

Levels of Testing



⌘ Software products are tested at three levels:

☑ Unit testing

☑ Integration testing

☑ System testing

Unit testing

⌘ During unit testing, modules are tested in isolation:

☑ If all modules were to be tested together:

☒ it may not be easy to determine which module has the error.

Unit testing



- ⌘ Unit testing reduces debugging effort several folds.
- ☑ Programmers carry out unit testing immediately after they complete the coding of a module.

Stubs



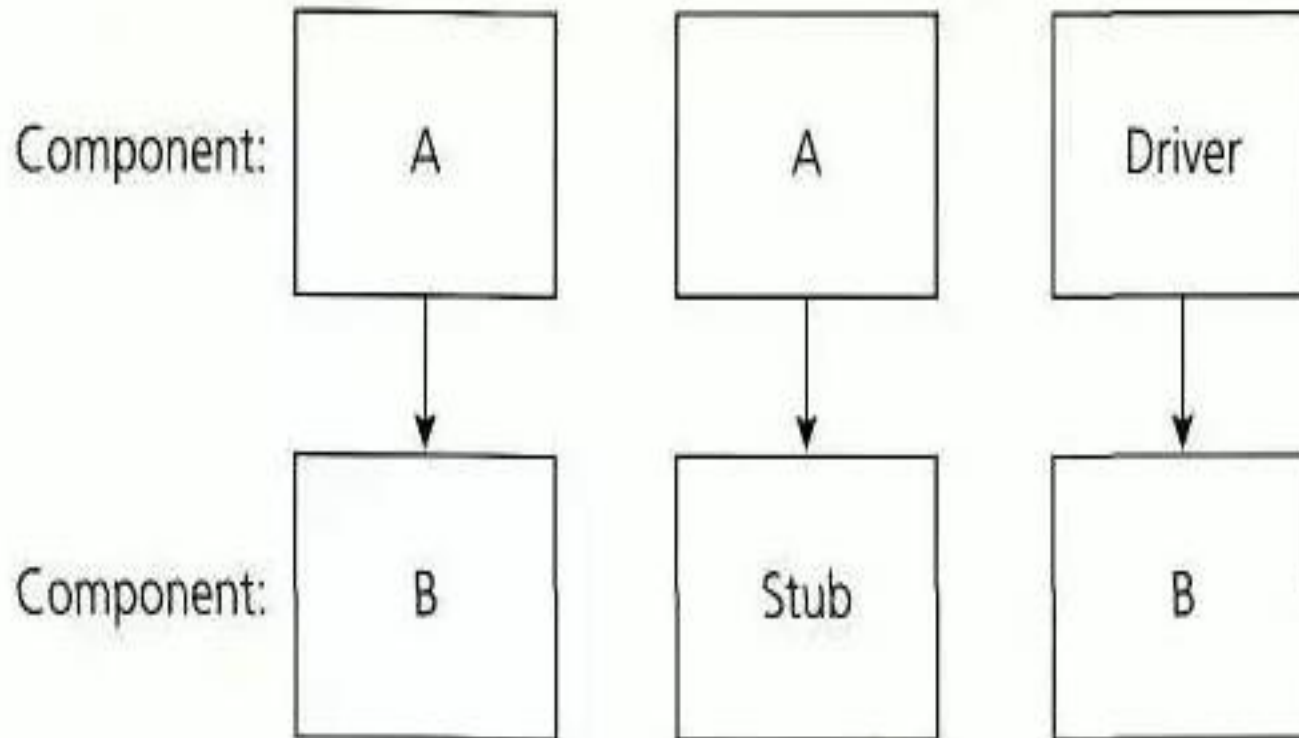
- ⌘ The upper modules are prepared first and are ready for testing. While the bottom modules are not yet prepared by the developers. So in order to form the complete application we create dummy programs for the lower modules in the application. Hence all the functionalities can be tested.
- ⌘ The main purpose of a stub is to allow testing of the upper levels of the code. Hence, when the lower levels of the code are not yet developed.

Driver

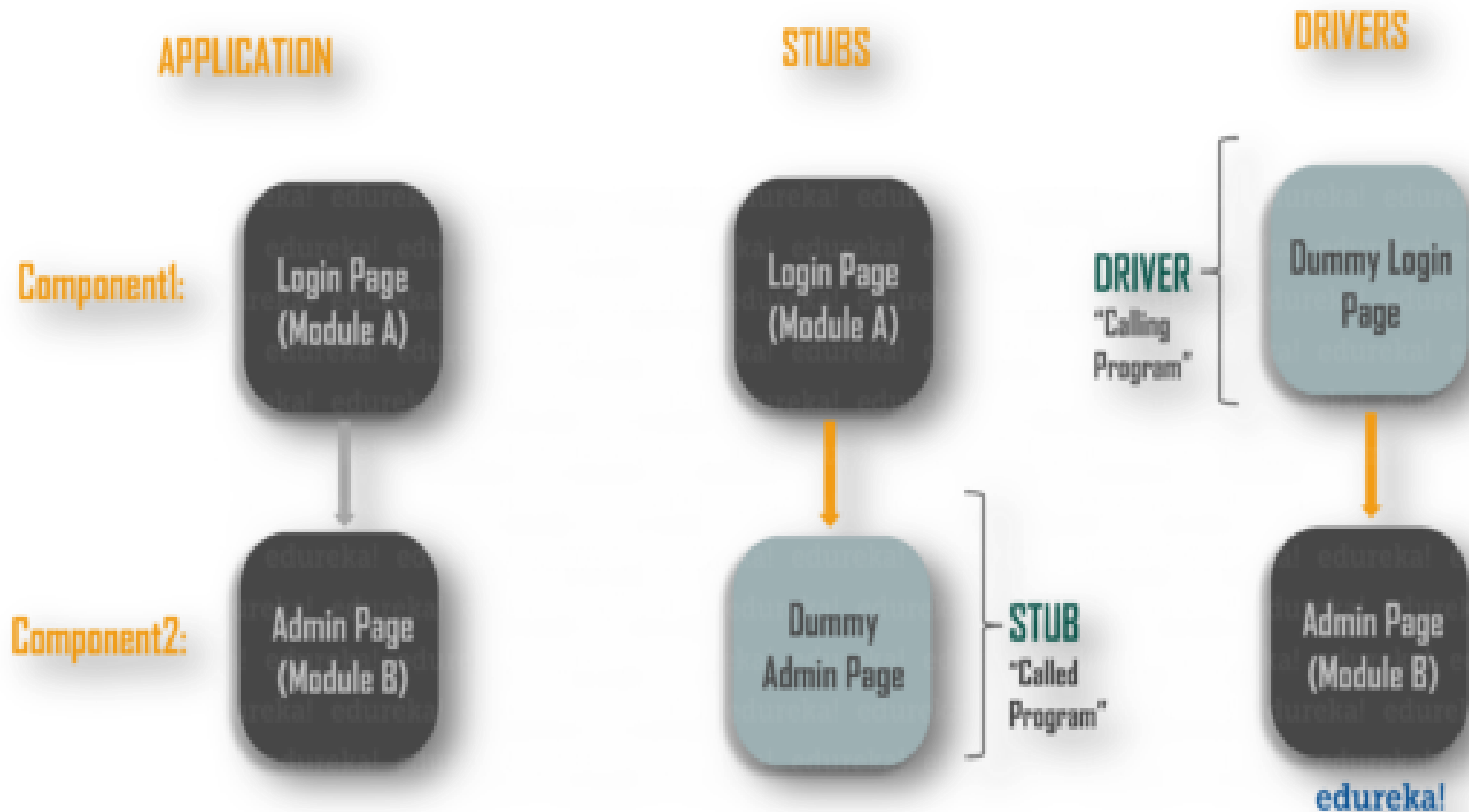


- ⌘ A driver is basically a piece of code through which other programs or pieces of code or modules can be called. Drivers are the main program through which other modules are called.
- ⌘ If we want to test any module it is required that we should have a main program which will call the testing module. Without the dummy program or driver, the complete testing of the module is not possible.
- ⌘ Testing of the bottom level modules is not possible with the help of main program. So we prepare a dummy program or driver to call the bottom level modules and perform its testing.

Stubs and Drivers Example



Stubs and Drivers Example



Integration testing



- ⌘ After different modules of a system have been coded and unit tested:
 - ☑ modules are integrated in steps according to an integration plan
 - ☑ partially integrated system is tested at each integration step.

Integration Testing



⌘ Develop the integration plan by examining the structure chart :

☑ big bang approach

☑ top-down approach

☑ bottom-up approach

☑ mixed approach

Big bang Integration Testing

⌘ Big bang approach is the simplest integration testing approach:

☐ all the modules are simply put together and tested.

☐ this technique is used only for very small systems.

Big bang Integration Testing

⌘ Main problems with this approach:

☐ if an error is found:

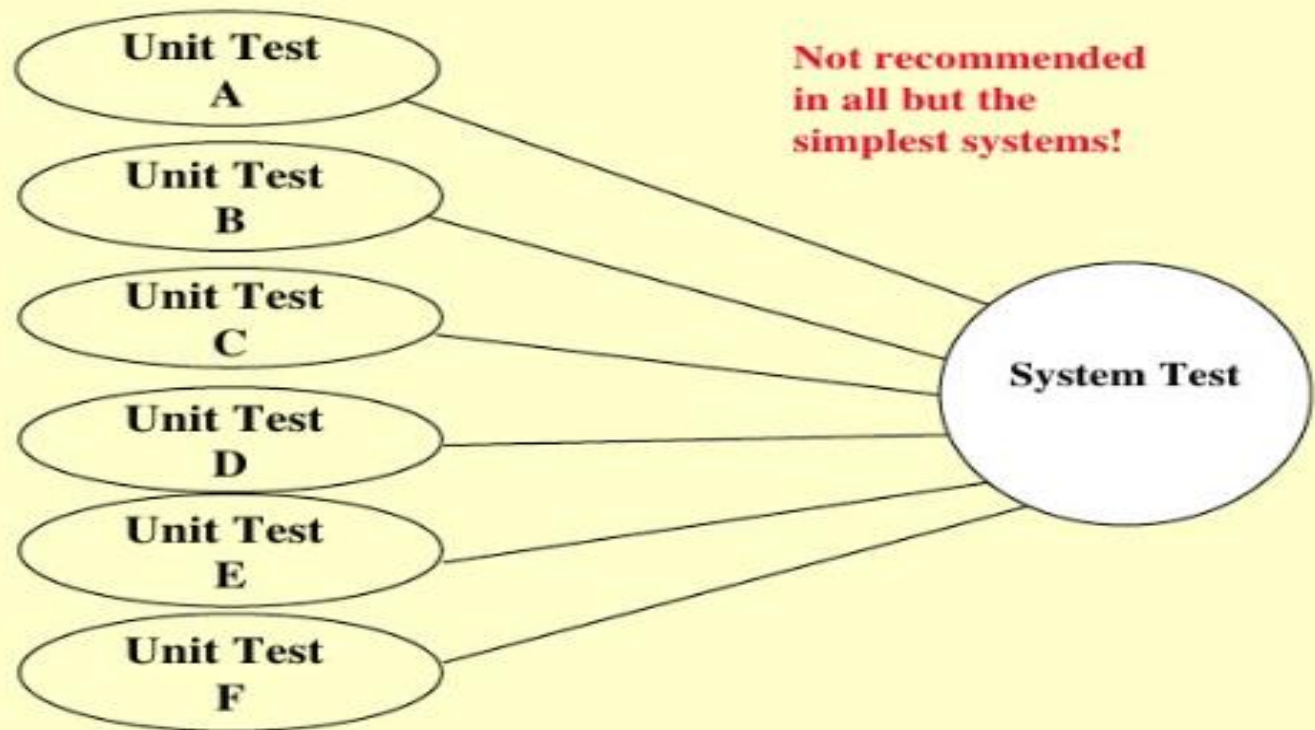
☒ it is very difficult to localize the error

☒ the error may potentially belong to any of the modules being integrated.

☐ debugging errors found during big bang integration testing are very expensive to fix.

Big bang Integration Testing

Integration Testing: *Big-Bang* Approach



Bottom-up Integration Testing

- ⌘ Integrate and test the bottom level modules first.
- ⌘ A disadvantage of bottom-up testing:
 - ☐ when the system is made up of a large number of small subsystems that are at the same level.
 - ☐ This extreme case corresponds to the big bang approach.

Bottom-up Integration Testing

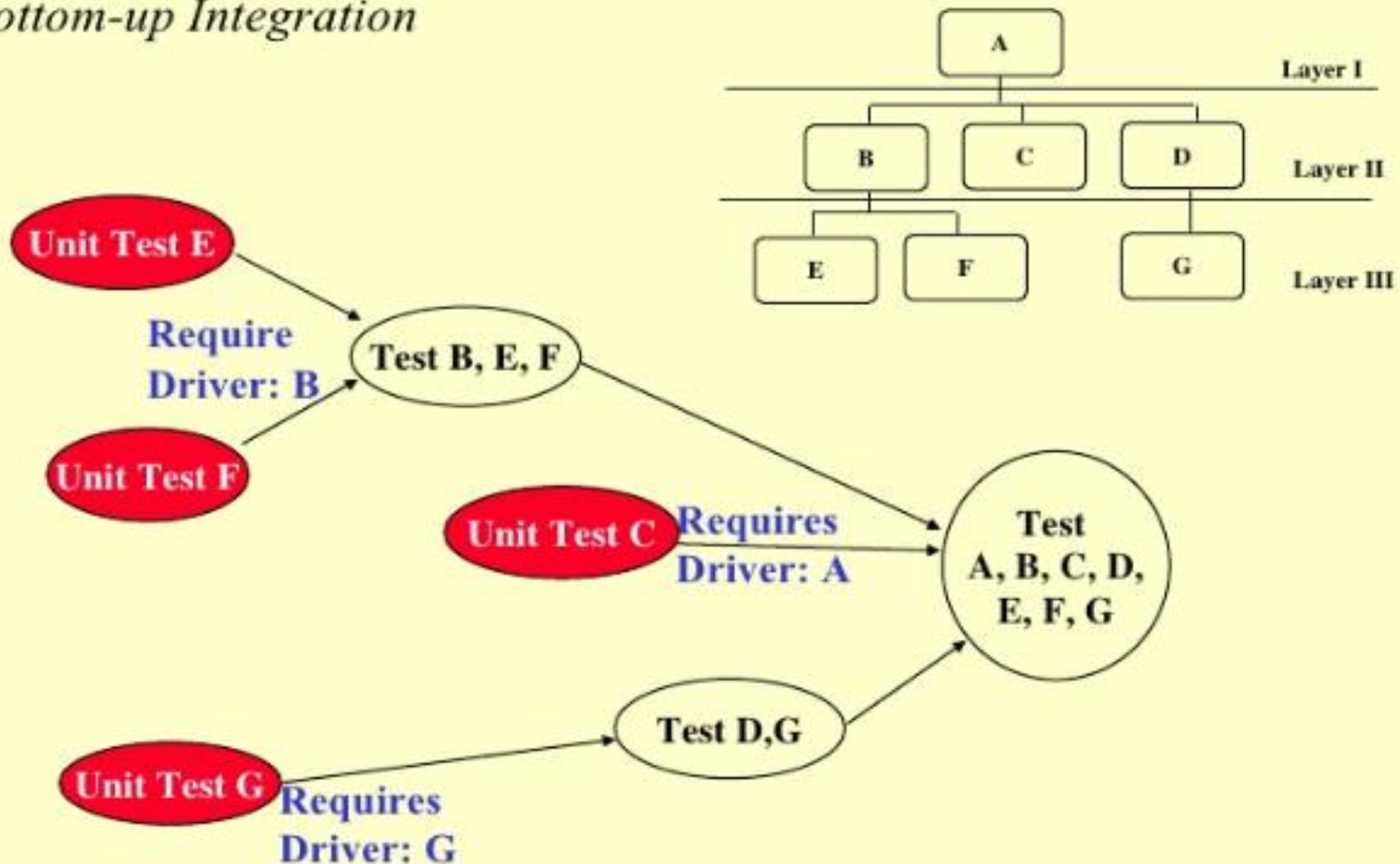


⌘ Advantages:

- ⌘ A principle advantage of this integration testing is that several disjoint subsystems can be tested simultaneously.
- ⌘ Low level modules get tested thoroughly which increases the reliability of the system.

Bottom-up Integration Testing

Bottom-up Integration



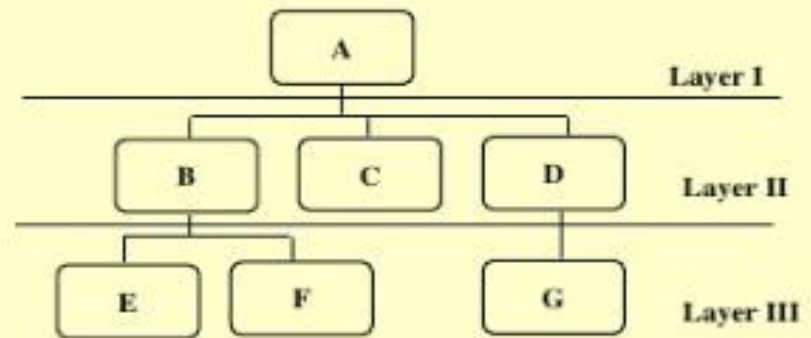
Top-down integration testing



- ⌘ Top-down integration testing starts with the main routine:
 - ☐ and one or two subordinate routines in the system.
- ⌘ After the top-level 'skeleton' has been tested:
 - ☐ immediate subordinate modules of the 'skeleton' are combined with it and tested.

Top-down integration testing

Top-down Integration Testing



Test A

Layer I

Test A, B, C, D

Layer I + II

Test
A, B, C, D,
E, F, G

All Layers

Requires
stubs:

B C D

E F G

Top-down integration testing



- ⌘ An advantage of top-down integration testing is that it requires writing only stubs, and stubs are simpler to write compared to drivers.
- ⌘ A disadvantage of the top-down integration testing approach is that in the absence of lower-level routines, it becomes difficult to exercise the top-level routines in the desired manner since the lower level routines usually perform input/output (I/O) operations.

Mixed integration testing



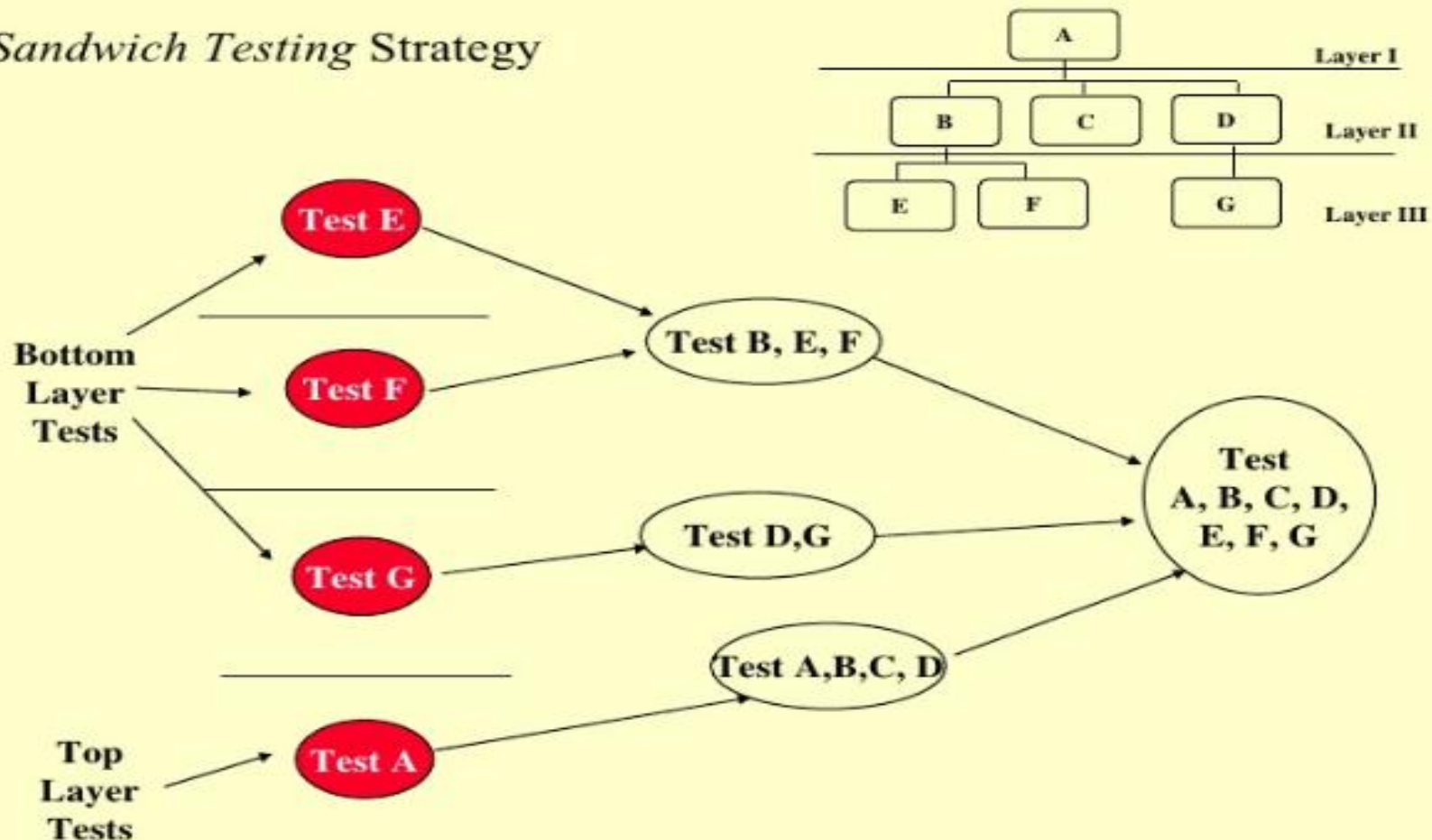
⌘ Mixed (or sandwiched) integration testing:

☑ uses both top-down and bottom-up testing approaches.

☑ Most common approach

Mixed integration testing

Sandwich Testing Strategy



Integration Testing


⌘ In top-down approach:

☒ testing waits till all top-level modules are coded and unit tested.

⌘ In bottom-up approach:

☒ testing can start only after bottom level modules are ready.

Phased versus Incremental Integration Testing



- ⌘ Integration can be incremental or phased.
- ⌘ In incremental integration testing,
 - ☑ only one new module is added to the partial system each time.

Phased versus Incremental Integration Testing



⌘ In phased integration,

⌘ a group of related modules are added to the partially integrated system each time.

⌘ Big-bang testing:

⌘ a degenerate case of the phased integration testing.

Phased versus Incremental Integration Testing

- ⌘ Phased integration requires less number of integration steps:
 - ☒ compared to the incremental integration approach.
- ⌘ However, when failures are detected,
 - ☒ it is easier to debug if using incremental testing
 - ☒ since errors are very likely to be in the newly integrated module.

System Testing



⌘ System testing involves:

☑ validating a fully developed system against its requirements.

System Testing



⌘ There are three main kinds of system testing:

- ☑ Alpha Testing

- ☑ Beta Testing

- ☑ Acceptance Testing

Alpha Testing



⌘ System testing is carried out by the test team within the developing organization.

Beta Testing



⌘ System testing performed by a select group of friendly customers.

Acceptance Testing



- ⌘ System testing performed by the customer himself:
 - ☑ to determine whether the system should be accepted or rejected.

System Testing



⌘ During system testing, in addition to functional tests:

📈 performance tests are performed.


Performance Testing



⌘ Addresses non-functional requirements.

- ☑ May sometimes involve testing hardware and software together.
- ☑ There are several categories of performance testing.

Stress testing



- ⌘ Evaluates system performance
 - ☑ when stressed for short periods of time.
- ⌘ Stress testing
 - ☑ also known as **endurance testing**.

Stress testing



- ⌘ Stress testing a Non-Functional testing technique that is performed as part of performance testing. During stress testing, the system is monitored after subjecting the system to overload to ensure that the system can sustain the stress.
- ⌘ The recovery of the system from such phase (after stress) is very critical as it is highly likely to happen in production environment.

Stress Testing



⌘ If the requirements is to handle a specified number of users, or devices:

☑ stress testing evaluates system performance when all users or devices are busy simultaneously.

Stress Testing



- ⌘ A most prominent use **of stress testing is to determine the limit, at which the system or software or hardware breaks.** It also checks whether the system demonstrates effective error management under extreme conditions.
- ⌘ The application under testing will be stressed when 5GB data is copied from the website and pasted in notepad. Notepad is under stress and gives 'Not Responded' error message.

Stress Testing (Real World Examples)



- ⌘ Commercial shopping apps or websites need to perform stress testing as the load becomes very high during festivals, sale or special offer period.
- ⌘ Web or emailing apps need to be stress tested.
- ⌘ Social networking websites or apps, blogs etc., need to be stress tested etc.

Volume Testing



- ⌘ **VOLUME TESTING** is a type of Software Testing, where the software is subjected to a huge volume of data. It is also referred to as **flood testing**. Volume testing is done to analyze the system performance by increasing the volume of data in the database.
- ⌘ Fields, records, and files are stressed to check if their size can accommodate all possible data volumes.
- ⌘ Test the site behavior when there are more than 10,000 laptops are available under the 'laptops' category.

Configuration Testing

- ⌘ Configuration testing is defined as a software testing type, that checks an application with multiple combinations of software and hardware to find out the optimal configurations that the system can work without any flaws or bugs.

Configuration Testing

⌘ Various Configurations:

- ⌘ **Operating System Configuration:** Win XP, Win 7 32/64 bit, Win 8 32/64 bit, Win 10 etc.
- ⌘ **Database Configuration:** Oracle, DB2, MySql, MSSQL Server etc.
- ⌘ **Browser Configuration:** IE 8, IE 9, Chrome, Microsoft Edge etc.

Recovery Testing



- ⌘ These tests check response to:
 - ☑ presence of faults or to the loss of data, power, devices, or services
 - ☑ subject system to loss of resources
 - ☒ check if the system recovers properly.

Recovery Testing

- ⌘ While downloading a movie over a Wifi network, if we move to a place where there is no network, then the downloading process will be interrupted. Now to check if the process recovers from the interruption and continues working as before, we move back to a place where there is a Wifi network. If the downloading resumes, then the software has a good recovery rate.
- ⌘ Another example could be when a browser is working on multiple sessions, we can stimulate software failure by restarting the system. After restarting the system, we can check if it recovers from the failure and reloads all the sessions it was previously working on.

Maintenance Testing



⌘ Verify that:

- ☑ all required artifacts for maintenance exist
- ☑ they function properly

Documentation tests



⌘ Check that required documents exist and are consistent:

- ☑ user guides,
- ☑ maintenance guides,
- ☑ technical documents

Usability tests



⌘ All aspects of user interfaces are tested:

- ☑ Display screens
- ☑ messages
- ☑ report formats
- ☑ navigation and selection problems

Security Testing

- ⌘ **SECURITY TESTING** is a type of Software Testing that uncovers vulnerabilities, threats, risks in a software application and prevents malicious attacks from intruders.
- ⌘ This is an example of a very basic security test which anyone can perform on a web application:
 - a. Log into the web application.
 - b. Log out of the web application.
 - c. Click the BACK button of the browser (Check if you are asked to log in again or if you are provided the logged-in application.)

Regression Testing



⌘ Does not belong to either unit test, integration test, or system test.

☒ In stead, it is a separate dimension to these three forms of testing.

Regression testing



⌘ Regression testing is the running of test suite:

- ☑ after each change to the system or after each bug fix

- ☑ ensures that no new bug has been introduced due to the change or the bug fix.

Regression testing



⌘ Regression tests assure:

- ☑ the new system's performance is at least as good as the old system
- ☑ always used during phased system development.