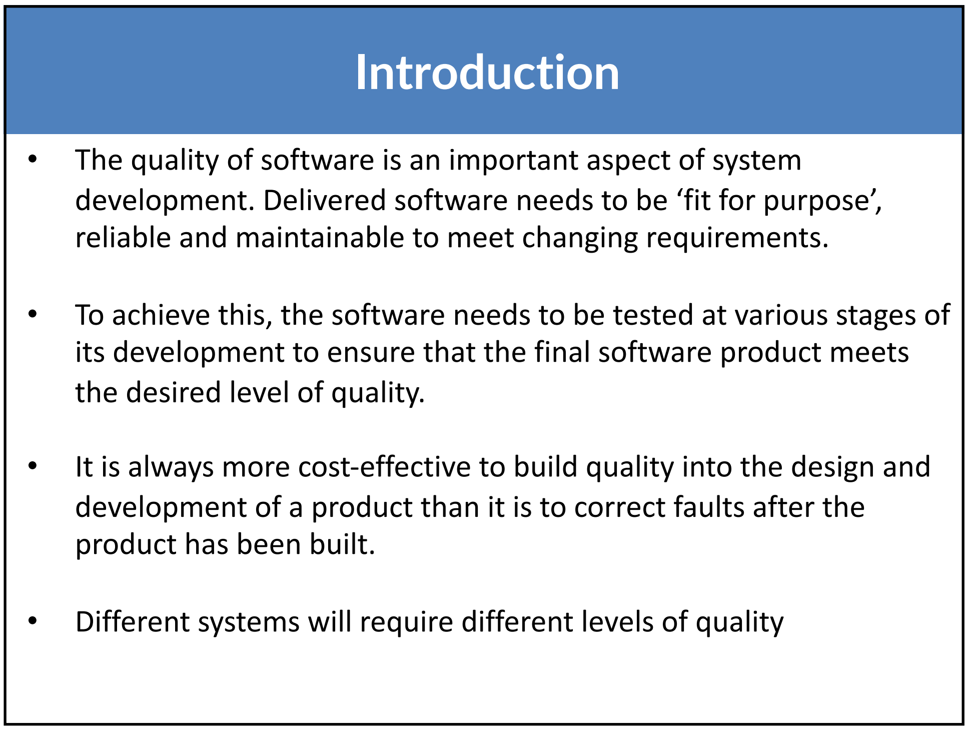
IF3141 SISTEM INFORMASI
Quality & Testing

Semester I 2023/2024

1

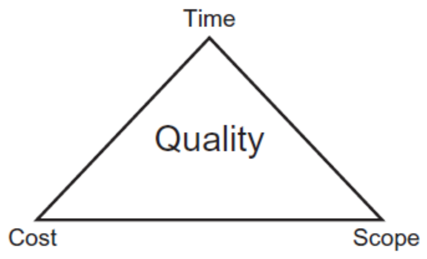


Introduction

- The quality of software is an important aspect of system development. Delivered software needs to be ‘fit for purpose’, reliable and maintainable to meet changing requirements.
- To achieve this, the software needs to be tested at various stages of its development to ensure that the final software product meets the desired level of quality.
- It is always more cost-effective to build quality into the design and development of a product than it is to correct faults after the product has been built.
- Different systems will require different levels of quality

2

Quality Triangle



- When developing systems, there is inevitably a trade-off between time, cost and scope, and these factors have a direct impact on the quality of the system being developed.

- Understanding the impacts of changes to time, cost and scope on quality is a key skill for software development managers when negotiating with key stakeholders.

3

Quality Triangle

- **Time** – extend the deadline by, say, another week. This however, will incur additional cost, and with time pressure on the software developer, the quality of the code might reduce.

- **Cost** – add another software developer to the task. However, the second software developer might be inexperienced in the particular type or area of the software, which again might result in poor quality code.

- **Scope** – it might be possible to agree with the users and sponsor that the incomplete software program is used, with an acceptance of higher number of defects.

4

What is Quality?

- Some terms that are sometimes used to describe quality include:
  - 'excellence';
  - 'fitness for purpose';
  - 'conformance to requirements'.
- Based on ISO 8402:1994, quality is

The totality of features and characteristics of a product or service that bear on its ability to satisfy stated or ***implied needs***

- 'implied needs' can easily be misinterpreted. For example, a user requirement for a listing of products might imply that they should be shown in some kind of sequence, for instance by product category, whereas the system might show the products listed in product code sequence

5

Good Software Characteristics

- **Meets functional requirements**
  - The system must do what it is specified in the requirements document. For example, if there is a requirement to display customer details, then the system must provide this feature.
- **Meets non-functional requirements**
  - This includes characteristics such as reliability, response time, availability, ease of use, and security.
- **Has inherent qualities**
  - This includes characteristics such as maintainability, efficiency, flexibility/expandability (scalability), reusability, testability, modularity, well-documented, portability, and so on.

6

Testing

- Once the required level of quality characteristics has been defined and agreed, the question then remains how to ensure, or test, that the software exhibits these characteristics.

- Testing is a crucial activity in the Systems Development lifecycle, but many people mistakenly believe that software programs can be fully tested and that, with this complete testing, programs will work correctly and with all errors removed.

7

Complete testing is impossible

- there are too many paths and combinations of paths through the software to test completely;
- the domain of possible inputs is too large;
- user interface issues are too complex;
- there are too many layers of interconnecting hardware components and software applications and modules to replicate into a test environment;
- the budget, time and resources that are available limit the amount and extent of testing that can be carried out.

- While most hardware and software components nowadays are developed and tested using software tools, these tools themselves would have to be tested ☺

8

Complexity Example

- **An internet-based application running on a mobile phone**
  - The layers of software and hardware include:
    - the software application itself;
    - the mobile phone's operating system;
    - the user interface;
    - the physical connectivity of key depressions, swipe, screen-touch sensitivity;
    - connectivity to the local internet access points and routers;
    - the hardware, software and operating system on the router;
    - connectivity to the internet service provider, and onward connectivity to the host service provider, and all of the associated hardware and software.

9

Testing Principle

- Therefore, a system cannot be guaranteed to be fault-free. This means that the users' expectations have to be carefully managed and the testing activity itself has to be focused towards identifying as many defects as possible, or picking up critical defects, within the time and budget available.

10

Testing Principle

- Testing principle according to ISTQB (International Software Testing Qualifications Board):
  1. Testing shows the presence of defects
     - Testing can show that defects exist, but it cannot prove that the software has no defects. Successive testing reduces the probability of defects, but cannot guarantee that all defects have been removed.
  2. Exhaustive testing is impossible
     - Even with enough time to allow every path through the software to be tested, the variety of user inputs, and complexity of layers of hardware and software components and interconnections makes exhaustive testing impossible.
  3. Early testing is beneficial
     - Errors and faults found earlier in the development process are easier and cheaper to fix than if found later in the process.

11

Testing Principle

4. Defect clustering
   - Defects are not evenly spread through a system – they tend to cluster together. This could be because modules have been written by a particular programmer or a particular software package or interface may exhibit similar defects.
5. There is a ‘pesticide paradox’
   - Using the same tests over and over again is unlikely to reveal new defects. To overcome this, test cases need to be reviewed and revised, and new test cases need to be written to test different paths and areas of the software.
6. Testing is context dependent
   - The type of tests will depend on the system being tested. For example, the tests needed for an internet application mortgage application system will be different to the tests needed for an aircraft navigation system.
7. Absence of errors fallacy
   - The system must meet the needs of the users; having a defect-free system is not in itself sufficient.

12

Testing Goals

- ensure that the quality of the final system is as high as possible by:
  - **removing errors** – mistakes made by humans;
  - **removing defects** – faults in software code, also known as bugs;
  - **preventing failure** – preventing the system from failing to do what it should do, and preventing the system from doing something it should not do.

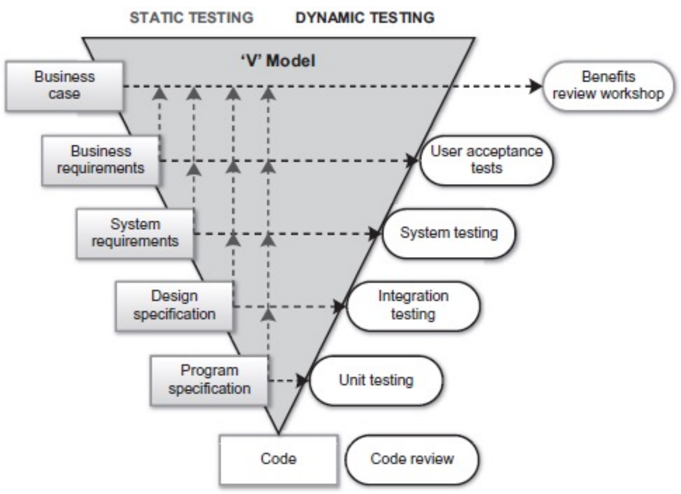
13

Static Testing vs Dynamic Testing

- **Static testing** is the term used to identify and correct errors, that is, mistakes made by humans, such as in the production of documents, writing of program source code, creation of lists of data, and so on. Static testing specifically applies to any form of testing where the program code itself is not exercised.
- **Dynamic testing** is the term used to identify and correct defects, and specifically applies to any form of testing where the program code is exercised with the use of test data.

14

Static Testing & Dynamic Testing in V Model



15

Static Testing

- Static testing uses techniques that do not exercise the program code, and as such can be applied early in the lifecycle before the code is even written.
- Static testing primarily involves the use of reviews of documents and other artefacts as they are produced. These are compared against documents and artefacts produced in earlier stages of the lifecycle for completeness and correctness.

16

Static Testing

- The products that should be reviewed using static testing include:
  - **Requirements specification**
    - checked against the project initiation document (PID) or terms of reference (TOR), and against the business case,
  - **Functional specification**
    - checked against the requirements specification for technical correctness and feasibility, and to ensure that they correctly specify how the system is to be built.
  - **Design specification**
    - checked against the functional specification and requirements specification for technical accuracy and completeness and to ensure that the design will lead to the system features and behaviour as envisaged in the functional specification.
  - **Module or program specification**
    - checked against the design specification to ensure completeness, maintainability, efficiency, reliability, and so on. They should also be checked to verify that data is correctly passed to and from other programs.

17

Review Technique for Static Testing

- **Informal review**
  - This is usually a peer review, for example by one analyst reviewing another analyst's work, or one programmer reviewing a source code listing produced by another programmer.
- **Walkthrough**
  - the product (for example, the requirements specification) is walked through line by line, section by section, or page by page, to identify any errors.
- **Technical review**
  - the product is reviewed by a group of technical experts typically using checklists of common problems to identify errors.
- **Inspection**
  - formal review meeting, with clearly defined roles and objectives

18

Dynamic Testing

- The purpose of dynamic testing is to check the quality of the software by executing the program modules to identify any defects found in them.
- The software is executed in a number of progressive stages, with each stage building on the confidence of the system gained in preceding stages.

19

Type of Dynamic Testing

- **Unit testing, or module or component testing**
  - Each completed program module, or unit, is tested against its module or program specification to check that, in isolation, it does what it is supposed to do
- **Integration testing**
  - The individual modules are progressively integrated together and tests are carried out to ensure that, for example, the modules pass data back and forth between themselves correctly, and that they comply with the design specification.
- **System testing**
  - The system is tested as a whole by testers who check that the system does what it is supposed to do, and as detailed in the functional specification
  - Tests are designed to 'break' the system as well as to check its correct functioning, and include testing the system against non-functional requirements such as performance, reliability, security, and so on.
- **User acceptance testing**
  - the users test the system with the aim of ensuring that the system delivers the features needed to support business processes and data, and that the system conforms to the requirements specification.

20

Technique of Dynamic Testing

- **Black-box techniques**
  - The tester is not concerned with how the program works, but is only interested in whether the outputs are correct based on the inputs.
  - Black-box testing is typically used in **user acceptance testing**, **system testing**, and **integration testing**.
- **White-box techniques**
  - White-box testing is concerned with the internal working of the system. By examining the program code or module specification, the tester would design tests and test data that would exercise the program logic and data processing.
  - White-box testing is usually employed in **unit testing**.
- **Experience-based techniques**
  - The experience of key techniques stakeholders is used to identify areas of the system that are likely to uncover defects.
  - For example, users may know of particular business processes and scenarios that are complex, or service managers may identify certain areas that impact service levels; designers might know of interfaces with external system that have proved problematic in the past.

21

Re-testing

- Re-testing is the term that is used to describe the test applied to any part of a system that has been changed. Its purpose is to check that changes have been made correctly.
- Quite often, as a result of carrying out dynamic testing, programs need to be corrected and re-tested in order to remove any defects found.

22

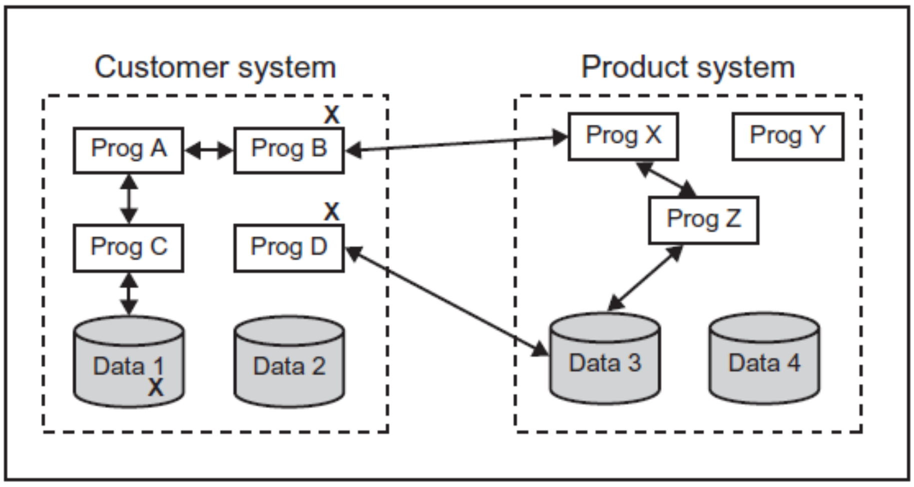
Regression Testing

- Unlike re-testing, which is concerned with testing the changed parts of a system, regression testing is concerned with making sure that the unchanged parts of the system have not inadvertently been affected and still function as they should.

- As regression tests are based on the unchanged parts of the system, they lend themselves to be designed and built once and then used repeatedly. Therefore, regression tests are ideal candidates for automation, using a test tool. The tests can be stored within the tool and executed whenever they are needed.

23

Example



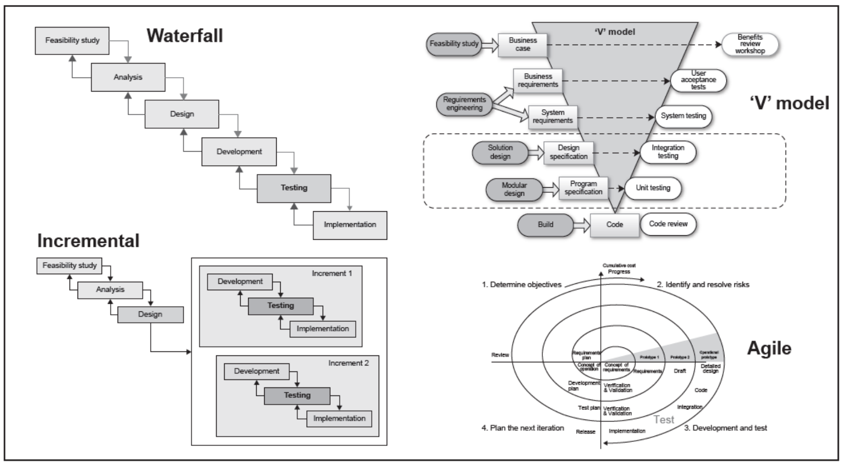
24

Testing in SDLC

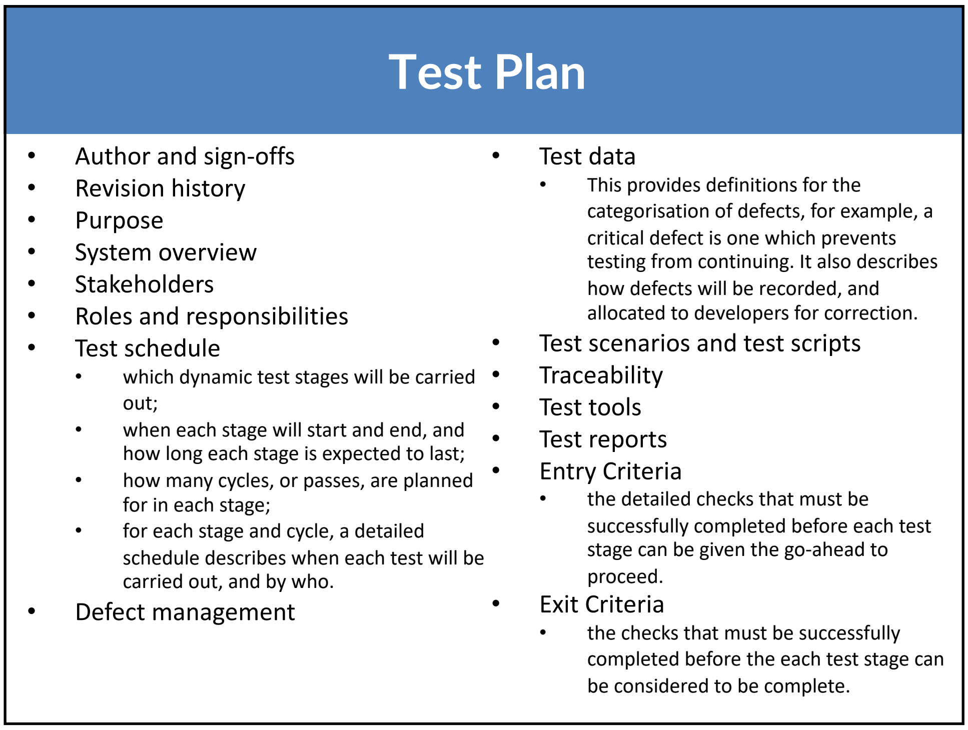
- In the Waterfall model, testing is shown as a discrete stage, between development and implementation, indicating that all testing activity is carried out after the solution has been developed and before it is deployed for operational use.
- In the Incremental model, testing also takes place at the stage between development and implementation but is repeated for each increment.
- In the Agile model, testing is a continuous activity and, through the building of prototypes, is used to help develop an understanding of the requirements as well as the capability of the technology. Successive prototypes are put through greater and more rigorous tests, which are based on two key principles:
    - **Verification** – that the system is being built in the right way; and
    - **Validation** – that the right system is being built.

25

Testing in Lifecycle



26



Test Plan

- Author and sign-offs
- Revision history
- Purpose
- System overview
- Stakeholders
- Roles and responsibilities
- Test schedule
  - which dynamic test stages will be carried out;
  - when each stage will start and end, and how long each stage is expected to last;
  - how many cycles, or passes, are planned for in each stage;
  - for each stage and cycle, a detailed schedule describes when each test will be carried out, and by who.
- Defect management
- Test data
  - This provides definitions for the categorisation of defects, for example, a critical defect is one which prevents testing from continuing. It also describes how defects will be recorded, and allocated to developers for correction.
- Test scenarios and test scripts
- Traceability
- Test tools
- Test reports
- Entry Criteria
  - the detailed checks that must be successfully completed before each test stage can be given the go-ahead to proceed.
- Exit Criteria
  - the checks that must be successfully completed before the each test stage can be considered to be complete.

27