INTEGRATION Testing Tutorial: Big Bang, Top Down & Bottom Up

What is Integration Testing?

In integration Testing, individual software modules are integrated logically and tested as a group.

A typical software project consists of multiple software modules, coded by different programmers.

integration Testing focuses on checking data communication amongst these modules.

Hence it is also termed as **'I & T'** (Integration and Testing), **'String Testing'** and sometimes 'Thread Testing'.

Why do Integration Testing?:

Although each software module is unit tested, defects still exist for various reasons like

- A Module in general is designed by an individual software developer whose understanding and programming logic may differ from other programmers. integration Testing becomes necessary to verify the software modules work in unity
- At the time of module development, there are wide chances of change in requirements by the clients. These new requirements may not be unit tested and hence system integration Testing becomes necessary.
- Interfaces of the software modules with the database could be erroneous
- External Hardware interfaces, if any, could be erroneous
- Inadequate exception handling could cause issues.

Please be patient. The Video will load in some time. If you still face issue viewing video click here (/faq.html#1)

Integration Test Case:

Integration Test Case (/test-case.html)differs from other test cases in the sense it **focuses mainly on the interfaces & flow of data/information between the modules**. Here priority is to be given for the **integrating links** rather than the unit functions which are already tested.

Sample Integration Test Cases for the following scenario:Application has 3 modules say 'Login Page', 'Mail box' and 'Delete mails' and each of them are integrated logically.

Here do not concentrate much on the Login Page testing as it's already been done in Unit Testing (/unit-testing-guide.html). But check how it's linked to the Mail Box Page.

Similarly Mail Box: Check its integration to the Delete Mails Module.

Test Case ID	Test Case Objective	Test Case Description	Expected Result
1	Check the interface link between the Login and Mailbox module	Enter login credentials and click on the Login button	To be directed to the Mail Box
2	Check the interface link between the Mailbox and Delete Mails Module	From Mail box select the an email and click delete button	Selected email should appear in the Deleted/Trash folder

Approaches/Methodologies/Strategies of Integration Testing:

The Software Industry uses variety of strategies to execute Integration testing , viz.

- Big Bang Approach :
- Incremental Approach: which is further divided into following
  - Top Down Approach
  - Bottom Up Approach
  - Sandwich Approach - Combination of Top Down and Bottom Up

Below are the different strategies, the way they are executed and their limitations as well advantages.

Big Bang Approach:

Here all component are integrated together at **once**, and then tested.

Advantages:

- Convenient for small systems.

Disadvantages:

- Fault Localization is difficult.
- Given the sheer number of interfaces that need to be tested in this approach, some interfaces links to be tested could be missed easily.
- Since the integration testing can commence only after "all" the modules are designed, testing team will have less time for execution in the testing phase.
- Since all modules are tested at once, high risk critical modules are not isolated and tested on priority. Peripheral modules which deal with user interfaces are also not isolated and tested on priority.

Incremental Approach:

In this approach, testing is done by joining two or more modules that are ***logically related***. Then the other related modules are added and tested for the proper functioning. Process continues until all of the modules are joined and tested successfully.

This process is carried out by using dummy programs called **Stubs and Drivers**. Stubs and Drivers do not implement the entire programming logic of the software module but just simulate data communication with the calling module.

Stub: Is called by the Module under Test.

Driver: Calls the Module to be tested.

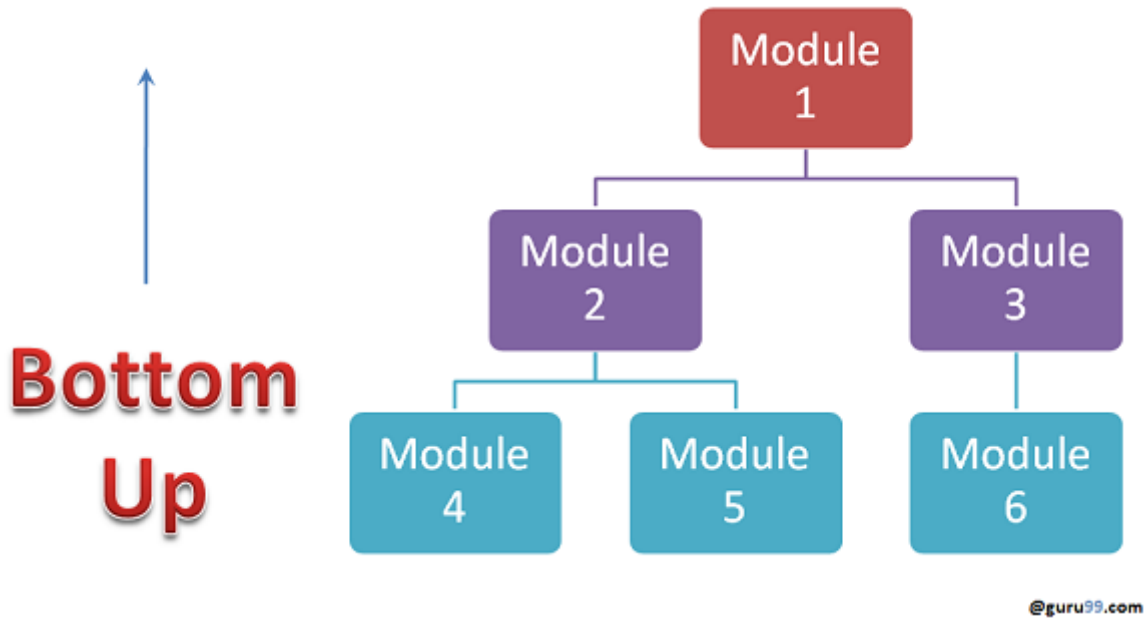
Incremental Approach in turn is carried out by two different Methods:

- **Bottom Up**
- **Top Down**

Bottom up Integration

In the bottom up strategy, each module at lower levels is tested with higher modules until all modules are tested. It takes help of Drivers for testing

Diagrammatic Representation:



(//cdn.guru99.com/images/bottom-up-integration-testing.png) **Advantages:**

- Fault localization is easier.
- No time is wasted waiting for all modules to be developed unlike Big-bang approach

Disadvantages:

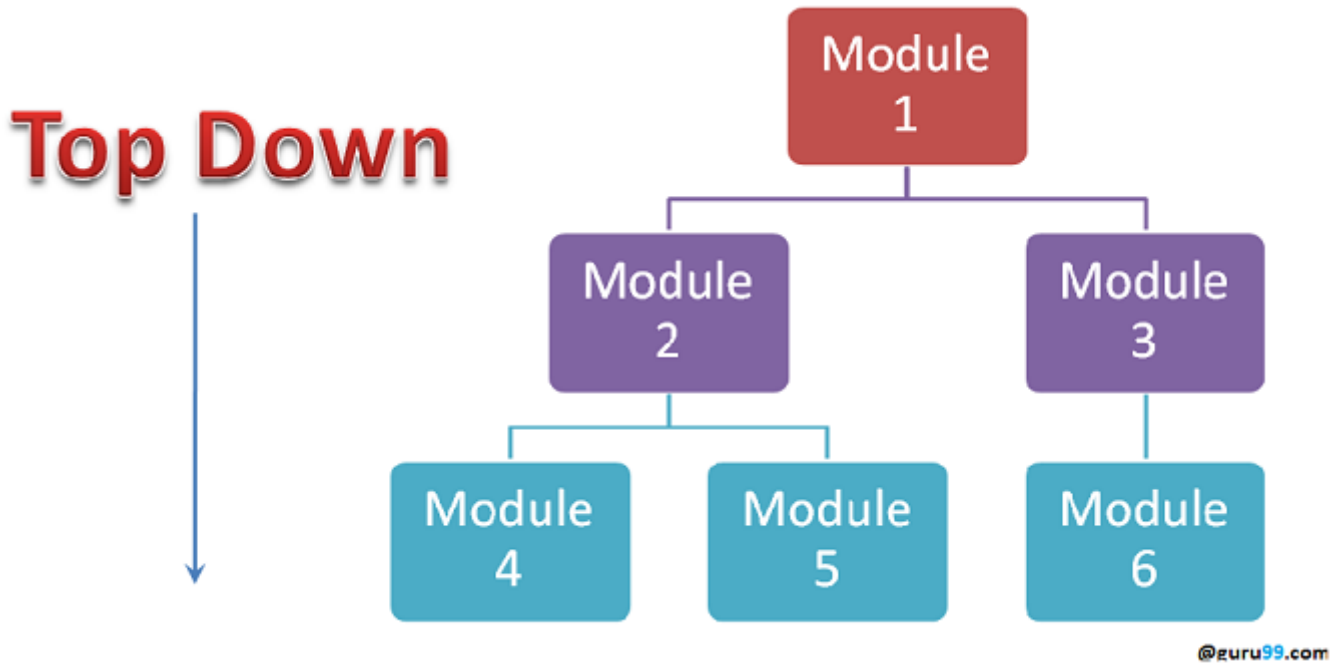
- Critical modules (at the top level of software architecture) which control the flow of application are tested last and may be prone to defects.
- Early prototype is not possible

Top down Integration:

In Top to down approach, testing takes place from top to down following the control flow of the software system.

Takes help of stubs for testing.

Diagrammatic Representation:



(//cdn.guru99.com/images/top-down-integration-testing.png) **Advantages:**

- Fault Localization is easier.
- Possibility to obtain an early prototype.
- Critical Modules are tested on priority; major design flaws could be found and fixed first.

Disadvantages:

- Needs many Stubs.
- Modules at lower level are tested inadequately.

Integration Testing Procedure

The integration test procedure irrespective of the test strategies (discussed above):

1. Prepare the Integration Tests Plan
2. Design the Test Scenarios, Cases, and Scripts.
3. Executing the test Cases followed by reporting the defects.
4. Tracking & re-testing the defects.
5. Steps 3 and 4 are repeated until the completion of Integration is successfully.

Brief Description of Integration Test Plans:

It includes following attributes:

- Methods/Approaches to test (as discussed above).
- Scopes and Out of Scopes Items of Integration Testing.
- Roles and Responsibilities.
- Pre-requisites for Integration testing.
- Testing environment.
- Risk and Mitigation Plans.

Entry and Exit Criteria.

Entry and Exit Criteria to Integration testing phase in any software development model

Entry Criteria:

- Unit Tested Components/Modules
- All High prioritized bugs fixed and closed
- All Modules to be code completed and integrated successfully.
- Integration tests Plan, test case, scenarios to be signed off and documented.
- Required Test Environment (/test-environment-software-testing.html)to be set up for Integration testing

Exit Criteria:

- Successful Testing of Integrated Application.
- Executed Test Cases are documented
- All High prioritized bugs fixed and closed
- Technical documents to be submitted followed by release Notes.

Best Practices/ Guidelines for Integration Testing

- First determine the Integration Test Strategy (/how-to-create-test-strategy-document.html)that could be adopted and later prepare the test cases and test data accordingly.

- Study the Architecture design of the Application and identify the Critical Modules. These need to be tested on priority.
- Obtain the interface designs from the Architectural team and create test cases to verify all of the interfaces in detail. Interface to database/external hardware/software application must be tested in detail.
- After the test cases, it's the test data which plays the critical role.
- Always have the mock data prepared, prior to executing. Do not select test data while executing the test cases.

Prev (/unit-testing-guide.html) **Report a Bug** Next (/system-testing.html)

YOU MIGHT LIKE:

SOFTWARE TESTING

(/defect-severity-in-software-testing.html)

(/defect-severity-in-software-testing.html)

Severity & Priority in Testing: Introduction & Differences

(/defect-severity-in-software-testing.html)

SOFTWARE TESTING

(/storage-testing.html)

(/storage-testing.html)

Storage Testing in Software Testing

(/storage-testing.html)

SOFTWARE TESTING

(/testing-telecom-application-with-sample-testcases.html)

(/testing-telecom-application-with-sample-testcases.html)

Testing Telecom Domain Application with Sample Testcases

(/testing-telecom-application-with-sample-testcases.html)

SOFTWARE TESTING

(/download-sample-test-case-template-with-explanation-of-important-fields.html)

(/download-sample-test-case-template-with-explanation-of-important-fields.html)

Download Sample Test Case Template with Explanation of Important Fields

SOFTWARE TESTING

(/protocol-testing.html)

(/protocol-testing.html)

Protocol Testing

(/protocol-testing.html)

SOFTWARE TESTING

(/destructive-testing.html)

(/destructive-testing.html)

Destructive Testing Tutorial: Methods & Strategy

(/destructive-testing.html)

(/download-sample-test-case-template-with-explanation-of-important-fields.html)

Testing Tutorials

About

About US (/about-us.html)
Advertise with Us (/advertise-us.html)
Write For Us (/become-an-instructor.html)
Contact US (/contact-us.html)

Career Suggestion

SAP Career Suggestion Tool (/best-sap-module.html)
Software Testing as a Career (/software-testing-career-complete-guide.html)
Top Tools List (/testing-development-tools.html)
Certificates (/certificate-it-professional.html)

Interesting

Books to Read! (/books.html)
Suggest a Tutorial
Blog (/blog/)
Quiz (/tests.html)

Execute online

Execute Java Online (/try-java-editor.html)
Execute Javascript (/execute-javascript-online.html)
Execute HTML (/execute-html-online.html)
Execute Python (/execute-python-online.html)