AH Computing
Science
TESTING & DOCUMENTATION
Testing & Documentation (cfe H)

- Test Plans
- Comprehensive Testing
- Systematic Testing
  - Syntax
  - Execution
  - Logic Errors
- Testing tools and Techniques
  - Dry runs
  - Trace Tools
  - Breakpoints
Testing & Documentation

- Component Testing
- Integration Testing
- Acceptance (Beta) Testing
- Usability Testing
- Accessibility Testing
Testing can only demonstrate the **presence of errors** not the **absence of errors**

Testing must be both **Systematic** and **Comprehensive**

The **Test Plan** is a document created:

- To systematically and comprehensively test the system
- Ensure the system **matches the original specification**
- Is written at the **design stage**
- **Includes the software specification** to measure results against
- Include a **schedule and time frame** for testing
- List **what has and has not to be tested** – it must be exhaustive
- Include **test data and expected results and outcomes** *(Normal, Extreme & Exceptional)*

This stage of testing is known as **Alpha testing**
Testing & Documentation
Comprehensive/Systematic (cfe H)

- **Comprehensive testing** means that every aspect of the system has been tested (module, subsystem & System)

- **Systematic testing** means that the tests are done in a way that is planned and documented
  
  - Syntax error - an error where the code is grammatically incorrect and cannot be translated
  
  - Execution error – an error that only appears when the code is run rather than at translation

  - Logic errors – The statement is grammatically correct but incorrect results are produced due to the logic of the statement or structure being incorrect
Testing & Documentation

Testing Techniques and Tools (cfe H)

The following tools and techniques can be used to help the testing process:

- **Dry Runs**
  - A manual run through of the program using pen and paper, noting variable values and outcomes. Checks that the code will do what is expected.

- **Trace Tables**
  - Similar to a dry run, a table of test values and expected results is constructed. Most often used to test at the module/subprogram level.

- **Trace Tools**
  - Some programming environments have TRACE facilities built in. Variable can be selected and their values will be displayed as the program executes.

- **Breakpoints**
  - Some programming environments allow the programmer to set breakpoints in the program. The program will stop executing at this point and the values of variables noted.
Component Testing –
- is where modules that make up the system are tested independently of each other.
- Takes place during the development process.
- Does not need the entire system to be complete.
- Testing individual components (modules) makes it easier to identify errors than testing the whole system together.
- Programmer create ‘stubs’ pieces of code to feed the component the values it requires.
- Programmer may create a Driver - a piece of code designed to test a component.
- Component testing should be documented.
Integration Testing –

- Test if already written and tested components work together correctly
- Data flows in the expected manner and data integrity is maintained
- Integration test plans typically include
  - The original specification
  - Test documentation of the component testing
  - List of test data to be used
  - Conditions for the integration test to take place (which modules must be complete)
  - Criteria for a successful test

Component and Integration testing is known as Alpha testing and is performed by members of the development team or a test group appointed by the developers.
Acceptance Testing

- Testing which is done by the client/client group or by potential customers.
- Completed after development to see if the software is acceptable to the customer.
- Software should have most errors identified and corrected during development and alpha testing.
- Some software companies release an Alpha version of the software to see get feedback from clients.
- As Beta testing is done by actual users of the software new problems and bugs may be spotted.
Usability testing

This type of testing checks that the system is as easy to use and understand as possible. The following criteria are often used to assess Usability:

**Appropriate** - is the interface appropriate?, does it mirror the real world so that it is familiar to the user. “Skeuomorphism” has become popular where application mimic older technologies e.g. a camera app may include shutter controls etc that mimic a real camera.

**Customisable** – The interface is customisable to the needs and abilities of the user. This may include shortcuts for experienced users. Common functions used by the user may be added to tool bars and least frequently used controls removed.

**Accessibility** – The interface should be customisable to take account of user who may require a screen reader, larger fonts, contrasting colour schemes to enable them to access the program

**Consistent** – controls should work in a logical way and related activities performed in similar ways

**Controllable** – Critical operation should always allow the user to cancel the operation incase a mistake has been made

**Helpful** - help for users should always be available and easy to access.

▶ Click [here](#) for more on the history of user interfaces
Testing & Documentation

Accessibility Testing

Accessibility

When assessing how accessible a system is the following should be considered:

- Level of users motor control
- Visual impairment
- Colour blindness
- Hearing impairment
- Epilepsy
- Dyslexia

Where possible developers should try to build as accessible a system as possible.

Developers should document Acceptance, Usability and Accessibility tests.