International Journal of Research

International Journal of Research (IJR)
IJR e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 05, May 2015
Available at http://internationaljournalofresearch.org

Review paper on Software Testing

Vishakha & Rishabh Sharma
Dronacharay college of engineering , Gurgaon
Vishakha1360@gmail.com, rishabhsharma.244@gmail.com

1 Abstract

Software testing applied for concurrent programs is a challenging activity. Considering the relevance of concurrent programs testing, several research have been conducted in this area, especially involving adaptation of the techniques and criteria applied in sequential programs. In this paper different testing levels and testing methods have been discussed in order to understand the software testing.

2 Introduction

Software testing is performed to verify that the completed software package functions according to the expectations defined by the requirements/specifications. It is the process where a program is executed in order to find any error.According to ANSI/IEEE 1059 standard [1, 2], Testing can be defined as —A process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item. Another more appropriate definition is this: [3] Testing is the process of executing a program with the intent of finding errors. The concept of testing is as old as coding and is change along with time. Gelperin and Hetzel [4] proposed the concept of the testing process model based on associated publishing event.

3 Testing methods

3.1Static vs. dynamic testing

There are many approaches available in software testing. Reviews, walkthroughs, or inspections are referred to as static testing, whereas actually executing programmed code with a given set of test cases is referred to as dynamic testing. Static testing is often implicit, as proofreading, plus when programming tools/text editors check source code structure or compilers (pre-compilers) check syntax and data flow as static program analysis. Dynamic testing takes place when the program itself is run. Dynamic testing may begin before the program is 100% complete in order to test particular sections of code and are applied to discrete functions or modules. Typical techniques for this are either using stubs/drivers or execution from a debugger environment.

Static testing involves verification, whereas dynamic testing involves validation. Together they help improve software quality. Among the techniques for static analysis, mutation testing can be used to ensure the test-cases will detect errors which are introduced by mutating the source code.

3.2The box approach

Software testing methods are traditionally divided into white- and black-box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

3.2.1White-Box testing

White Box Testing is the testing of a software solution's internal coding and infrastructure.It focuses primarily on strengthening security, the flow of inputs and outputs through the application, and improving design and usability.White box testing is also known as **clear, open, structural,**

and glass box testing.Whitebox testing is based on the inner workings of an application and revolves around internal testing. The term "whitebox" was used because of the see-through box concept. The clear box or whitebox name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings.

3.2.2Black box testing

Black box testing is on functionality of the system as a **whole.** The term **'behavioral testing'** is also used for black box testing and white box testing is also sometimes called **'structural testing'**. Behavioral test design is slightly different from black-box test design because the use of internal knowledge isn't strictly forbidden, but it's still discouraged.Blackbox testing, involves testing from an external or end-user type perspective.

3.2.3Gray box testing

Gray box testing, also called gray box analysis, is a strategy for software debugging in which the tester has limited knowledge of the internal details of the program. A gray box is a device, program or system whose workings are partially understood. Gray box testing is considered to be non-intrusive and unbiased because it does not require that the tester have access to the source code. With respect to internal processes, gray box testing treats a program as a black box that must be analyzed from the outside. During a gray box test, the person may know how the system components interact but not have detailed knowledge about internal program functions and operation. A clear distinction exists between the developer and the tester, thereby minimizing the risk of personnel conflicts.

4 Testing levels

Testing levels are basically to identify missing areas and prevent overlap and repetition between the development life cycle phases. In software development life cycle models there are defined phases like requirement gathering and analysis, design, coding or implementation, testing and deployment. Each phase goes through the testing. Hence there are various levels of testing. The various levels of testing are:

4.1 Unit testing

It is basically done by the developers to make sure that their code is working fine and meet the user specifications. They test their piece of code which they have written like classes, functions, interfaces and procedures.

4.2 Component testing

It is also called as module testing. The basic difference between the unit testing and component testing is in unit testing the developers test their piece of code but in component testing the whole component is tested. For example, in a student record application there are two modules one which will save the records of the students and other module is to upload the results of the students. Both the modules are developed separately and when they are tested one by one then we call this as a component or module testing.

4.3 Integration testing

Integration testing is done when two modules are integrated, in order to test the behavior and functionality of both the modules after integration. Below are few types of integration testing-

- Big bang integration testing
- Top down
- Bottom up
- Functional incremental

4.4 Component integration testing:

In the example above when both the modules and components are integrated then the testing done is called as Component integration testing. This testing is basically done to ensure that the code should not break after integrating the two modules.

4.5 System integration testing

System integration testing (SIT) is a testing where testers basically test that in the same environment all the related systems should maintain data integrity and can operate in coordination with other systems.

4.6 System testing

In system testing the testers basically test the compatibility of the application with the system.

4.7 Acceptance testing

Acceptance testing are basically done to ensure that the requirements of the specification are met.

5 References

[1.] Almasi, G. S. and Gottlieb, A.: Highly Parallel Computing. The Benjamin Cummings Publishing Company, Redwood City, CA, USA (1994)

[2.] Bertolino, A.: Software testing research: Achievements, challenges, dreams. In InternationConference on Software Engineering, pages 85103. IEEE Computer Society(2007)

[3.] Bradbury, J. S. Using program mutation for the empirical assessment of fault detection techniques: a comparison of concurrency testing and model checking. PhD thesis, Kingston, Ont., Canada (2007)

[4.] Chen, J. and MacDonald, S.: Towards a better collaboration of static and dynamic analyses for testing concurrent programs. In: PADTAD '08: Proceedings of the 6th workshop on Parallel and distributed systems, pp. 1{9. ACM, New York, NY, USA(2008)