****What is Integration Testing: Learn with Integration Testing Examples****

Integration testing is done to test the modules/components when integrated to verify that they work as expected i.e. to test the modules which are working fine individually does not have issues when integrated.

When talking in terms of testing large application using black box testing technique, involves the combination of many modules which are tightly coupled with each other. We can apply the Integration testing technique concepts for testing these types of scenarios.

### What is integration testing?

The meaning of Integration testing is quite straightforward- *Integrate/combine the unit tested module one by one and test the behavior as a combined unit.*

The main function or goal of this testing is to test the interfaces between the units/modules.

We normally do Integration testing after “Unit testing”. Once all the individual units are created and tested, we start combining those “Unit Tested” modules and start doing the integrated testing.

The main function or goal of this testing is to test the interfaces between the units/modules.

The individual modules are first tested in isolation. Once the modules are unit tested, they are integrated one by one, till all the modules are integrated, to check the combinational behavior, and validate whether the requirements are implemented correctly or not.

Here we should understand that Integration testing does not happen at the end of the cycle, rather it is conducted simultaneously with the development. So in most of the times, all the modules are not actually available to test and here is what the challenge comes to test something which does not exist!

### ****Why Integration Test?****

We feel that Integration testing is complex and requires some development and logical skill. That’s true! Then what is the purpose of integrating this testing into our testing strategy?

****Here are some reasons:****

1. In the real world, when applications are developed, it is broken down into smaller modules and individual developers are assigned 1 module. The logic implemented by one developer is quite different than another developer, so it becomes important to check whether the logic implemented by a developer is as per the expectations and rendering the correct value in accordance with the prescribed standards.
2. Many a time the face or the structure of data changes when it travels from one module to another. Some values are appended or removed, which causes issues in the later modules.
3. Modules also interact with some third party tools or APIs which also need to be tested that the data accepted by that API / tool is correct and that the response generated is also as expected.
4. A very common problem in testing – Frequent requirement change! :) Many a time developer deploys the changes without unit testing it. Integration testing becomes important at that time.

### ****Advantages****

****There are several advantages of this testing and few of them are listed below.****

* This testing makes sure that the integrated modules/components work properly.
* Integration testing can be started once the modules to be tested are available. It does not require the other module to be completed for testing to be done, as Stubs and Drivers can be used for the same.
* It detects the errors related to the interface.

### Challenges

****Listed below are few challenges that are involved in Integration Test.****

****#1)**** Integration testing means testing two or more integrated systems in order to ensure that the system works properly. Not only the integration links should be tested but an exhaustive testing considering the environment should be done to ensure that the integrated system works properly.

There might be different paths and permutations which can be applied to test the integrated system.

****#2)**** Managing Integration testing becomes complex because of few factors involved in it like the database, Platform, environment etc.

****#3)**** While integrating any new system with the legacy system, it requires a lot of changes and testing efforts. Same applies while integrating any two legacy systems.

****#4)**** Integrating two different systems developed by two different companies is a big challenge as for how one of the systems will impact the other system if any changes are done in any one of the systems is not sure.

In order to minimize the impact while developing a system, few things should be taken into consideration like possible integration with other systems, etc.

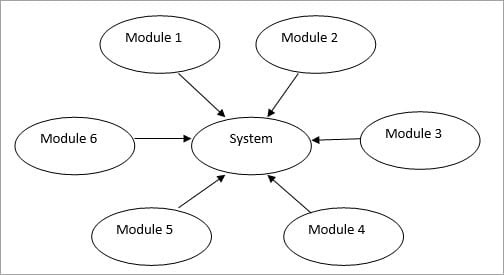
### Types of Integration Testing

****Given below is a type of Test Integration along with its advantages and disadvantages.****

#### Big Bang Approach:

Big bang approach integrates all the modules in one go i.e. it does not go for integrating the modules one by one. It verifies if the system works as expected or not once integrated. If any issue is detected in the completely integrated module, then it becomes difficult to find out which module has caused the issue.

Big bang approach is a time-consuming process of finding a module which has a defect itself as that would take time and once the defect is detected, fixing the same would cost high as the defect is detected at the later stage.

[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2018/05/Big-bang-approach.jpg)

****Advantages of Big Bang approach:****

* It is a good approach for small systems.

****Disadvantages of Big Bang Approach:****

* It is difficult to detect the module which is causing an issue.
* Big Bang approach requires all the modules all together for testing, which in turn, leads to less time for testing as designing, development, Integration would take most of the time.
* Testing takes place at once only which thereby leaves no time for critical module testing in isolation.

****Integration Testing Steps:****

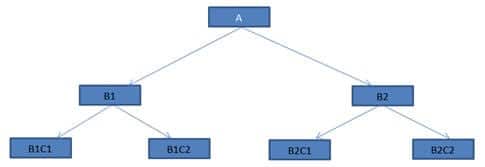
1. Prepare Integration [Test Plan.](https://www.softwaretestinghelp.com/how-to-write-test-plan-document-software-testing-training-day3/)
2. Prepare integration test scenarios & test cases.
3. Prepare test automation scripts.
4. Execute test cases.
5. Report the defects.
6. Track and re-test the defects.
7. Re-testing & testing goes on until integration testing is complete.

### ****Test Integration Approaches****

There are fundamentally 2 approaches for doing test integration:

1. ****Bottom-up approach****
2. ****Top-down approach.****

Let’s consider the below figure to test the approaches:

[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/12/Integration-1.jpg)

#### Bottom-up approach:

Bottom-up testing, as the name suggests starts from the lowest or the innermost unit of the application, and gradually moves up. The Integration testing starts from the lowest module and gradually progresses towards the upper modules of the application. This integration continues till all the modules are integrated and the entire application is tested as a single unit.

In this case, modules B1C1, B1C2 & B2C1, B2C2 are the lowest module which is unit tested. Module B1 & B2 are not yet developed. The functionality of Module B1 and B2 is that it calls the modules B1C1, B1C2 & B2C1, B2C2. Since B1 and B2 are not yet developed, we would need some program or a “stimulator” which will call the B1C1, B1C2 & B2C1, B2C2 modules. These stimulator programs are called ****DRIVERS****.

In simple words, ****DRIVERS**** are the dummy programs which are used to call the functions of the lowest module in a case when the calling function does not exist. The bottom-up technique requires module driver to feed test case input to the interface of the module being tested.

The advantage of this approach is that, if a major fault exists at the lowest unit of the program, it is easier to detect it, and corrective measures can be taken.

The disadvantage is that the main program actually does not exist until the last module is integrated and tested. As a result, the higher level design flaws will be detected only at the end.

#### Top-down approach

This technique starts from the topmost module and gradually progress towards the lower modules. Only the top module is unit tested in isolation. After this, the lower modules are integrated one by one. The process is repeated until all the modules are integrated and tested.

In the context of our figure, testing starts from Module A, and lower modules B1 and B2 are integrated one by one. Now here the lower modules B1 and B2 are not actually available for integration. So in order to test the topmost modules A, we develop “****STUBS****”.

“Stubs” can be referred to as code a snippet which accepts the inputs/requests from the top module and returns the results/ response. This way, in spite of the lower modules, do not exist, we are able to test the top module.

In practical scenarios, the behavior of stubs is not that simple as it seems. In this era of complex modules and architecture, the called module, most of the time involves complex business logic like connecting to a database. As a result, creating Stubs becomes as complex and time taking as the real module. In some cases, Stub module may turn out to be bigger than the stimulated module.

Both Stubs and drivers are dummy piece of code which is used for testing the “non- existing” modules. They trigger the functions/method and return the response, which is compared to the expected behavior

****Let’s conclude some difference between**[Stubs and Driver](https://w3.cs.jmu.edu/bernstdh/web/common/help/stubs-and-drivers.php" \t "https://www.softwaretestinghelp.com/what-is-integration-testing/_blank)**:****

| **Stubs** | **Driver** |
| --- | --- |
| Used in Top down approach | Used in Bottom up approach |
| Top most module is tested first | Lowest modules are tested first. |
| Stimulates the lower level of components | Stimulates the higher level of components |
| Dummy program of lower level components | Dummy program for Higher level component |

The only change is Constant in this world, so we have another approach called “****Sandwich testing****” which combines the features of both Top-down and bottom-up approach. When we test huge programs like Operating systems, we have to have some more techniques which are efficient and boosts more confidence. Sandwich testing plays a very important role here, where both, the Top down and bottom up testing are started simultaneously.

Integration starts with the middle layer and moves simultaneously towards up and down. In case of our figure, our testing will start from B1 and B2, where one arm will test the upper module A and another arm will test the lower modules B1C1, B1C2 & B2C1, B2C2.

Since both the approach starts simultaneously, this technique is a bit complex and requires more people along with specific skill sets and thus adds to the cost.

### ****GUI application Integration Test****

Now let’s talk about how we can imply integration testing in Black box technique.

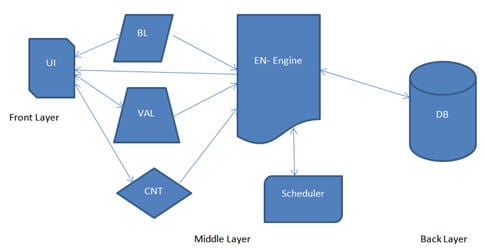
We all understand that a web application is a multitier application. We have a front end which is visible to the user, we have a middle layer which has business logic, we have some more middle layer which does some validations, integrate some third party APIs etc., then we have the back layer which is the database.

#### Integration testing example:

****Let’s check the below example:****

I am the owner of an advertising company and I post ads on different websites. At the end of the month, I want to see how many people saw my ads and how many people clicked on my ads. I need a report for my ads displayed and I charge accordingly to my clients.

*GenNext software* developed this product for me and below was the architecture:

[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/12/Integration-2.jpg)

****UI**** – User Interface module, which is visible to the end user, where all the inputs are given.  
****BL**** – Is the Business Logic module, which has all the all the calculations and business specific methods.  
****VAL**** – Is the Validation module, which has all the validations of the correctness of the input.  
****CNT**** – Is the content module which has all the static contents, specific to the inputs entered by the user. These contents are displayed in the reports.  
****EN**** – Is the Engine module, this module reads all the data that comes from BL, VAL and CNT module and extracts the SQL query and triggers it to the database.  
****Scheduler**** – Is a module which schedules all the reports based on the user selection (monthly, quarterly, semiannually & annually)  
****DB**** – Is the Database.

Now, having seen the architecture of the entire web application, as a single unit, Integration testing, in this case, will focus on the flow of data between the modules.

### ****Entry/Exit Criteria for Integration Testing****

****Entry Criteria:****

* Integration test plan document is signed off and approved.
* Integration test cases have been prepared.
* Test data has been created.
* [Unit testing](https://www.softwaretestinghelp.com/unit-testing/) of developed modules/Components is complete.
* All the critical and high Priority defects are closed.
* The test environment is set up for integration.

****Exit Criteria:****

* All the integration test cases have been executed.
* No critical and Priority P1 & P2 defects are opened.
* Test Report has been prepared.

### ****Integration Test Cases****

Integration test cases focus mainly on the****interface between the modules, integrated links, data transfer**** between the modules as modules/components that are already unit tested i.e. the functionality and the other testing aspects have already been covered.

So, the main idea is to test if integrating two working modules works as expected when integrated.

****For Example Integration Test cases for Linkedin application will include:****

* Verifying the interface link between the login page and the home page i.e. when a user enters the credentials and logs it should be directed to the homepage.
* Verifying the interface link between the home page and the profile page i.e. profile page should open up.
* Verify the interface link between the network page and your connection pages i.e. clicking accept button on Invitations of the network page should show the accepted invitation in your connection page once clicked.
* Verify the interface link between the Notification pages and say congrats button i.e. clicking say congrats button should direct towards the new message window.

Many integration test cases can be written for this specific site. The above four points are just an example to understand what Integration test cases are included in testing.

### Is Integration a White box or Black box Technique?

Integration testing technique can be counted in both black boxes as well as [white box technique](https://www.softwaretestinghelp.com/white-box-testing-techniques-with-example/). [Black box technique](https://www.softwaretestinghelp.com/black-box-testing/) is where a tester need not have any internal knowledge of the system i.e. coding knowledge is not required whereas white box technique needs internal knowledge of the application.

Now while performing integration testing it could include testing the two integrated web services which will fetch the data from the database & provide the data as required which means it could be tested using white box testing technique whereas integrating a new feature in the website can be tested using the black box technique.

So, it’s not specific that integration testing is a black box or white box technique.

### Integration Testing Tools

There are several tools available for this testing.

****Given below is a list of tools:****

* Rational Integration Tester
* Protractor
* Steam
* TESSY