Software Testing

Unit 8: Software Testing (5 Hrs.)
Introduction; Validation and Verification Testing; Software Inspection; Software Testing Process; Development Testing; Test-Driven Development; Release Testing; User Testing

Software 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, anomalies or information about the program’s non-functional attributes.
- Can reveal the presence of errors NOT their absence.
- Testing is part of a more general verification and validation process, which also includes static validation techniques.

Program testing goals

- ✧ To demonstrate to the developer and the customer that the software meets its requirements.
  - For custom software, this means that there should be at least one test for every requirement in the requirements document. For generic software products, it means that there should be tests for all of the system features, plus combinations of these features, that will be incorporated in the product release.
- ✧ To discover situations in which the behavior of the software is incorrect, undesirable or does not conform to its specification.
  - Defect testing is concerned with rooting out undesirable system behavior such as system crashes, unwanted interactions with other systems, incorrect computations and data corruption.

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 unclear and need not reflect how the system is normally used.

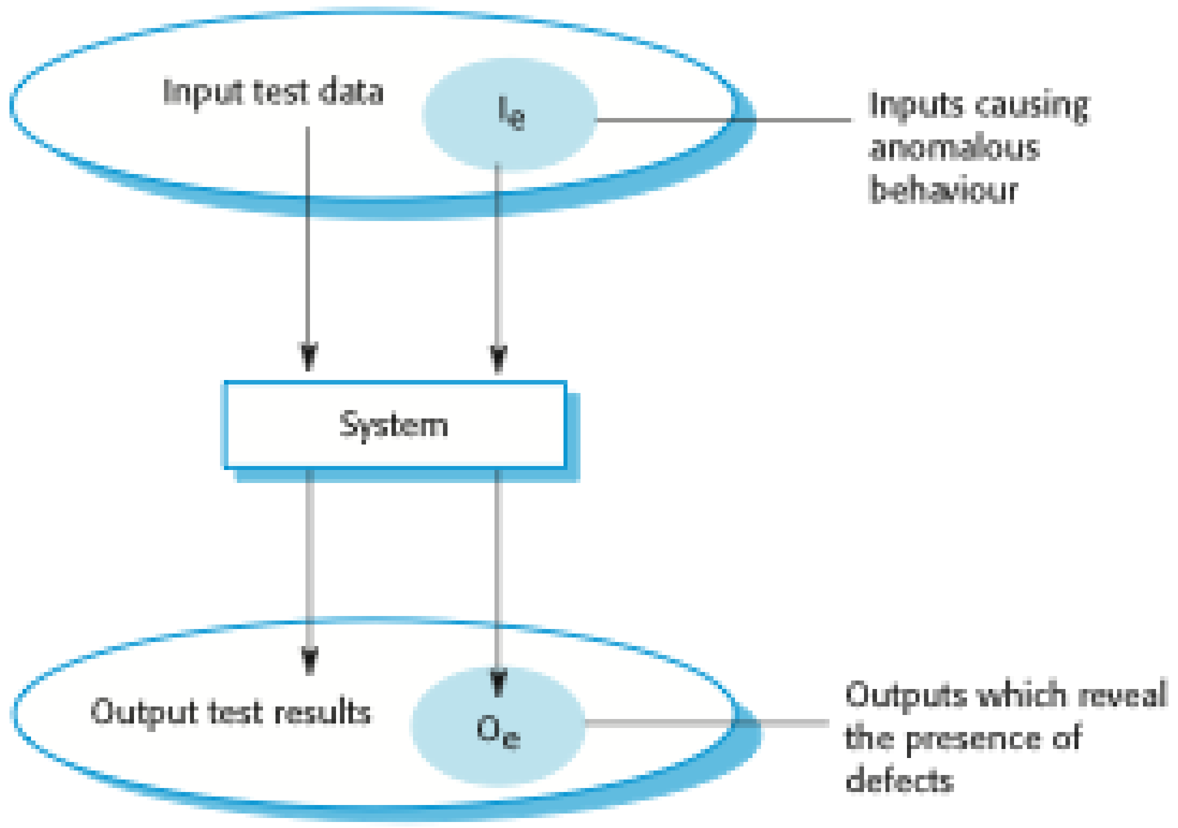
Testing process goals

✧ V
a
l
i
d
a
t
i
o
n

t
e
s
t
i
n
g

▪ T

An input-output model of program testing


Input test data
I_e
Inputs causing anomalous behaviour
System
Output test results
O_e
Outputs which reveal the presence of defects

✧ V
e
r
i
f
i
c
a
t
i
o
n
:

Verification vs validation

"
A
r

V & V confidence

✧ A
i
m

o
f

V

&

V

i
s

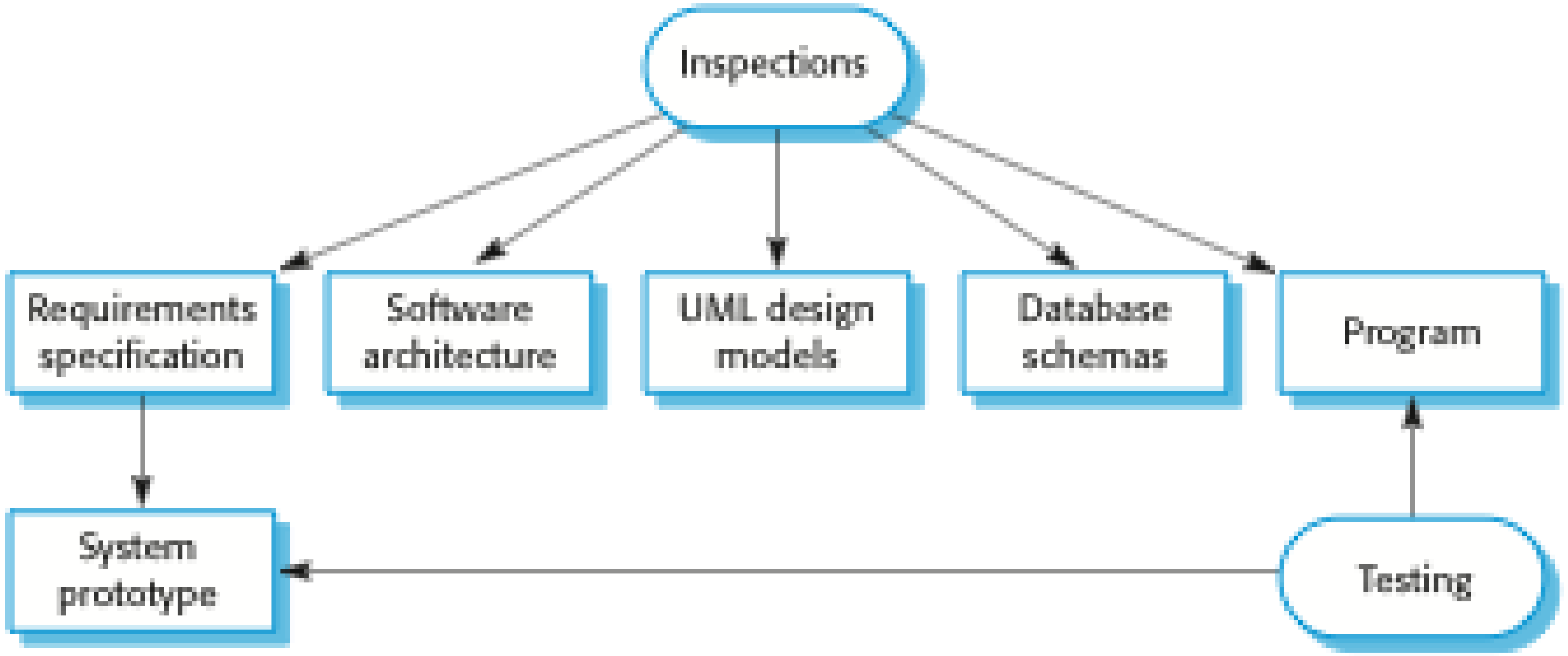
t
o

e
s

Inspections and testing

- Software inspections Concerned with analysis of the static system representation to discover problems (static verification)
  - May be supplement 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



Software inspections

✧ These involve peopl

Advantages of inspections

- ✧ During testing, errors can mask (hide) other errors. Because inspection is a static process, you don’t have to be concerned with interactions between errors.
- ✧ Incomplete versions of a system can be inspected without additional costs. If a program is incomplete, then you need to develop specialized test harnesses to test the parts that are available.
- ✧ As well as searching for program defects, an inspection can also consider broader quality attributes of a program, such as compliance with standards, portability and maintainability.

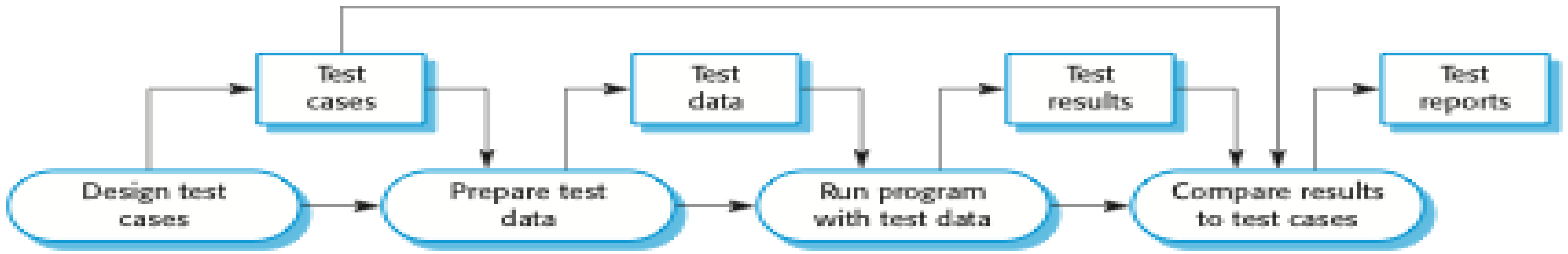
✧ I
n
s
p
e
c
t
i
o
n
s

a
n
d

t
e
s

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.

Unit testing

✧ U
n
i
t

t
e
s
t
i
n
g

i
s

t
h
e

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 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.

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.

Component testing

- ✧ Software components are often composite components that are made up of several interacting objects.
  - For example, in the weather station system, the reconfiguration component includes objects that deal with each aspect of the reconfiguration.
- ✧ You access the functionality of these objects through the defined component interface.
- ✧ Testing composite components should therefore focus on showing that the component interface behaves according to its specification.
  - You can assume that unit tests on the individual objects within the component have been completed.

Interface testing

- Objectives are to detect faults due to interface errors or invalid assumptions about interfaces.
- Interface types
  - Parameter interfaces Data passed from one method or procedure to another.
  - Shared memory interfaces Block of memory is shared between procedures or functions.
  - Procedural interfaces Sub-system encapsulates a set of procedures to be called by other sub-systems.
  - Message passing interfaces Sub-systems request services from other sub-systems

Interface errors

- ✧ Interface misuse
  - ▪ A calling component calls another component and makes an error in its use of its interface e.g. parameters in the wrong order.
- ✧ Interface misunderstanding
  - ▪ A calling component embeds assumptions about the behaviour of the called component which are incorrect.
- ✧ Timing errors
  - ▪ The called and the calling component operate at different speeds and out-of-date information is accessed.

System testing

- ✧ System testing during development involves integrating components to create a version of the system and then testing the integrated system.
- ✧ The focus in system testing is testing the interactions between components.
- ✧ System testing checks that components are compatible, interact correctly and transfer the right data at the right time across their interfaces.
- ✧ System testing tests the evolving behavior of a system.

Testing policies

- ✧ Exhaustive system testing is impossible so testing policies which define the required system test coverage may be developed.
- ✧ Examples of testing policies:
  - All system functions that are accessed through menus should be tested.
  - Combinations of functions (e.g. text formatting) that are accessed through the same menu must be tested.
  - Where user input is provided, all functions must be tested with both correct and incorrect input.

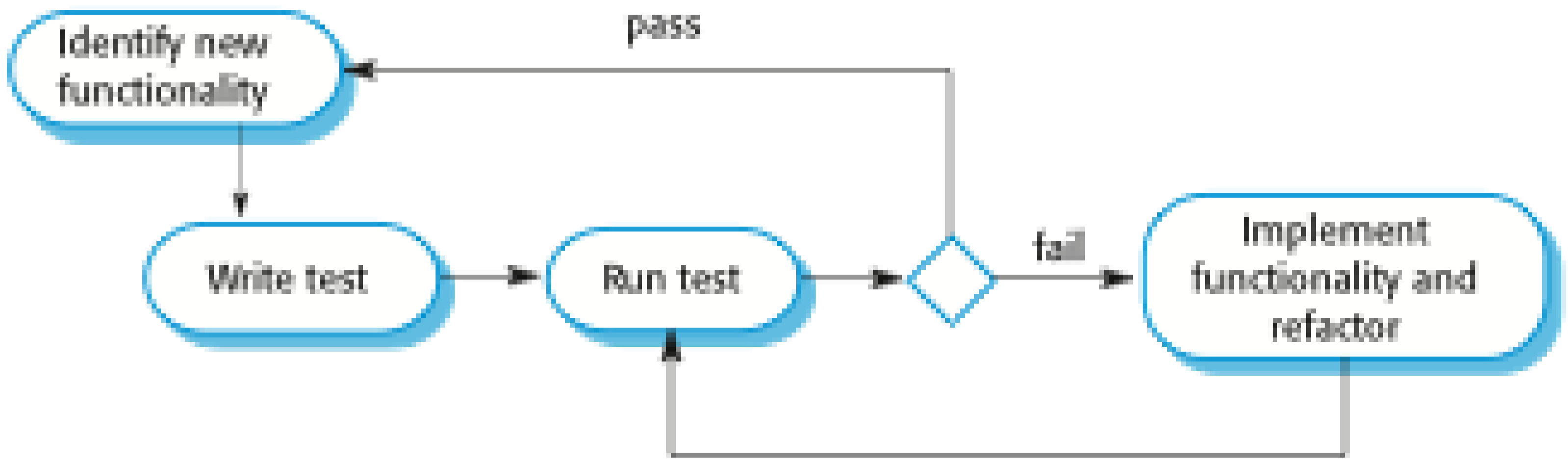
Test-driven development

- ✧ Test-driven development (TDD) is an approach to program development in which you inter-leave testing and code development.
- ✧ Tests are written before code and ‘passing’ the tests is the critical driver of development.
- ✧ You develop code incrementally, along with a test for that increment. You don’t move on to the next increment until the code that you have developed passes its test.
- ✧ TDD was introduced as part of agile methods such as Extreme Programming. However, it can also be used in plan-driven development processes.

TDD process activities

- ✧ Start by identifying the increment of functionality that is required. This should normally be small and implementable in a few lines of code.
- ✧ Write a test for this functionality and implement this as an automated test.
- ✧ Run the test, along with all other tests that have been implemented. Initially, you have not implemented the functionality so the new test will fail.
- ✧ Implement the functionality and re-run the test.
- ✧ Once all tests run successfully, you move on to implementing the next chunk of functionality.

Test-driven development


Identify new functionality
pass
Write test
Run test
fail
Implement functionality and refactor

Benefits of test-driven development

- ✧ Code coverage
  - Every code segment that you write has at least one associated test so all code written has at least one test.
- ✧ Regression testing
  - A regression test suite is developed incrementally as a program is developed.
- ✧ Simplified debugging
  - When a test fails, it should be obvious where the problem lies. The newly written code needs to be checked and modified.
- ✧ System documentation
  - The tests themselves are a form of documentation that describe what the code should be doing.

Regression testing

- Regression testing is testing the system to check that changes have not ‘broken’ previously working code.
- In a manual testing process, regression testing is expensive but, with automated testing, it is simple and straightforward. All tests are rerun every time a change is made to the program.
- Tests must run ‘successfully’ before the change is committed.

Release testing

- ✧ Release testing is the process of testing a particular release of a system that is intended for use outside of the development team.
- ✧ The primary goal of the release testing process is to convince the customer of the system that it is good enough for use.
  - Release testing, therefore, has to show that the system delivers its specified functionality, performance and dependability, and that it does not fail during normal use.
- ✧ Release testing is usually a black-box testing process where tests are only derived from the system specification.

Release testing and system testing

- ✧ Release testing is a form of system testing.
- ✧ Important differences:
  - ▪ A separate team that has not been involved in the system development, should be responsible for release testing.
  - ▪ System testing by the development team should focus on discovering bugs in the system (defect testing).
  - ▪ The objective of release testing is to check that the system meets its requirements and is good enough for external use (validation testing).

Requirements based testing

- Requirements-based testing involves examining each requirement and developing a test or tests for it.

Performance testing

- P a r t o f r e l e a s e t e s

User testing

- User or customer testing is a stage in the testing process in which users or customers provide input and advice on system testing.
- User testing is essential, even when comprehensive system and release testing have been carried out.
  - The reason for this is that influences from the user’s working environment have a major effect on the reliability, performance, usability and robustness of a system. These cannot be replicated in a testing environment.

Types of user testing

- ✧ Alpha testing
  - ▪ Users of the software work with the development team to test the software at the developer’s site.
- ✧ Beta testing
  - ▪ A release of the software is made available to users to allow them to experiment and to raise problems that they discover with the system developers.
- ✧ Acceptance testing
  - ▪ Customers test a system to decide whether or not it is ready to be accepted from the system developers and deployed in the customer environment. Primarily for custom systems.

Stages in the acceptance testing process

- ✧ Define acceptance criteria
- ✧ Plan acceptance testing
- ✧ Derive acceptance tests
- ✧ Run acceptance tests
- ✧ Negotiate test results
- ✧ Reject/accept system