* Testing is required to ensure that all functionality of software works as required.
* Decides where the product is fit for purpose.
* Finds faults early, does not fix the product, only outlines the areas that need improving.
* Gives an assessment of quality, understands areas where it is lacking

Why Test?

* Things go wrong
* Gives user peace of mind
* Reduce risks, minimise them
* Assess the quality of the system, doesn’t improve it
* Exhaustive testing:
* Protect against things such as SQL injection
* Error (mistake), leads to defective or faulty software
* Defect (fault), leads to a runtime error
* Failure, runtime error

Debugging and Testing

* Testing: systematic exploration of a component or system with the main aim of finding and reporting defects
* Debugging: Used by developers to identify the cause of bugs or defects in code and undertake corrections (not testing)

7 Testing Principles

1. Testing shows the presence of bugs
2. Exhaustive testing is impossible
3. Early testing: cost of bug fixing increases the later they are found
4. Defect clustering: Pareto Principle 80/20
5. The pesticide paradox
6. Testing is context dependent
7. Absence of errors fallacy: Just because you haven’t found the bug does not mean it is not there

Developer/Tester Mindset

* The approach of a tester: Developer can take things to heart about their software
* The approach of a developer: Make sure that the system they are delivering is robust and stable
* Using these different viewpoints to mitigate risk and increase quality

Fundamental Test Process

* Test planning and control
  + Determines what is going to be tested
  + How it is going to be tested
  + Who is doing the testing
  + How it will be achieved
  + You will also be **determining your exit criteria** (checklist, definition of done)
* Test analysis and design
  + The fine detail of what to test
  + Test conditions, cases and procedures
  + **The Test Base is reviewed**
  + We look at how to combine test conditions into test cases
* Test implementation and execution
  + This is the most viable part of testing
  + Prioritising test cases
  + Creating test suites from collected test cases
  + Environment set-up
  + Log testing activities and defects
  + Running tests
* Evaluating exit criteria and reporting
  + Has the criteria been met to satisfy stakeholders?
  + Does the system do as expected?
  + Determine if more tests need to be made
  + Is the system in a ready state for release?
* Test closure activities
  + Make sure docs are up to date and archived
  + Passing over testware to maintenance teams if available
  + Lessons learnt

Test basis (requirements, specification, user stories, market research)

Test suite (All of the test cases going to do during that phase of testing i.e. during the sprint of a project)

Methodology you choose depends on factors such as

-

-

-

* The Waterfall Model: Relevant when there are strict requirements such as laws that will not change)
* The V-Model (Requirement specification, Functional specification, Technical specification, Coding, Unit testing, Integration testing, System testing, Acceptance testing)
* Incremental (Agile)

Page 41

* Capture of user needs
* Definition of functions required to meet user needs
* Technical design of functionality
* Detailed design of each module or unit

Validation = Check the documents against the original User Requirements to make sure that the requirements are still being satisfied.

Verification

* Current
* No information has been lost
* Contains no ambiguities or errors

It is important to remember that each level of testing has objectives specific to that level

* Aka Component/Program/Module testing
* Each Unit of Code is usually created in isolation (for integration at this stage)
* Each of those units of code will relate to a certain function or area and be tested by the developer to make sure it works
* As more code is developed, more and more tests are created

Units are integrated and tested together.

There are two types to remember:

* Big Bang Integration: Tests all together. Used when time constrained and simple system.
* Incremental Integration: Top down, Bottom up, Hybrid
* See Top Down in book

Top Down

Advantages

* Useful in creating software intended to be generic
* This will allow for early demonstrations of the products functionality
* May help identify requirement changes and issues

Disadvantages

* Stubs a create a lot of work
* Stub definition can be difficult
* Reproducing test conditions may not be possible

Bottom Up

Advantages

Disadvantages

System Testing

* Looks at behaviour
* Interacting with system as user would
* Harnesses complete end-to-end scenarios in the way that the customer would use the system
* Using it as. Intended to be used

You can’t test everything or use every type

Things that affect type of system testing done

* Size of the company
* Time available for test
* Resources available
* Learning curve

Different approaches for this type of testing

Risk-based testing

Requirements-based testing

Business processes-based testing

Use case testing

Acceptance Testing

Testing Against User Expectations

* Does the system meet all requirements
* This is a question that should have been asked frequently throughout the cycle
* Typically verifies the functional fitness of the system for business users
* **Note:** Acceptance testing can take place before system testing starts

It is a crucial element for making sure the product will be functional in its perceived environment

Additional aspects to think about this point are:

* Back-up
* Procedures for disaster recovery
* End user training
* Maintenance procedures
* Security procedures

Contract and Regulation Acceptance Testing

Contractual – Acceptance criteria outlined in a contract

Regulations – Some industries have rigid regulations that must be aided by:

* Government
* Legal
* Safety reasons

Ignorance is not a defence that can be used. It is your responsibility to work within the boundaries of the law

Alpha/Beta Testing

* Alpha – Performed at the developing site, but not by the developing team
* Beta – Performed by the customers at their own location

Allows for feedback from potential or existing customers

Types of Testing – High Level

* Functional – Specification based, black-box (can be done at all levels)
* Non-Functional – Performance, Usability (often used to check the readiness of the system)
* Structural – Control Flow, menu structure, white-box (can be done at all levels)
* Change Related Maintenance testing) – Carried out on a live system

Structural (White Box)

How the code makes the functionality work. What is happening ‘under the hood’

* Decision Coverage
* Statement Coverage

Confirmation and Regression Testing

* Retesting (Confirmation) – After a defect has been fixed, the software should be retested to confirm original defect has been removed
* Regression testing – Carried out on every other part of the system to check that a fixed defect hasn’t changed other parts of the system
  + Repeated testing of already tested program
  + Performed when software or environment is changed
  + Based on risk
  + Performed at all levels (functional, non-functional, structural)

Change Related Maintenance Testing

Testing a live environment when there has been:

* Modification
* Migration
* Retirement of software

Impact analysis (risk) and metrics from previous projects are very important in this area

* They help estimate the amount of re-testing and regression testing
* What are the possible consequences?
* What areas will remain unchanged?