ISBN: 978-93-83758-09-8

Study of Intelligent Agent for Component Based Software Testing

[1]Priyanka Gandhi, [2]Richa Sharma, [3]Damanpreet Kaur

[1,2]Astt.Prof (HITM),[3]Lecturer (HITM)

[1]priyankagandhi2206@gmail.com , [2]ajricha.sharma@gmail.com , [3]daman1589@gmail.com

Abstract: Component-based software engineering (CBSE) advocates the acquisition, adaptation, and integration of reusable software components, including commercial-of-the-shelf (COTS) products, to rapidly develop and deploy complex software systems with minimum engineering effort and resource cost. Component-Based Development (CBD) offers a radically new approach to the design, construction, implementation and evolution of software applications. Software applications are assembled from components from a variety of sources; the components themselves may be written in several different programming languages and run on several different platforms. To ensure the delivery of quality component based software, effective and efficient testing is the key process in software development. The development of Component-Based Systems introduces fundamental changes in the way systems are acquired, integrated, deployed and evolved. This approach is based on the idea that software systems can be developed by selecting appropriate off-the-shelf components and then assembling them with well-defined software architecture.

Keywords: CBSE, CBD, software, testing.

I. INTRODUCTION

Testing is essential in the development of any software system .Testing is required to assess a system's functionality and quality of operation in its final environment. This is especially of importance for system being assembled from any self contained software components. To ensure the delivery of quality software, effective and efficient testing is the key process in software development. Component based software systems are becoming prevalent at a rapid race, it also increases reusability. It requires the need to adequately test these components based on their application and integration in each system. The unavailability of source code precludes extrapolating standard testing approaches and requires developing procedures that provides each user with sufficient information to allow adequate testing in each usage environment. Module testing in traditional software environment, software is divided into modules which are named and addressable components and these are integrated to satisfy the problem requirement. Modularity is a single attribute that allows a program to be intellectually manageable. Integration testing is an important part of the testing process in which individual software modules are combined and tested as a group. Integration Testing is performed after unit testing and before system testing. Its objective is to identify defects in the interfaces and interactions between components. We use Unified Modeling Language (UML) and its sequence diagram for Integration Test case generation.UML design Model consist of Variety of diagrams. Each diagram describes a view of design. For Example, a class diagram describes a structural view and sequence and activity diagrams describe behavioral views.

II. COMPONENT BASED SOFTWARE ENGINEERING

CBSE is a branch of software engineering which is concerned with development of software systems based on existing in-house and/or COTS components. Reusing previously developed components in developing software has many benefits. Important benefits are reduced cost and time to market. This will definitely increase the quality of a stand-alone component as well as the systems where it is being used. This can greatly help in materializing the benefits of standard domains. A standard domain can always provide well-defined standard components, which can be easily reused in any case. So component technology heavily supports code reuse, which was previously not the case. The component based software engineering is based on the following principle: "Software reuse is the process of creating software systems from existing software rather than building them from scratch." Today the trend in computer-based products, such as cars and mobile phones, is shorter and shorter lifecycles. As a consequence, time spent on development of new products or new versions of a product must be reduced. One solution to this emerging problem is to reuse software design and solutions in new versions of systems and products.

ISBN: 978-93-83758-09-8

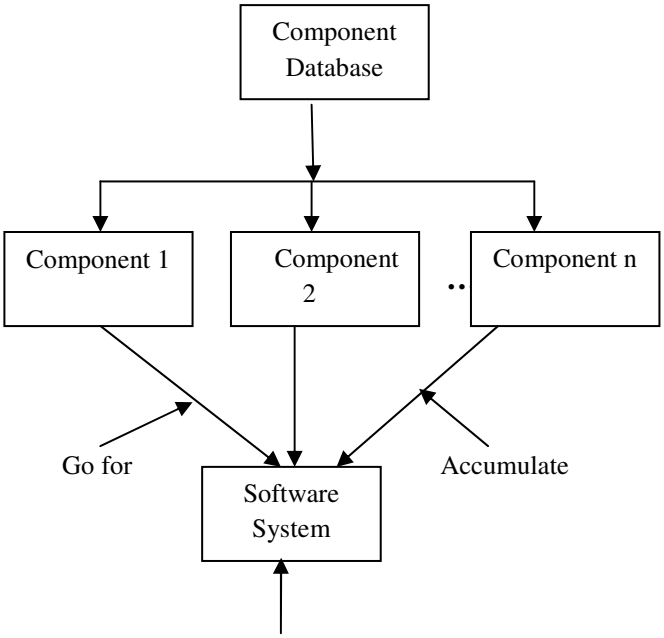

Figure 1: Component selection process

Commercial Off the shelf (COTS) component:

- A component is an independent and replaceable part of a system that fulfills a clear function.
- A component works in the context of a well defined architecture.
- A component communicates with other components by its interfaces.
- Can be developed by different developers, using different languages and different platforms.

The characteristics of the component-based development are the following:

• Black-box reuse
•Reactive-control and component's granularity
•Using RAD (rapid application development) tools
• Contractually specified interfaces
•Introspection mechanism provided by the component systems.

I. Types of Components

- Real-Time Components

In real-time systems, components must collaborate to meet timing constraints. Furthermore, in order to keep production costs down, embedded systems resources are usually scarce, but they must still perform within tight deadlines.

- Common off the Shelf Components (COTS)

Reusing components made for earlier products as an approach to new system development is a promising way of achieving the mentioned development and system improvements. There is also the possibility to buy software components from component vendors, so called Commercial-Off-The-Shelf, COTS components. The use of Commercial-Off-The-Shelf, software components is increasing in today's development of new systems. Shorter system life cycles and decreased development budgets make it so. Using COTS Components can be one way of reducing development time and be competitive by getting products to the market fast and. Advantages that can be gained by developing a system using COTS components:

- Functionality is instantly accessible to the developer.
- Components may be less costly than those developed in-house.
- The component vendor may be an expert in the particular area of the component Functionality.

II. Other kinds of Components

Some examples of components:

- A JAVA class.
- A cluster of JAVA classes, with a particular class serving as the exported interface, and the other classes functioning as part of the implementation.
- A Windows DLL.
- A COM object.
- A Java Bean.
- A CORBA-based server object.
- A UNIX shell program (i.e., functioning within a pipe-and-filter style architecture).

III. TESTING TECHNIQUES

Some of the techniques are:-

- Adequate testing
- Integrated testing technique.
- Automated software robustness testing.
- Boundary value analysis.
- Self testing of component based software.
- Object oriented component testing.
- Event flow model.
- Regression testing.
- Modular Regression Testing

IV. CONCLUSION

ISBN: 978-93-83758-09-8

When a software system is developed then to ensure the quality, reliability, robustness and functionality of system testing is necessary. In this paper we study about component based software engineering and some testing techniques. Component based software engineering is still at its young stage of life and there is much area for research in this field.
From our research we explore some testing techniques. In our future work we will test the components with best one testing tool.

V. REFERENCES

[1] Sami Beydeda and Volker Gruhn, "An Integrated testing Technique for Component-Based Software", IEEE Computer Society, 2001.
[2] Philip T Cox and Baoming Song, "A formal Model for Component-Based Software", IEEE Computer Society, Document number 07695-474-4/01, pp.304-310. 2001
[3] Fevzi Belli and Christof J. Budnik, "Towards Self-Testing of Component-Based Software", Proceeding of the 29th Annual International Software and Applications Conference (COMPSAC '05), IEEE, 2005.
[4] Sami Beydeda and Volker Gruhn, "An Integrated testing Technique for Component-Based Software", IEEE Computer Society, 2001
[5] S Phani Shashank et.al "A Systematic Literature Survey of Integration Testing in Component-Based Software Engineering", IEEE 2010.
[6] Alan W.Brown, Kurt C.Walinau, "The Current State of CBSE", In Proceedings of IEEE Software, Document number 0740-7459/98, pp. 37-46. September/ October 1998
[7] Alejandra Cechich and Mario Piattini-Velthuis, "Component-Based Software Enginnering", proceedings of The European Journal for the Informatics Professional UPGRADE August 2003 Vol. IV , No 4, pp.15-19.
[8] David S. Rosenblum, "Adequate Testing of Component-Based Software", Technical Report 97-34 Department of Information and Computer Science University of California, Irvine, CA 92697-3425, 11 Aug 1997.