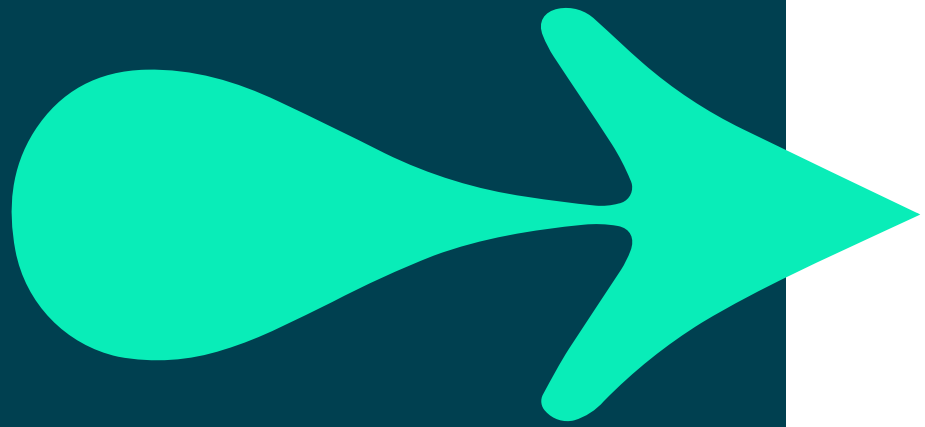
POWERING POTENTIAL

What is Testing?

Module 5 – Software Testing

MODULE OBJECTIVES

- What is software testing?
- Why is software testing important?
- What are the benefits of software testing?

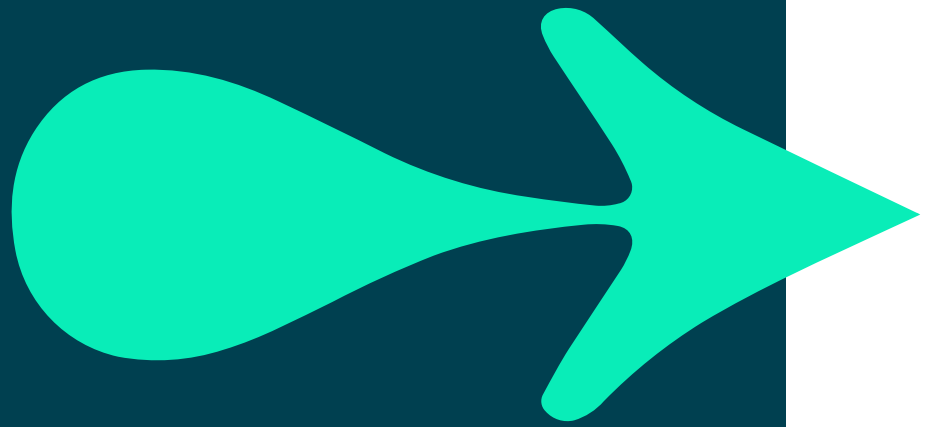


WHAT IS SOFTWARE TESTING?

Software testing:

- Checks whether the application under test (aut) meets the expected requirements
- Ensures the program code is free of:
  - Logical errors
  - Arithmetic calculation errors
  - Interface errors
  - Application errors
- Ensures different modules work well together, that the interface between them is correct
- Ensures the system as a whole works
- Eliminates runtime errors and ensures everything works on the target OS/client system

WHY IS SOFTWARE TESTING SO IMPORTANT?



Testing help to solve problems early because:

- Bugs are expensive to fix in production
- They are costly to fix
- Lives could be affected
- Airline, Transport, Hospital...
- Reputation could be affected; it takes a lifetime to establish reputation but a single day to lose it!

A few examples of disasters caused by bad software testing:

https://dzone.com/articles/the-biggest-software-failures-in-recent-years

WHAT ARE THE DIFFERENT TYPES OF TESTS?

There are many types of tests, but they can be categorised as these general types:

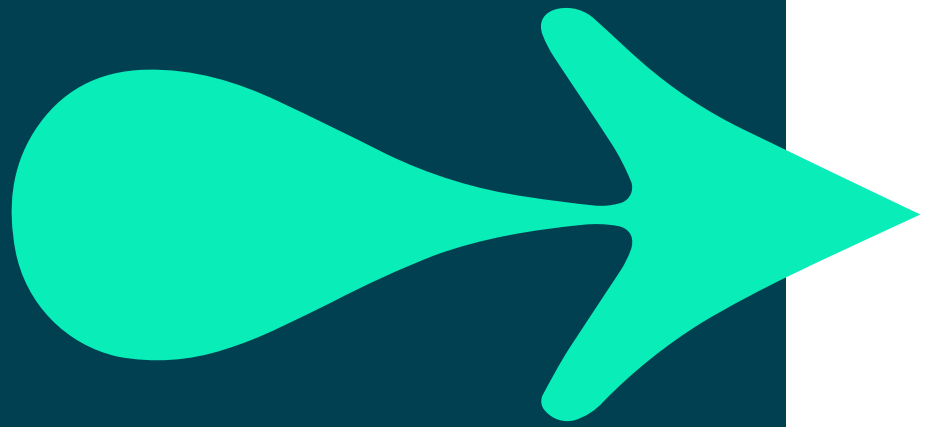
- Unit testing
- System testing
- Integration testing
- Acceptance testing

TEST CATEGORIES

There are many categories of tests, including:

- Functional testing
- Unit testing (making sure individual units, such as classes and methods, are defect free)
- Integration testing: making sure the integrated units work well together, which requires knowledge of the application's design
- User acceptance testing
- Last testing, which is usually performed by users to test the software functions
- Non-functional testing
- Regression and maintenance testing:
  - To make sure software still performs after a change
  - To make sure no new bug is introduced and the software still works after a change in code
- System testing (see the next page)

WHAT IS SYSTEM TESTING?

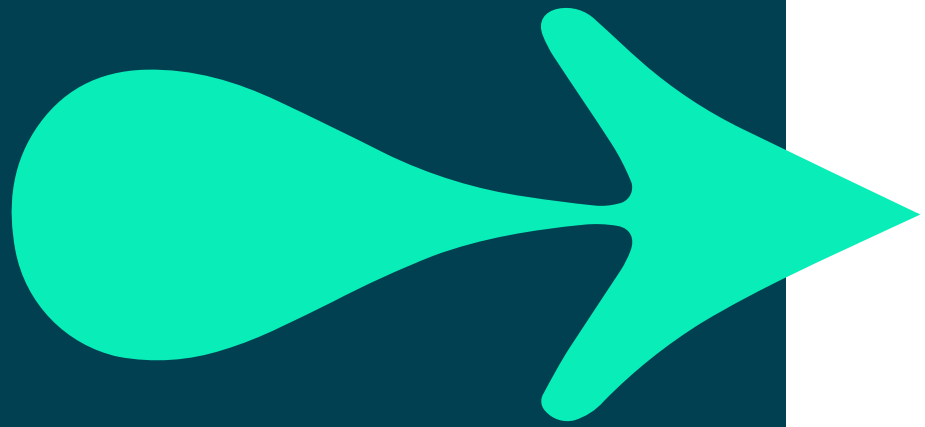


System testing is the testing of the functionality of the system as a whole.

System testing ensures that the expected functions (requirements) are carried out. For example:

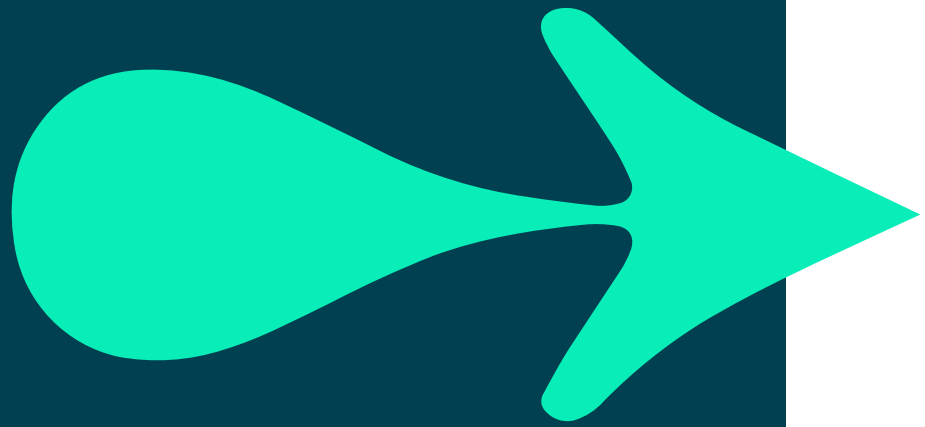
- Recoverability: can we continue after a disaster?
- Performance: e.g. can many users use the system and get a good performance?
- Scalability: if multiple users are using the system at the time, will the system perform as expected (within sla range) when the number of users or connected systems increase
  can servers scale horizontally or vertically?
- Many geographies: ensures that the system performs well across different physical distances between its components
- Reliability: ensures that the system can operate over long span of time (as defined in a sla)

SYSTEM REQUIREMENTS



- Security is a system requirement
- Authentication and authorisation
- Data validation
- Transport security
- Data protection
- Session management on web sites
- Usability
  - How usable is our system? How do we measure this?
    See: https://www.Usability.Gov/how-to-and-tools/methods/system-usability-scale.Html
- Compatibility
  - Can it run in different browsers, database, hardware, OS, mobile devices, and networks?
- And more...!

SOFTWARE TESTING – NON-FUNCTIONAL TESTS



Non-functional tests are as important as the functional tests (if not more).

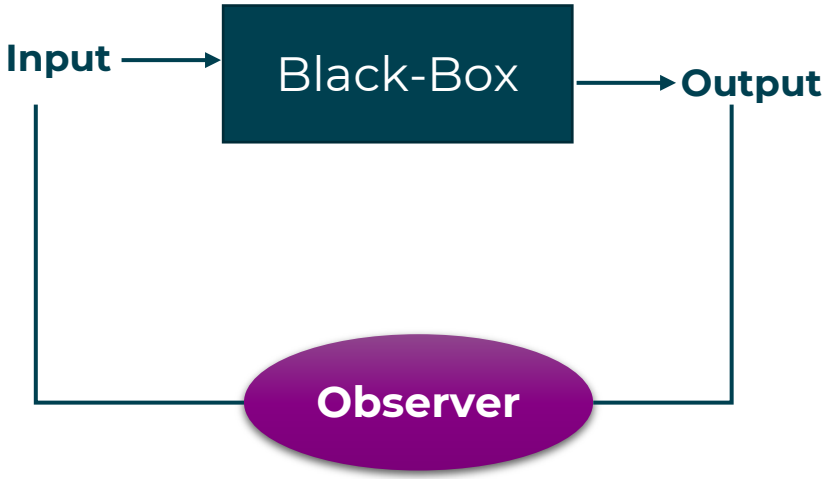
- **Availability, reliability**
  - e.g. students can access their work 99% of the time without failure between 9-5pm
- **Performance**
  - Operations per second
- **Security**
- **Scalability**
  - Scale horizontally, vertically
- **Capacity**
  - Transactions per second, response time
  - Delivering sufficient functionality
- **User experience** (UX)

What are the different types of test?

The different types of tests can be further categorised as:

- **Black Box Testing**
  - No insight into code is necessary
  - Often involves manual testing of the requirements
  - Software is seen as a black box with input and output interfaces

- **White Box Testing**
  - Code knowledge is required to create a code coverage
  - Tests the code and its logic




```
var writeResponse = function(res, str) {
   res.writeHead(200, { 'Content-Type': 'application/json' });
   res.end(str);
}
var writeJsonResponse = function(res, str) {
   writeResponse(res, JSON.stringify(str));
}

apis =
"/customerByID/\n/customers\n/orders\n/ordersByCustomerid/\n/orders\n/orderbyCus
tomerid/"
var writeHelp = function(res) {
   res.writeHead(404, { 'Content-Type': 'text/plain' });
   res.end(apis);
}

var notFound = function(res) {
   res.writeHead(404, { 'Content-Type': 'text/plain' });
   res.end("The data you have requested could not be found");
}

const server = require('http').createServer();
```

MANUAL AND AUTOMATED TESTS

Tests can either be performed manually by a tester or they can be automated to run with little manual intervention.

Let's explore the pros and cons of **manual** testing and **automated** testing.

In real life, you would use a mix of these when required, but favour automation. Why? Let's see...

MANUAL TESTING METHOD

- It can NOT be designed and performed by anyone
- Requires deep analysis of the requirements and specialised knowledge of testing
- It is NOT an easy discipline
- It is sometimes better than automated testing because it uses the intuition and the expertise of the tester
    - We often require both types of test, manual and automated
- It is sometimes easier to manually test, at least some modules of your application
    - Consider setting up a complex and costly test bed for automation
- Even small changes in the UI requires a re-write of an automation test script, whereas a manual tester would be less affected by such changes
- May not catch 100% of the bugs

Exploratory tests can be included by an expert manual tester to find bugs that are not directly related to the requirements.

AUTOMATED TESTING

- Manually checking every method, field, and every track through your software may not be possible
  - It is time-consuming, costly, and boring!
- Encourage developers to make rapid changes and run all tests to make sure the software still works
  - Automated tests are necessary as part of a ci/cd process
- Best to add new tests to test any new code (of course!)
  - By adding new tests, you increase code coverage
- Creates a reliable, and consistent result (when designed well)
- Can be a source of reusable test scripts

AUTOMATED TESTING...

- Some types of tests are not easy to perform using the manual testing methods and there are tests which are impossible to carry out using a human operator. These include:
  - Performance tests
  - Threading and parallel execution
  - Load tests
  - Penetration tests
  - Brut attacks

Just be aware that any automation test needs to be simple and bug free!

AUTOMATION TOOLS AND TESTS

In this course, you will use automation tools to test application code and behaviour.

Unit testing

- Units of code (methods and classes)
- Using java, javascript, python and C#
- Integration testing to make sure all the fully tested code modules work together
- Creating mocks and stubs for modules not yet available or not viable to integrate with during unit testing

Regression testing

- Make sure no damage is done by adding new code

Integration testing

- Making sure software modules works well together
- UI and business layer, business layer and database

SOFTWARE TESTING PRINCIPLES

Exhaustive testing is not possible because of many constraints such as:

- Time
- Cost of exhaustive testing
- Knowledge of code complexity

Effort must be based on risk factor

- Based on tester's experience

An example of why exhaustive tests are so hard:
Every time I move a file I must test:

- Security rights
- Destination folder exits
- Destination has enough space
- Destination has a read-only file with the same name
- Then, start writing code to test the logic of the move

More testing is required if part of a multi-threaded app.
Endless possibilities to keep an army of testers employed!

SOFTWARE TESTING PRINCIPLES...

Defect clustering

- Bugs keep each other company in small number of modules
- 80% of bugs appear in 20% of modules. Concentrate on these!
- Use your experience to concentrate effort on the right modules

Pesticide paradox

- If you use the same test, using the same test data, developers will make sure the code pass the same tests!

- **Solution:** use a different set of data to challenge your tests and find new bugs!

SOFTWARE TESTING PRINCIPLES...

Tests show up all the defects:

- Testing reduces the possibility of defects, but it does not guarantee a defect free code
- 100% coverage is not always possible
- Absence of error fallacy

Code could be correct, but it may not satisfy the user requirements!

- Could be testing the wrong things, wrong modules, wrong requirements!
- Could be written by a biased developer

SOFTWARE TESTING PRINCIPLES...

Test as early as possible

- This one is self-explanatory!
- Sdlc during requirement gathering and right through every stage
- Software testing should never be an after thought. It is an important part of software development
- Start designing test right at the start of an SDLC and then throughout
- Define requirement and uats, unit test, integration...uat tests before release

Context sensitivity

- The way you test an e-commerce site is different than a POS or e-dynamic site

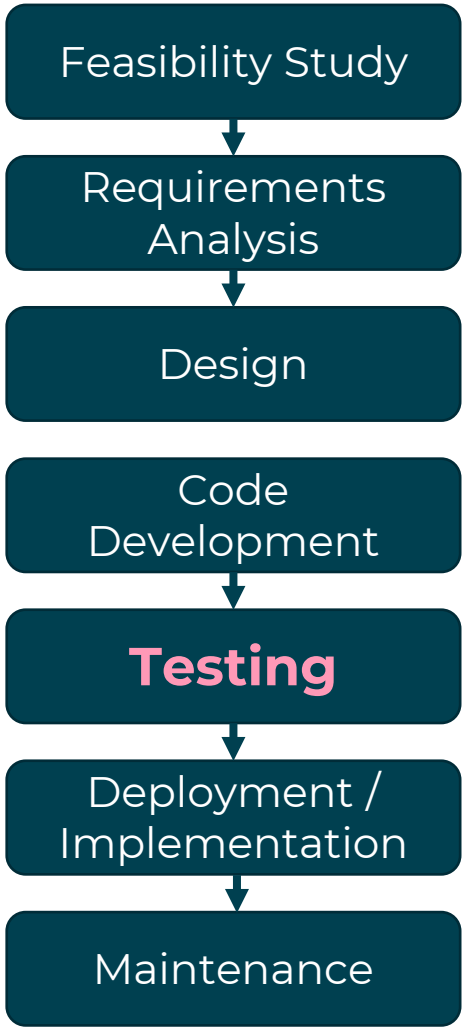
SDLC – Waterfall tests

A typical sequence in ADLC is:

- **Gather requirements** and **analyse**
- **Design** your code, OS, Modelling techniques like UML, choose a language, database, etc.
- **Build** modules of code
- **Test** everything
- **Deploy**
- **Maintain** (new requirements, enhancements, and bugs)

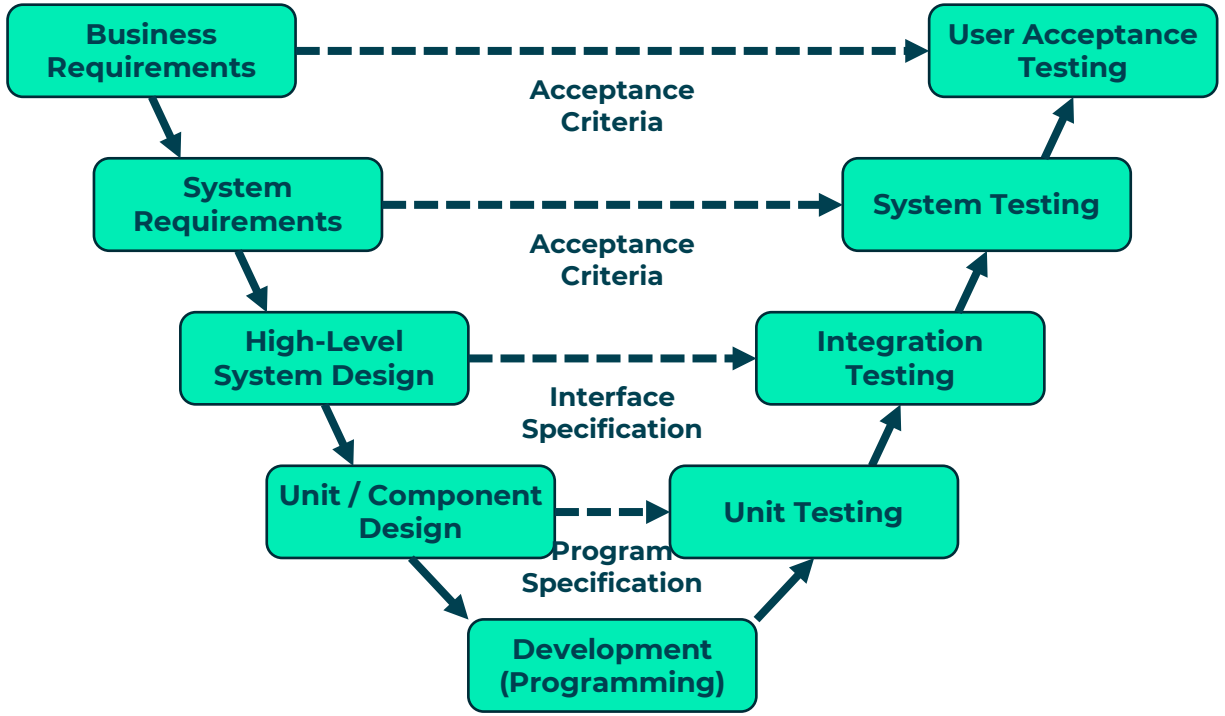
The main problem with this mode is changes in requirements:

- Changing software in late stages is very expensive
- Test early and pay less!



How can V-Model help with testing?

- V-Model of development develops a test at each stage
  - Requirements gathering and analysis System testing
  - High-level design of modules Integration tests
  - Low-level code design Unit tests
  - Ensures no requirement is missed or ignored
  - Can be used in iterative phases with each phase following a V-model testing strategy




End of Section