

Topics covered



- ✧ Development testing
- ✧ Test-driven development
- ✧ Release testing
- ✧ User testing

Program testing



- ✧ Testing is intended to show that a program does what it is intended to do and to discover program defects before it is put into use.
- ✧ When you test software, you execute a program using artificial data.
- ✧ You check the results of the test run for errors.
- ✧ Can reveal the presence of errors NOT their absence.
- ✧ Testing is part of a more general verification and validation process.

Program testing goals



- ✧ To demonstrate to the developer and the customer that the software meets its requirements.
- ✧ To discover situations in which the behavior of the software is incorrect, undesirable or does not conform to its specification.

Validation and defect testing



- ✧ The first goal leads to **validation testing**
 - You expect the system to perform correctly using a given set of test cases that reflect the system's expected use.
- ✧ The second goal leads to **defect testing**
 - The test cases are designed to expose defects. The test cases in defect testing can be deliberately obscure and need not reflect how the system is normally used.

Testing process goals



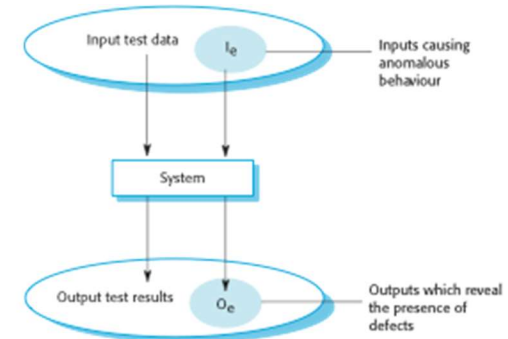
✧ Validation testing

- To demonstrate to the developer and the system customer that the software meets its requirements
- A successful test shows that the system operates as intended.

✧ Defect testing

- To discover faults or defects in the software where its behaviour is incorrect or not in conformance with its specification
- A successful test is a test that makes the system perform incorrectly and so exposes a defect in the system.

An input-output model of program testing



Verification vs validation



✧ Verification:

"Are we building the product right".

✧ The software should conform to its specification.

✧ Validation:

"Are we building the right product".

✧ The software should do what the user really requires.

Inspections and testing



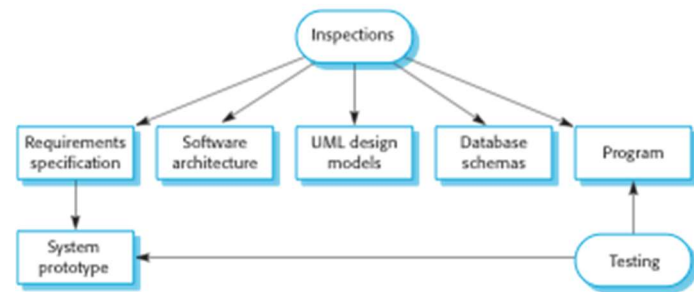
✧ **Software inspections** Concerned with analysis of the static system representation to discover problems (static verification)

- May be supplemented by tool-based document and code analysis.

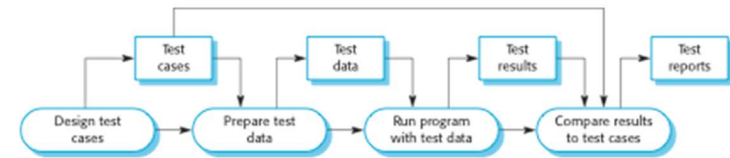
✧ **Software testing** Concerned with exercising and observing product behaviour (dynamic verification)

- The system is executed with test data and its operational behaviour is observed.

Inspections and testing



A model of the software testing process



Stages of testing

- ✧ Development testing, where the system is tested during development to discover bugs and defects.
- ✧ Release testing, where a separate testing team test a complete version of the system before it is released to users.
- ✧ User testing, where users or potential users of a system test the system in their own environment.

Development testing

- ✧ Development testing includes all testing activities that are carried out by the team developing the system.
 - Unit testing, where individual program units or object classes are tested. Unit testing should focus on testing the functionality of objects or methods.
 - Component testing, where several individual units are integrated to create composite components. Component testing should focus on testing component interfaces.
 - System testing, where some or all of the components in a system are integrated and the system is tested as a whole. System testing should focus on testing component interactions.

Automated testing



- ✧ Whenever possible, unit testing should be automated so that tests are run and checked without manual intervention.
- ✧ In automated unit testing, you make use of a test automation framework (such as JUnit) to write and run your program tests.
- ✧ Unit testing frameworks provide generic test classes that you extend to create specific test cases. They can then run all of the tests that you have implemented and report, often through some GUI, on the success of otherwise of the tests.

Automated test components



- ✧ A setup part, where you initialize the system with the test case, namely the inputs and expected outputs.
- ✧ A call part, where you call the object or method to be tested.
- ✧ An assertion part where you compare the result of the call with the expected result. If the assertion evaluates to true, the test has been successful if false, then it has failed.

Unit test effectiveness



- ✧ The test cases should show that, when used as expected, the component that you are testing does what it is supposed to do.
- ✧ If there are defects in the component, these should be revealed by test cases.
- ✧ This leads to 2 types of unit test case:
 - The first of these should reflect normal operation of a program and should show that the component works as expected.
 - The other kind of test case should be based on testing experience of where common problems arise. It should use abnormal inputs to check that these are properly processed and do not crash the component.

Testing strategies



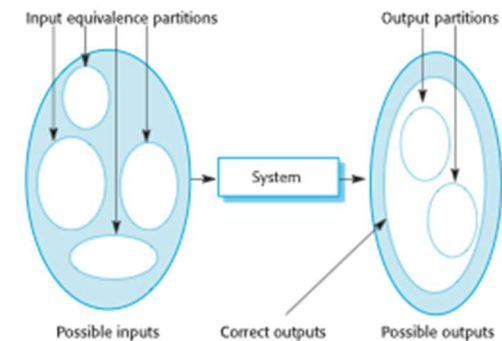
- ✧ Partition testing, where you identify groups of inputs that have common characteristics and should be processed in the same way.
 - You should choose tests from within each of these groups.
- ✧ Guideline-based testing, where you use testing guidelines to choose test cases.
 - These guidelines reflect previous experience of the kinds of errors that programmers often make when developing components.

Partition testing



- ✧ Input data and output results often fall into different classes where all members of a class are related.
- ✧ Each of these classes is an **equivalence partition** or domain where the program behaves in an equivalent way for each class member.
- ✧ Test cases should be chosen from each partition.

Equivalence partitioning



Testing guidelines (sequences)



- ✧ Test software with sequences which have only a single value.
- ✧ Use sequences of different sizes in different tests.
- ✧ Derive tests so that the first, middle and last elements of the sequence are accessed.
- ✧ Test with sequences of zero length.

General testing guidelines



- ✧ Choose inputs that force the system to generate all error messages
- ✧ Design inputs that cause input buffers to overflow
- ✧ Repeat the same input or series of inputs numerous times
- ✧ Force invalid outputs to be generated
- ✧ Force computation results to be too large or too small.