**What is Integration Testing?**

**INTEGRATION TESTING** is defined as a type of testing where software modules are integrated logically and tested as a group. A typical software project consists of multiple software modules, coded by different programmers. The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated

**Example of Integration Test Case**

Integration[Test Case](https://www.guru99.com/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’, ‘Mailbox’ and ‘Delete emails’ and each of them is integrated logically.

Here do not concentrate much on the Login Page testing as it’s already been done in [Unit Testing](https://www.guru99.com/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 Mailbox select the email and click a delete button | Selected email should appear in the Deleted/Trash folder |

## Approaches, Strategies, Methodologies of Integration Testing

**Big Bang Testing**

**Big Bang Testing** is an Integration testing approach in which all the components or modules are integrated together at once and then tested as a unit. This combined set of components is considered as an entity while testing. If all of the components in the unit are not completed, the integration process will not execute.

**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 link to be tested could be missed easily.
* Since the Integration testing can commence only after “all” the modules are designed, the 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 Testing**

In the **Incremental Testing** approach, testing is done by integrating two or more modules that are logically related to each other and then tested for proper functioning of the application. Then the other related modules are integrated incrementally and the process continues until all the logically related modules are integrated and tested successfully.

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

* Bottom Up
* Top Down

**Stubs and Drivers**

**Stubs and Drivers** are the dummy programs in Integration testing used to facilitate the software testing activity. These programs act as a substitutes for the missing models in the testing. They do not implement the entire programming logic of the software module but they simulate data communication with the calling module while testing.

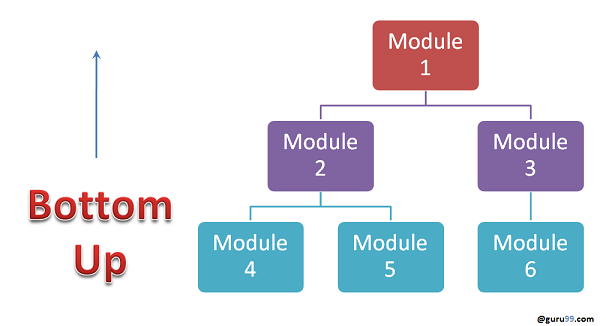
**Stub**: Is called by the Module under Test.

**Driver**: Calls the Module to be tested.

**Bottom-up Integration Testing**

**Bottom-up Integration Testing** is a strategy in which the lower level modules are tested first. These tested modules are then further used to facilitate the testing of higher level modules. The process continues until all modules at top level are tested. Once the lower level modules are tested and integrated, then the next level of modules are formed.

**Diagrammatic Representation**:



**Advantages:**

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

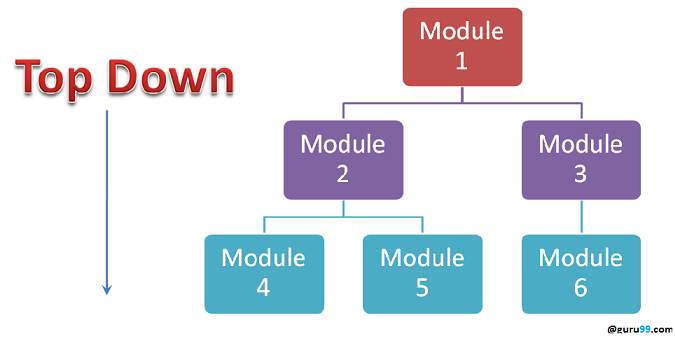
**Disadvantages:**

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

**Top-down Integration Testing**

**Top Down Integration Testing** is a method in which integration testing takes place from top to bottom following the control flow of software system. The higher level modules are tested first and then lower level modules are tested and integrated in order to check the software functionality. Stubs are used for testing if some modules are not ready.

**Diagrammatic Representation:**



**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 a lower level are tested inadequately.