beatriz Pérez Lamancha, Pedro Reales Mateo, Ignacio Rodríguez de Guzmán, Macario Polo Usaola, and Mario Piattini Velthius, "Automated model-based testing using the UML testing profile and QVT," Proceedings of the 6th International Workshop on Model-Driven Engineering, Verification and Validation (MoDeVVa 2009), Denver, Colorado, USA, 05 Oct 2009. ACM International Conference Proceedings Series, vol. 413, ACM Press, 2009.

[11] Hong Zhu, Patrick A. V. Hall and John H. R. May, "Software Unit Test Coverage and Adequacy," ACM Computing Survey, vol. 29, no. 4, pp. 366–427, Dec 1997.

[12] Paul Baker, Zhen Ru Dai, Jens Grabowski, Øystein Haugen, Ina Schieferdecker, and Clay Williams, Model-Driven Testing Using the UML Testing Profile. Springer, 08 Nov 2007.

[13] Brown, Alan W., Wallnau, Kurt C. (1998): The Current State of CBSE. IEEE Software Journal, September/October 1998, pp. 37-46.

[14] Han, Jun (1998): Characterization of Components. In proceedings of International Workshop on Component-Based Software Engineering, 1998.

[15] Manfred Broy, Anton Deimel, Juergen Henn, Kai Koskimies, Frantisek Plasil, Gustav Pomberger, Wolfgang Pree, Michael Stal, and Clemens Szyperski, "What Characterizes a (Software) Component?" Journal of Software – Concepts and Tools, vol. 19, no.1, pp. 49–56, June 1998, Springer.

[16] George T. Heineman and William T. Councill (Eds.), Component-Based Software Engineering:putting the pieces together. Addison-Wesley, May 2001.

[17] Jeffrey Voas, "Composing software component 'ilities'," IEEE Software, vol. 18, no. 4, pp. 16–17, July/Aug 2001.

[18] Ivica Crnkovic and Magnus Larsson (Eds.), Building Reliable Component-Based Software Systems. Artech House Inc., 2002.

[19] G. Rothermel, M. J. Harrold, and J. Dedhia. Regression test selection for C++ software. *Software Testing, Verification& Reliability*, 10(2):77–109, June 2000.

[20] T. H. Tse and Z. Xu. Test case generation for class-level object-oriented testing. In Proceedings of 9th International Software Quality Week (San Francisco, California, May 21–*24)*, pages 4T4.0–4T4.12, 1996.

[21] C. D. Turner and D. J. Robson. The state-based testing of object-oriented programs. In *Proceedings of the International Conference on Software Maintenance*, pages 302–310, Sept. 1993.

[22] E. J. Weyuker. Testing component-based software: A cautionary tale. *IEEE Software*, 15(5):54–59, Sept./Oct. 1998.