6.7.2 Black Box Testing

Black box (or **functional**) testing checks the functional requirements and examines the input and output data of these requirements (see Figure 6.30). When black box testing is performed, only the sets of 'legal' input and corresponding outputs should be known to the tester and not the internal logic of the program to produce that output. Hence to determine the functionality, the outputs produced for the given sets of input are observed.

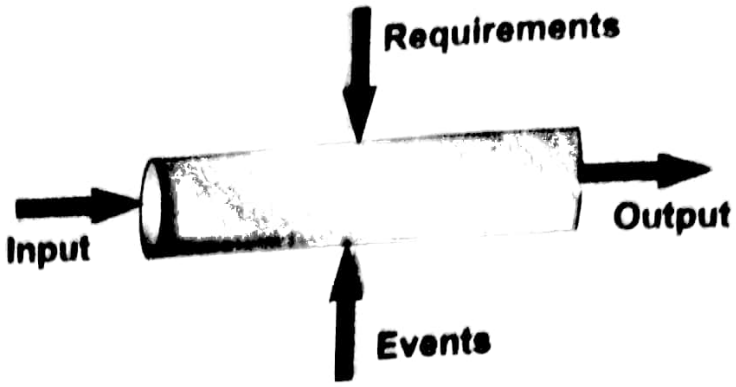

Figure 6.30 Black Box Testing

The black box testing is used to find the errors listed below (also see Figure 6.31).

- Interface errors such as functions, which are unable to send or receive data to/from other software.
- Incorrect functions that lead to undesired output when executed.
- Missing functions and erroneous data structures.
- Erroneous databases, which lead to incorrect outputs when the software uses the data present in these databases for processing.
- Incorrect conditions due to which the functions produce incorrect outputs when they are executed.
- Termination errors such as certain conditions due to which a function enters a loop that forces it to execute indefinitely.

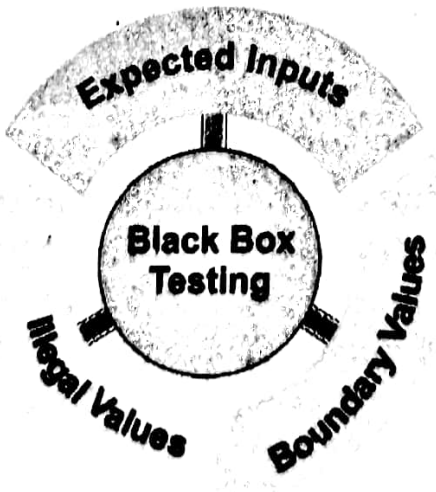

Figure 6.31 Types of Error Detection in Black Box Testing

In this testing, tester derives various test cases to exercise the functional requirements of the software without considering implementation details of the

code. Then, the software is run for the specified sets of input and the outputs produced for each input set is compared against the specifications to conform the correctness. If they are as specified by the user, then the software is considered to be correct else the software is tested for the presence of errors in it. The advantages and disadvantages associated with black box testing are listed in Table 6.9.

Table 6.9 Advantages and Disadvantages of Black Box Testing

Advantages	*Disadvantages*
• Tester does not require knowledge of internal logic of the program and the programming language used. • Reveals any ambiguities and inconsistencies in the functional specifications. • Efficient when used on larger systems. • A non-technical person can also perform black box testing.	• Exercising software for every possible test case requires a lot of time, thus, only a small number of test cases are used to test the functional requirements. • As test cases are developed without looking at the internal logic of program, testing may leave many paths in the program unexercised. • There may be duplication of test cases if tester is unaware of the test cases that have already been tried by the software developer. • Cannot be targeted towards particular code segments, hence is more error prone.