Stepping Towards Component-Based Software Testing Through A Contemporary Layout

Neena Gupta
Department of Computer Science
Kanya Gurukul Campus, Gurukul Kangri vishwavidyalaya
Haridwar, India
Email- neena71@hotmail.com

Abstract— Component- based software development is aimed for developing new software speedily by using minimum resources but outcome the maximum worth. Various components are integrated all together to form the successful software product therefore the quality of new software product completely depends upon these integrated components. To ensure the best quality of overall product, testing of each and every component is highly essential. But problems arise during testing when tester has limited access to the components. This paper discusses the importance of component testing and different problems during testing of such products. Some important strategies are also discussed in this paper and a contemporary layout for Component- based software testing is proposed.

Keywords- Component testing; software testing; testing strategies; component-based software engineering.

I. INTRODUCTION

By reusing already build components, new application software can be developed by component based software engineering using third party components and in house built components. The reusable components are integrated to implement a new application [1, and 2]. So the integration process of the components is very important in component based development and the quality of new developed application depends upon the quality of the components that are used in it.

Component Based Software Testing application is used to reduce the complexity in software implementation. It reduces the delivery time of the product and increases the software productivity [3, 4, and 5].

From the last some decades, demand for high quality software has increased drastically by the customers and end users, and they want their software product to be developed and deployed within time. Also each and every organization wants to have it's our software product within the specified budget. These are some of the reasons that make software developers to think about the component based software development to save time and cost by using already build components [6, 7, and 8].

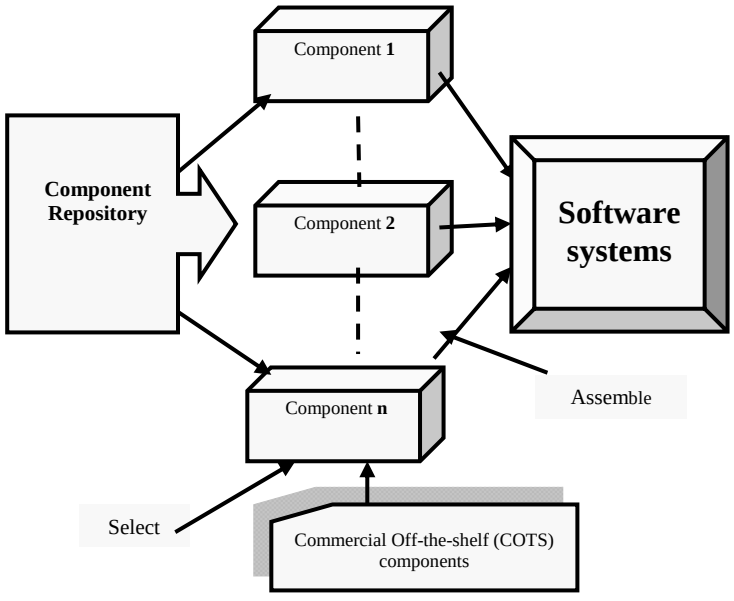

Figure 1. Framework of CBSE

But problem may arise during the testing of such component based product because this type of new product is developed by integrating existing and also newly developed components that were made or used in any other product. These components may not work for certain requirements in the future means these were customized for other application and now these are deployed in another context and deployed in another context in new software product [9]. Author in [14] divided the process of component-based software engineering into five phases. Some software development approaches have therefore emerged to analyze the reliability of component based applications. These techniques mainly fall into two categories [10, 11, and 12].

- **System Level:** System level is very reliable, estimated for the software application as complete.
- **Component Level:** Reliability of the individual component and their interconnection mechanism can be estimated by reliability of individual program.

II. PROPOSED TESTING METHODOLOGY

The Component-Based Software Testing is divided into three phases; Component Unit Testing, Component Deployment in the Real Environment as a Unit, and Component Deployment in Real Environment with other Components.

A. Component Unit Testing

The smallest test unit is the component and this testing is performed by the component developer. The component is initially tested in isolation to detect the errors in core functionality of a particular component. The tester may test the component by any method; white-box testing or black-box testing. This phase of testing does not address the functional correspondence of the tested component behavior to the environment in which it will be assembled later.

B. Component Deployment in Real Environment as a Unit

Unit testing cannot confirm the reliable behavior of the components in the new environment; hence testing by the component user is the next phase of testing and is essential to accomplish an acceptable reliability level. This phase of testing is concerned about the functionality of the component when it is incorporated in a particular environment. It is a type of integration testing and performed by the component user. The goal of this phase is the validation of the implementation of the component that will constitute the end product or system. The user checks whether the component correctly interact with the environment. To achieve this goal, the user monitors the interactions among the actual implementations of the architectural components during the execution of some test cases.

C. Component Deployment in Real Environment with Other Components

Council [13] termed testing by the component user during its implementation in the real environment **Second Party Testing.** In the previous phase we test the individual components in the real environment whereas this phase is concerned about the situation when multiple components work together to provide the system functionality. This phase can be further divided into two sub-phases; first sub-phase deals with the components interaction with other components and also the environment and the second sub-phase concerned about the testing of whole system.

In the first sub-phase, the components user test the interaction of the components as expected in the real environment. It is worth noting that the bugs found by the component user during this phase are not in general found during previous phases.

In the second sub-phase, the component user performs the testing of the system as a whole when all the components are integrated and the entire system is ready to run. The component user tests the load and performance testing. This phase does not require white-box testing, rather, components are tested in the system context to assess the performance of the whole system as a black box.

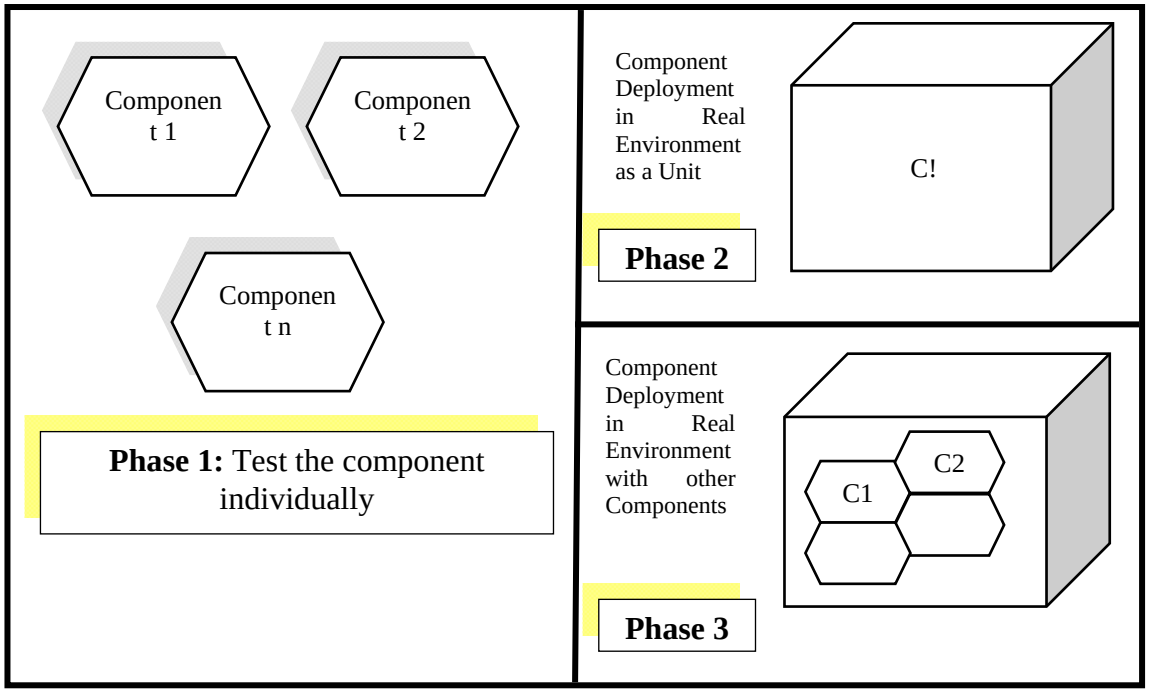

Figure 2: Component-Based Testing Methodology

III. PERFORMIG TESTING

Each phase of testing process consists of six stages; understanding the goal of the phase, understand the data involved, preparing the test plan, execute the test cases, evaluate the test results, and finally preparing the test reports. Only the first stage of every phase is different.

A. Understading the Goal of the Phase

The goal of the first phase is to study the components individually. This step produces an overall outline describing the details of available component(s). Component-study also involves the selection of components from the repository along with their details. The main goal of this phase is to take into account the essential aspects of the component-testing strategy, resource utilization along with risks and priorities. This step also defines the number of test cases involved, their durations as well as test completion criteria, risk areas and allocated resources. The actual implementation of the design provides the information necessary to construct the structural test cases. The language used is an important factor in the quality of the implementation because some types of faults are made impossible by the language and new sets of faults are introduced. For example, the infamous pointer errors are quite different between C++ and Java. The environment used to produce the code can also be useful in testing.

The goal of second phase is to test the functionality of the component when it is deployed in real environment. The actual deployment of the component provides the information necessary to construct the interaction test cases. The component may interact with its environment interact either directly or indirectly through the objects that implement the component's attributes.

The last phase of testing is to study the component's interaction with its environment and other components. Goal of this phase is not only restricted to test components, rather testing the system as a whole. Based on the goal of each phase, next phases are processed.

B. Understading the Data Involved

Many components will interact with external databases as the source of their data. The definitions of the records to store in the database used for testing come directly from the input data permutation tables. A number of separate databases will usually be constructed so that the number of different test conditions that a single database must support is limited. This is particularly important when one test condition, perhaps the presence of a record for a specific person, contradicts another condition, the absence of that person's record.

C. Prepare the Test Plan

This plan will provide information such as the expected levels of specification and code coverage. These values will be used to determine how many test cases to construct. The Planning phase of the component testing method includes an analysis of the component under test (CUT) to produce the test requirements. This analysis can begin as soon as the component's specification is available and continues after the final implementation is

available. The inputs to this phase include specification of the component with pre and post-conditions for each method, the state-transition model for the entire component and the set of object interaction diagrams that constitute the component's functional model. The outputs are test case and test data specifications that have been selected to provide specific levels of product coverage.

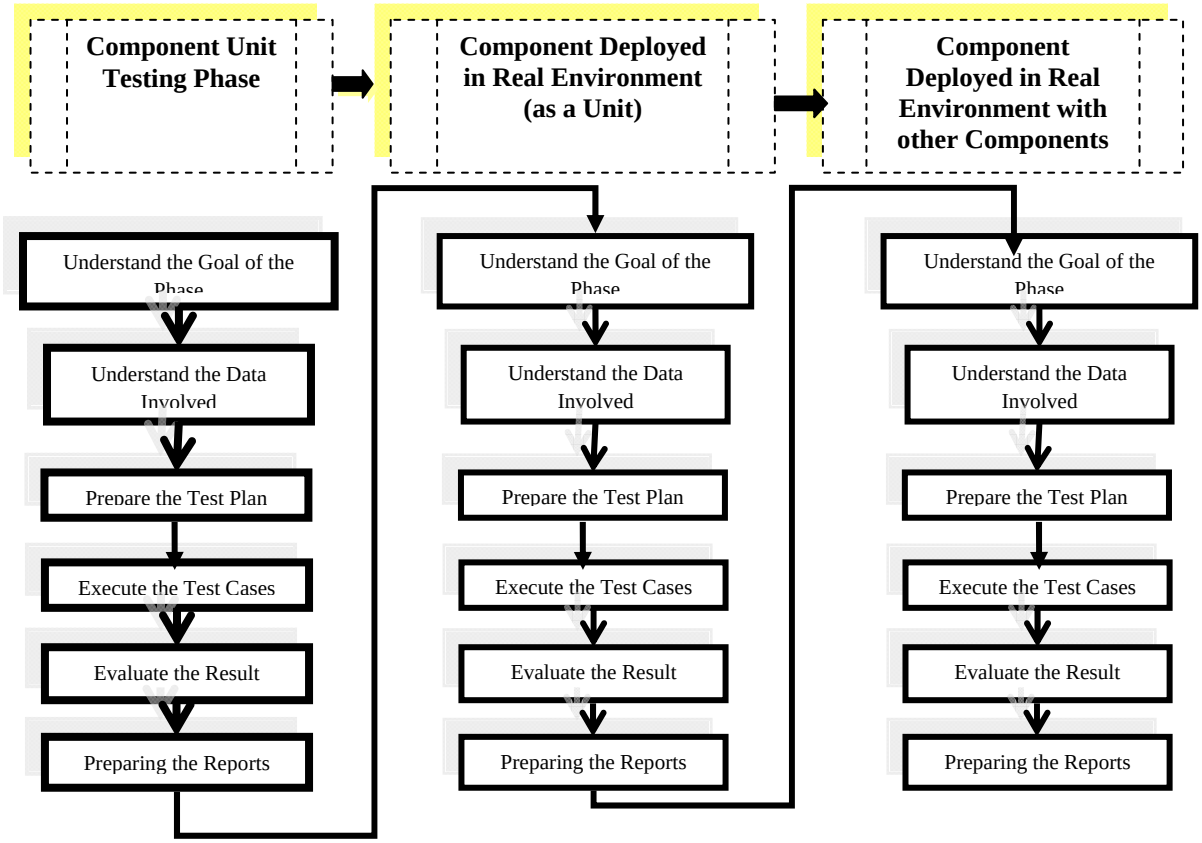

Figure 3: Testing Process

D. Execute the Test Cases

The Execution and Evaluation phase of the component-level testing method includes activities such as applying the test cases to the component under test. Approaches to evaluating the results from the test cases are also included. The inputs to this phase include the plans and infrastructure created in the previous phases. The outputs from this phase include test results to be fed back to the developers as well as information needed to improve the test coverage during the next iteration.

E. Evaluate the Results

The test software should support the automatic validation of as many of the test results as possible. The anticipated results can be *hard coded* in the test case when a specific answer is expected. For test situations where numerous transactions are run against a database, sometimes in different orders, the environment is too complex to determine an exact answer. In this case the test results are evaluated by *comparison* by executing independent queries directly to the database.

F. Preparing the Reports

A checklist enhances quality by ensuring that steps or criteria are not omitted. Table 4 shows a portion of one checklist from a test plan that is testing the GUI portion of each use case for a system. The checklists used in testing a set of components should cover the important non-performance issues for a system.

IV. CONCLUSION

In case of Component-Based Software development there can be many choices possible from amongst the components for a particular situation. Not all of these possible choices can appropriately fit into the system guaranteeing perfect integration with full functionality and performance. Such a situation calls for testing of multiple components at the time for ensuring their fitness for functionality and performance requirements. In this paper the importance of component testing and different problems during testing of such products has been discussed. Some important strategies are also discussed and a contemporary layout for Component- based software testing is proposed considering access limitations to the component. Still lot of work needs to be done in component – based software testing area. Some suggestions are also given which can lead to better testing to deliver a quality component - based software product.

REFERENCES

[1] J. H. Gao, H. S. J. Tsao and Y. Wu, “Testing and Quality Assurance for Component-Based Software,” Artech House, Norwood, 2003.
[2] J. Sametinger, “Software Engineering with Components,” Springer-Verlag, New York, 1997.
[3] C. Ebert and R. Dumke, “Software Measurement,” Springer-Verlag, Berlin, 2007.
[4] V. Tran and D. Liuo, “A Procurement-Centric Model for Engineering Component-Based Software Systems,” 5th International Symposium on Assessment of Software Tools (SAST'97), Pittsburgh, 2007.
[5] J. P. Carvallo, X. Franch and C. Quer, “Determining Criteria for Selecting Software Components: Lessons Learned,” IEEE Software, Vol. 24, No. 3, 2007, pp. 84-94. doi:10.1109/MS.2007.70
[6] A. H. Dogru and M. M. Tanik, “A Process Model for Component-Oriented Software Engineering,” IEEE Software, Vol. 20, No. 2, 2003, pp. 34-41. doi:10.1109/ MS.2003.1184164
[7] M. R. Qureshi and S. A. Hussain, “A Reusable Software Component-Based Development Process Model,” Advances in Software Engineering, Vol. 39, No. 2, 2008, pp. 88-94. doi:10.1016/j.advengsoft.2007.01.021
[8] Jerry Gao, "Component Testability and Component Testing Challenges", Technical report in 1999.
[9] Heineman, G. T.; Councill, W. T. “Component-Based Software Engineering: Putting the Pieces Together” Addison-Wesley, 2001.
[10] Ivica Crnkovic, Stig Larsson and Michel Chaudron “Componentbased Development Process and Component Lifecycle” Journal of Computing and Information Technology - CIT 13, 2005
[11] Weiqun Zheng,”Model-Based Software Component Testing”, 2012
[12] N. Laily Hashim, “Connector-based Integration Testing For Component-based Software”, Monash University, 2008
[13] Council, W. T., “Third Party Testing and the Quality of Software Components”, IEEE Computer, pp. 55-57, August, 1999.
[14] Lata Nautiyal, Neena Gupta, Sushil Chandra Dimri, “A Novel Approach to Component Based Testing”, ACM SIGSOFT Software Engineering Notes, Vol 39, Issue 6, pp. 1-4, 2014.