Study on Integration Testing and System Testing

D. Dhivya, Dr. K. Nirmala
1M.Phil Research Scholar, 2Associate Professor
1PG & Research Dept. of Computer Science,
1 Quaid-E-Millath Government College for Women (Autonomous), Anna Salai, Chennai 600002, Tamil Nadu, India

Abstract: Software testing is the procedure used to determine the accuracy, completeness, security and nature of a created software. Testing is the way of executing a program with the objective of finding mistakes. Integration testing focuses on two or more individual modules which might be grouped to form a partial system. Integration testing utilizes test information to assess the individual units, and their interfaces for consolidated conduct. System testing is trying out the entire process of the conduct, which incorporates the device to assess the system compliance with its particular necessities. This paper focuses the similar examination on Integration and System testing. The reason, approaches, points, desires and kinds of these testing are discussed in this paper.

Index Terms - Software Testing, Testing and Levels, Integration testing, System testing

I. INTRODUCTION

Software testing is something beyond mistake identification; it is the process which allows the software to work below controlled conditions, to verify that it behaves “as specified”, to find errors, and to validate whether it come across user specification. Software testing is done to find software defects or failures in advance [3]. Software testing is a technique aimed towards evaluating an attribute or capability of an application/product and determining that it meets its best. It is additionally used to test the product for other programming quality components like dependability, ease of use, security, capacity, proficiency, conveyance, similarity and so forth. It develops the integrity of the system with the aid off detecting deviations in design and error in the structures.

II. TESTING AND LEVELS

The distinctive levels of testing are utilized to approve the product at various levels of the improvement procedure.

2.1 Unit Testing
Unit testing is the small testable piece of a whole application. It is utilized to give a bit of code that must fulfill the prerequisites.

2.2 Integration Testing
In integration testing, the code is divided into individual segments and tested as a group. The main assignment of integration testing is to investigate the parameters which include functional requirements, performance requirements and reliability requirements that are positioned on major design object.

2.3 Function Testing
Functional testing can be referred as black-box testing. In functional testing, testing is done by providing validate input and thus outcomes are observed accordingly [2].

2.4 System Testing
It seeks to detect imperfections inside the product units that are incorporated together.

2.5 Acceptance Testing
It is otherwise called as operational acceptance testing or field acceptance testing since it keeps running by the predefined acceptance test techniques to direct the client about which information is to be utilized after one another.

2.6 Regression Testing
In regression testing, the applications are examined which were formerly developed and analyzes if the deviations took place when the modifications are made in existing or new programs.

III. INTEGRATION TESTING AN OVERVIEW
Integration is defined as the set of interactions among components [4]. Testing the interaction between the modules and communication with different frameworks remotely is called integration testing. Integration testing begins when two of the product
component are accessible and end when all segments interfaces have been tried. The last round of reconciliation including all segments is called Final Integration Testing (FIT), or system integration. Integration testing is both a type of testing and a phase of testing [4]. Since integration testing tests the connections among the modules, this testing—simply like white box, black box, and different kinds of testing— accompanies an arrangement of procedures and techniques.

![Fig 1: Integration Testing](image)

**3.1 Methodologies of Integration Testing**

There are a few approaches accessible, to in choose the order for integration testing. These are as per the following:

- Top-down integration
- Bottom-up integration
- Bi-directional integration
- System integration

**3.1.1 Top-down Integration**

Integration testing includes testing the highest segment interface with different parts in same request as you explore from top to bottom, till you cover every segment.

**3.1.2 Bottom-up Integration**

Bottom-up Integration is the polar opposite of top-down integration, wherein the components for a new product improvement turn out to be available in opposite order, beginning from the lowest.

**3.1.3 Bi-directional Integration**

Bi-directional integration is a mixture of the top-down and bottom-up integration strategies used collectively to derive integration steps. This approach is also known as “sandwich integration”.

**3.1.4 System Integration**

System integration is a way that every component of a system is incorporated and tested as a single unit. The salient point of this testing technique raise, is that of optimization. This approach is also known as “big-bang” integration.

<table>
<thead>
<tr>
<th>S. No</th>
<th>Factors</th>
<th>Suggested integration method</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Clear requirements and design</td>
<td>Top-down</td>
</tr>
<tr>
<td>2</td>
<td>Dynamically changing requirements, design, architecture</td>
<td>Bottom-up</td>
</tr>
<tr>
<td>3</td>
<td>Changing architecture, stable design</td>
<td>Bi-directional</td>
</tr>
<tr>
<td>4</td>
<td>Limited changes to existing architecture with less impact</td>
<td>Big bang</td>
</tr>
<tr>
<td>5</td>
<td>Combination of above</td>
<td>Select one of the above after careful analysis</td>
</tr>
</tbody>
</table>
IV. SYSTEM TESTING AN OVERVIEW

System testing is defined as a testing segment conducted at the whole integrated system, to evaluate the device compliance with its specific necessities. It’s far finished after unit, component, and integration testing. A system is an entire set of incorporated components that collectively deliver product functionality and features. In order to test the whole device, it’s far necessary to understand the product’s behaviour as a whole. System testing helps in revealing the imperfections that may not be specifically inferable from a module or an interface. System testing brings out issues that are central outline, design, and code of the entire product. System testing is the main period of testing which tests both functional and non-functional aspects of the product.

![Fig 2: System Testing](image)

4.1 Functional System Testing

Functional testing is performed at various stages and focus is on product level highlights. The common techniques of Functional System Testing are:

4.1.1 Design / Architecture verification

In this technique of functional testing, the experiments are created and checked against the outline and design to see whether they are actual *product-level test cases*.

4.1.2 Business Vertical Testing

Utilizing and testing the product for various business verticals, for example, banking, insurance, asset management, and so on, and verifying the business tasks and utilization, is known as business vertical testing.

4.1.3 Deployment Testing

Deployment testing that occurs in a product advancement organization to guarantee that client sending necessities are met. Deployment testing is additionally directed after the arrival of the product by using the resources and setup accessible in clients’ areas.

4.1.4 Beta Testing

One of the components utilized as a part of sending the product that is under test to the clients and getting the criticism. This is called beta testing.

4.1.5 Certification, Standards and Testing for Compliance

A product should be certified with the well known equipment, working framework, database, and other foundation pieces. This is called Certification testing. Testing the product to guarantee that principles are appropriately executed is called testing for standards. Testing the item for authoritative, legitimate, and statutory compliance is one of the basic exercises of the framework testing group.

4.2 Non-Functional System Testing

Non-Functional System Testing composes of testing types (also called quality factors), some of which are as per the following:

4.2.1 Performance / Load Testing

To evaluate the time taken or reaction time of the framework to perform its required capacities in correlation with various variants of the same product(s) or an alternate focused product(s) is called performance testing.

4.2.2 Scalability Testing

A testing that requires tremendous measure of resource to discover the greatest ability of the system parameters is called scalability testing.
4.2.3 Reliability Testing
To evaluate the capacity of the system or an independent part of the system to perform its required functions over and again for a predefined period of the time is called reliability testing.

4.2.4 Stress Testing
Assessing a framework beyond the limits of specified requirements or system resources (for example, disk space, processor utilization) to guarantee the framework does not break down unexpectedly is called stress testing.

4.2.5 Interoperability Testing
This testing is done to guarantee that at least two items can exchange data, utilize the data, and work nearly.

4.2.6 Localization Testing
Testing directed to check that the localized product works in different languages is called localization testing.

<table>
<thead>
<tr>
<th>TESTING ASPECTS</th>
<th>FUNCTIONAL TESTING</th>
<th>NON-FUNCTIONAL TESTING</th>
</tr>
</thead>
<tbody>
<tr>
<td>Involves</td>
<td>Product features and functionality</td>
<td>Quality factors</td>
</tr>
<tr>
<td>Tests</td>
<td>Product behaviour</td>
<td>Behaviour and experience</td>
</tr>
<tr>
<td>Result conclusion</td>
<td>Simple steps written to check expected results</td>
<td>Huge data collected and analysed</td>
</tr>
<tr>
<td>Result varies due to</td>
<td>Product implementation</td>
<td>Product implementation, resources, and configurations</td>
</tr>
<tr>
<td>Testing focus</td>
<td>Defect detection</td>
<td>Qualification of product</td>
</tr>
<tr>
<td>Knowledge required</td>
<td>Product and domain</td>
<td>Product, domain, design, architecture, statistical skills</td>
</tr>
<tr>
<td>Failures normally due to</td>
<td>Code</td>
<td>Architecture, design, and code</td>
</tr>
<tr>
<td>Testing phase</td>
<td>Unit, component, integration, system</td>
<td>System</td>
</tr>
<tr>
<td>Test case repeatability</td>
<td>Repeated many times</td>
<td>Repeated only in case of failures and for different configurations</td>
</tr>
<tr>
<td>Configuration</td>
<td>One-time setup for a set of test cases</td>
<td>Configuration changes for each test case</td>
</tr>
</tbody>
</table>

TABLE III - Comparative Analysis of Integration Testing and System Testing [1].

<table>
<thead>
<tr>
<th></th>
<th>Integration Testing</th>
<th>System Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technique</td>
<td>It is a systematic technique for constructing the program structure while at the same time conducting test to uncover errors associated with interfacing.</td>
<td>The system testing process is concerned with finding errors that result from unanticipated interaction between subsystem and system component.</td>
</tr>
<tr>
<td>Aim</td>
<td>It involves integrating independent software unites or components to form a sizeable build and then testing the assembly.</td>
<td>It involves integration the subsystem to make up the entire system.</td>
</tr>
<tr>
<td>Purpose</td>
<td>To prove that all areas of software units or components interface with each other and also to verify the functionality that there are no gaps in the dataflow.</td>
<td>Verifying end-to-end work flows and scenarios. All the software components, all the hardware components, all internal interfaces, all the external interfaces should be tested.</td>
</tr>
</tbody>
</table>
Environment | Integration testing takes place either in the development environment or in test environment using real data, if possible else simulated data need to be created to model real data. | System testing requires system test environment that comprise of deployment like environment from hardware and software requirements.

Expectations | The primary emphasis is verification of each component and inter-modular interfaces. | The primary emphasis is verification of the system as a whole.

It tries to test all testable requirements at least once by the end of testing | This serves as a final verification of requirements and design.

Hardware specification should be verified for correctness and compliance with specification. | Correct operation of external interface must be verified and some performance test may be conducted and used to model or extrapolate behaviour.

Considerations | Integrating independent software units or components to form a sizeable build and then testing the assembly | Perform functional test, Regression tests, Performance tests, Load test.

To find any issues in interface among units or components | Perform interface validation tests.

To find any gaps in the data flow | Perform security test.

Tasks | Integrate software units or components | Arrive at detailed system test plan.

Prepare integration test report | Perform system testing report.

Approaches | Top-down approach | Alpha testing.

Bottom up approach | Beta testing.

Sandwich integration | Acceptance testing.

Big bang

V. CONCLUSION

This paper describes about the software testing, their levels, integration and system testing. The motivation behind integration testing is to demonstrate that all regions of software units or parts interface with each other and furthermore to check the functionality that there are no gaps in the dataflow while in System testing fundamental point is discovering mistakes from unforeseen communication among subsystem and system component. Both these techniques for testing have their own role and significance in the lifecycle of software and their testing.

REFERENCES