8.7 INTEGRATION TESTING

During integration testing, different modules of a system are integrated using an integration plan. The integration plan specifies the steps and the order in which modules are combined to realize the full system. After each integration step, the partially integrated system is tested. The primary objective of integration testing is to test the module interfaces.

An important factor that guides the integration plan is the module dependency graph. We have already seen in Chapter 5 that the module

dependency graph denotes the order in which different modules call each other. A structure chart is a form of a module dependency graph. Thus, by examining the structure chart the integration plan can be developed based on any of the following approaches:

- Big-bang approach
- Top-down approach
- Bottom-up approach
- Mixed approach

Big-Bang Integration Testing

It is the simplest approach to integration testing, where all the modules making up a system are integrated in a single step. In simple words, all the modules are simply put together and tested. However, this technique is practicable only for very small systems. The main problem with this approach is that once an error is found during integration testing, it is very difficult to localize the error as the error may potentially belong to any of the modules being integrated. Therefore, debugging errors reported during big-bang integration testing are very expensive to fix.

Bottom-Up Integration Testing

In bottom-up testing, each subsystem is tested separately and then the full system is tested. A subsystem might consist of many modules which communicate among each other through well-defined interfaces. The primary purpose of testing each subsystem is to test the interfaces among various modules making up the subsystem. Both control and data interfaces are tested. The test cases must be carefully chosen to exercise the interfaces in all possible manners.

Large software systems normally require several levels of subsystem testing, where lower-level subsystems are successively combined to form higher-level subsystems. A principal advantage of bottom-up integration testing is that several disjoint subsystems can be tested simultaneously. In a pure bottom-up testing no stubs are required, only test-drivers are required. A disadvantage of bottom-up testing is the complexity that occurs when the system is made up of a large number of small subsystems. This extreme case corresponds to the big-bang approach.

Top-Down Integration Testing

Top-down integration testing starts with the main routine i.e. the root module, and one or two subordinate routines in the system. After the top-level 'skeleton' has been tested, the subroutines of the 'skeleton' are immediately combined with it and tested. The top-

down integration testing approach requires the use of program stubs to simulate the effect of lower-level routines that are called by the routines under test. A pure top-down integration does not require any driver routines. A disadvantage of the top-down integration testing approach is that in the absence of lower-level routines, many times it may become difficult to exercise the top-level routines in the desired manner since the lower level routines perform several low-level functions such as I/O.

Mixed Integration Testing

A mixed (or sandwiched) integration testing follows both top-down and bottom-up testing approaches. In the top-down approach, testing can start only after the top-level modules have been coded and unit tested. Similarly, bottom-up testing can start only after the bottom level modules are ready. The mixed approach overcomes these shortcomings of the top-down and bottom-up approaches and testing can start as and when modules become available. Therefore, this is one of the most commonly used approach to integration testing.

8.7.1 Phased vs. 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.
- In phased integration, a group of related modules is added to the partial system each time.

Phased integration requires less number of integration steps than those required in the incremental integration approach. However, when failures are detected, it is easier to debug the system in the incremental testing approach since it is known that the error is caused by the addition of a single module. However, in phased integration testing the error might be due to any of the newly added modules. In fact, big-bang testing is a degenerate case of the phased integration testing approach.