Software Development Processes

1.19 Identify features of both staged and iterative design and development processes

Software development refers to a set of computer science activities dedicated to the process of creating, designing, deploying and supporting software.

Software itself is the set of instructions or programs that tell a computer what to do. It is independent of hardware and makes computers programmable. There are three basic types:

- **System software** to provide core functions such as operating systems, disk management, utilities, hardware management and other operational necessities.

- **Programming software** to give programmers tools such as text editors, compilers, linkers, debuggers and other tools to create code.

- **Application software** (applications or apps) to help users perform tasks. Office productivity suites, data management software, media players and security programs are examples. Applications also refers to web and mobile applications like those used to shop on Amazon.com, socialize with Facebook or post pictures to Instagram.

Software development is primarily conducted by programmers, software engineers and software developers. These roles interact and overlap, and the dynamics between them vary greatly across development departments and communities.

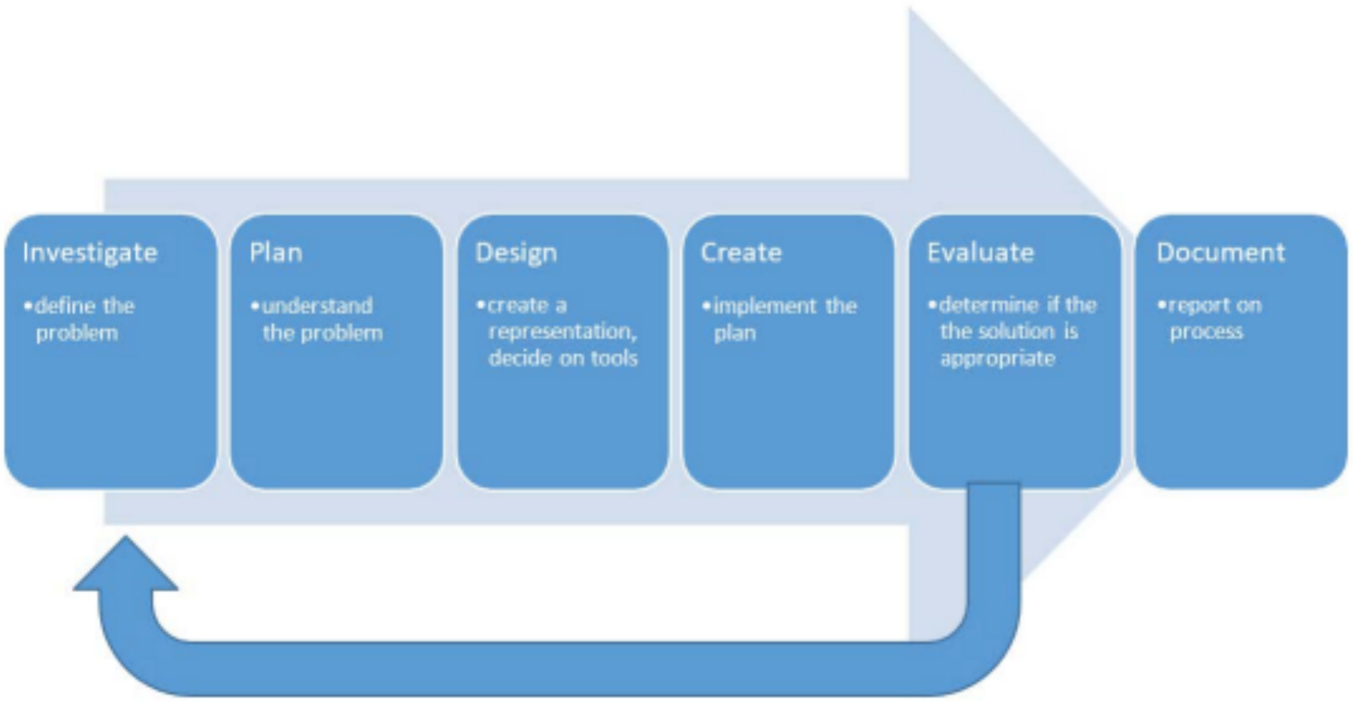
The systematic approach that is used in software engineering is sometimes called a software process. A software process is a sequence of activities that leads to the production of a software product. Four fundamental activities are common to all software processes.

1. **Software specification**, where customers and engineers define the software that is to be produced and the constraints on its operation.

2. **Software developmen**t, where the software is designed and programmed.

3. **Software validation**, where the software is checked to ensure that it is what the customer requires.

4. **Software evolution**, where the software is modified to reflect changing customer and market requirements



Software development process

Staged Design (Waterfall)

This takes the fundamental process activities of the development process and represents them as separate stages that cascade from one stage to another. This is a plan driven process as ideally you plan and schedule all of the development activities before starting the software development.

The stages of the waterfall model reflect the fundamental software development process.

1. **Requirements analysis and definition** The system's services, constraints, and goals are established by consultation with system users. They are then defined in detail and serve as a system specification.

2. **System and software design** The systems design process allocates the requirements to either hardware or software systems. It establishes an overall system architecture. Software design involves identifying and describing the fundamental software system abstractions and their relationships.

3. **Implementation and unit testin**g During this stage, the software design is realised as a set of programs or program units. Unit testing involves verifying that each unit meets its specification.

4. **Integration and system testing** The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met. After testing, the software system is delivered to the customer.

5. **Operation and maintenance** Normally, this is the longest life-cycle phase. The system is installed and put into practical use. Maintenance involves correcting errors that were not discovered in earlier stages of the life cycle, improving the implementation of system units, and enhancing the system's services as new requirements are discovered.

Each phase of the development should not start until the previous stage is complete and signed off.

Pros	Cons
Timescales are kept	Clients needs may be difficult to easily define
No financial surprises	Lack of flexibility
Testing is made easy	Longer delivery time
Deal with issues during design phase	If any issues are discovered late in the development, the whole process must start again.
What you plan you get	Client has little input in the development process

In reality, software development needs to be flexible and waterfall design model is probably not the right process where software requirements could change quickly and is more suited to rigid design processes such as hardware development.

Iterative Design (Agile)

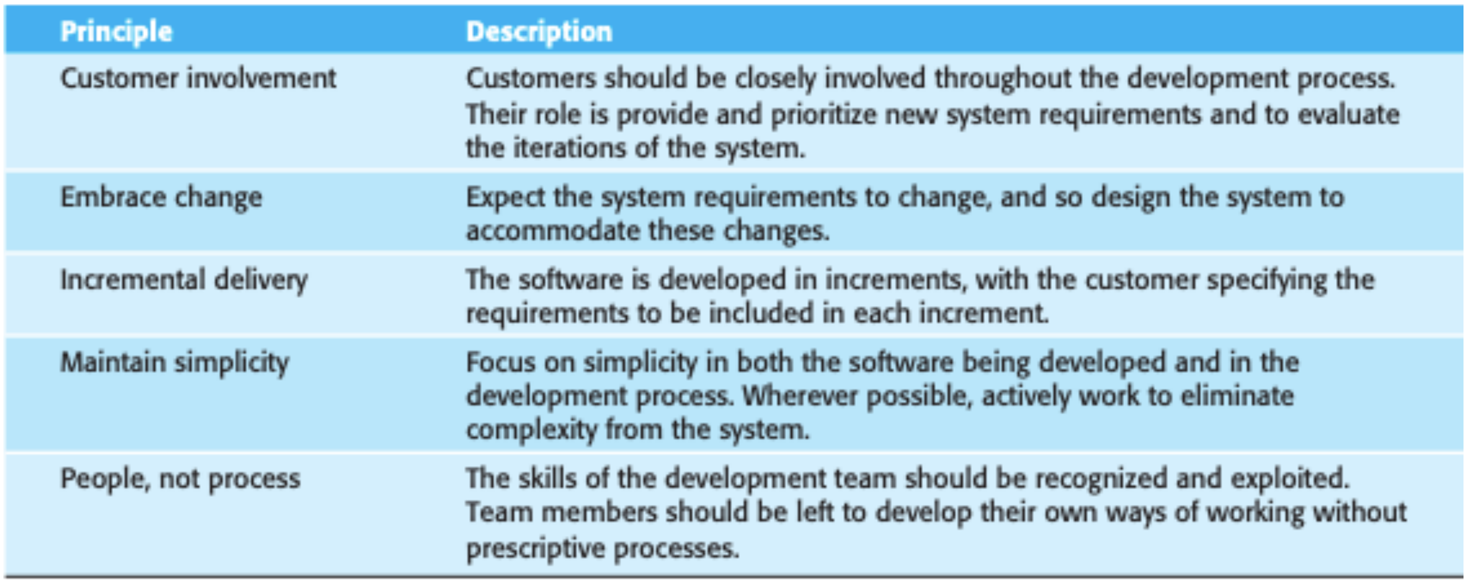
Iterative development is based on the idea of developing an initial prototype, getting feedback from users , and evolving the software through several versions until the required system has been developed.

The phases of software development are all done together rather than as seperate stages and this allows for rapid feedback across activities. At each stage, a **prototype** is built with user participation to ensure that the system is being developed in line with what the user wants. The success of the software development depends on

- Keeping the prototype simple
- Rapid feedback from the user
- Understanding that user requirements may change during development as they are forced to consider their needs in detail
- Being prepared to make changes as the model develops

Agile development is not a single iterative development process, instead it is a name for a range of different types of approaches that are iterative in nature. Agile development processes include methodologies such as Extreme Programming, SCRUM and Kanban.

The philosophy behind agile methods is reflected in the agile manifesto (http:// agilemanifesto.org) issued by the leading developers of these methods. This manifesto states:



Principle	Description
Customer involvement	Customers should be closely involved throughout the development process. Their role is provide and prioritize new system requirements and to evaluate the iterations of the system.
Embrace change	Expect the system requirements to change, and so design the system to accommodate these changes.
Incremental delivery	The software is developed in increments, with the customer specifying the requirements to be included in each increment.
Maintain simplicity	Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.
People, not process	The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.

Pros	Cons
Very flexible	Final product is not released first
Prototypes/products get to market faster	Documentation can often be lacking
Client is involved so their needs are clearly identified	Can be hard to pin down a delivery date for final product
Not as rigid as waterfall.	Hard to have a final product if customer keeps changing their mind or is unsure of what they want.
Immediate feedback on prototypes from customers	Can end up with spiralling costs if development goes on and on

Testing (Evaluation stage)

2.22 Explain the different stages in 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. When you test software, you are trying to do two things:

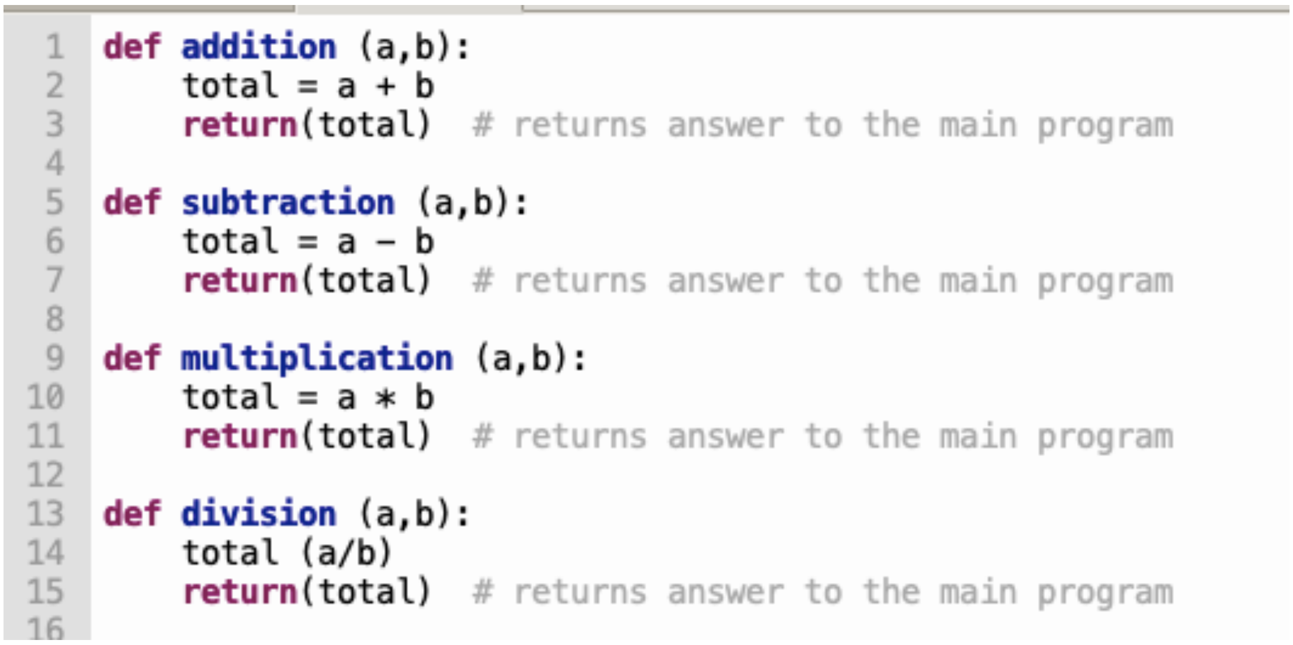
1. Demonstrate to the developer and the customer that the software meets its requirements.You may also test combinations of features to check for unwanted interactions between them.

2. Find inputs or input sequences where the output of the software is incorrect, undesirable, or does not conform to its specification. These are caused by defects (bugs) in the software. When you test software to find defects, you are trying to root out undesirable system behaviour such as system crashes, unwanted interactions with other systems, incorrect computations, and data corruption.

Unit testing

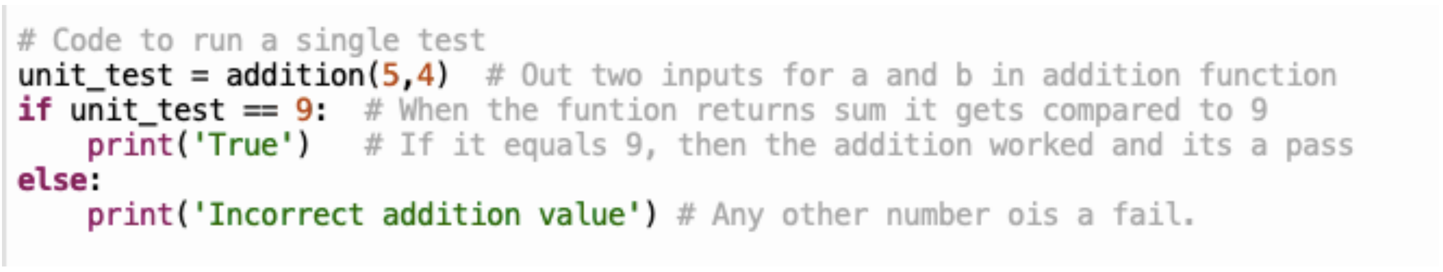
A unit test is a way of testing a unit - the smallest piece of code that can be logically isolated in a system. In most programming languages for example, that is a function, a class or method. The test will call to these units with different input parameters that should cover all the features of the unit, i.e the outputs.

For example, you may develop a program that can carry out different operations on numbers like a calculator. Each operation is broken into its own function. Each of those functions is a unit.



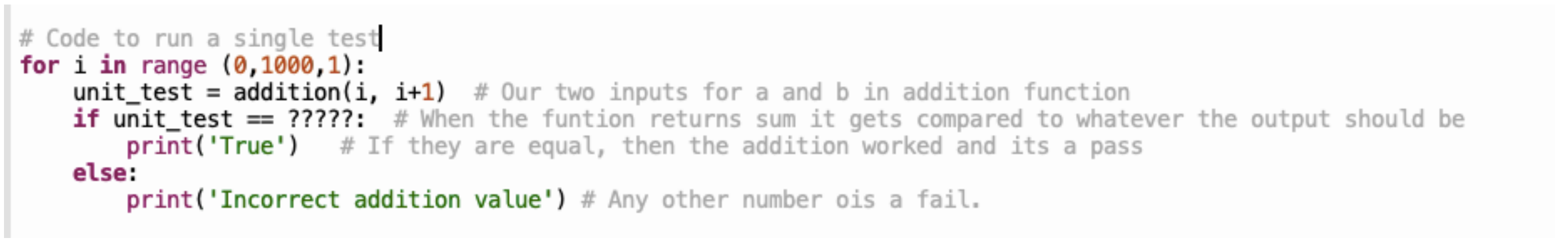
```
def addition (a,b):
    total = a + b
    return(total)  # returns answer to the main program

def subtraction (a,b):
    total = a - b
    return(total)  # returns answer to the main program

def multiplication (a,b):
    total = a * b
    return(total)  # returns answer to the main program

def division (a,b):
    total (a/b)
    return(total)  # returns answer to the main program

```

If you test each unit you must give the inputs and the expected output. So for example to unit test the addition function you could give the following example.



```
# Code to run a single test
unit_test = addition(5,4)  # Out two inputs for a and b in addition function
if unit_test == 9:  # When the funtion returns sum it gets compared to 9
    print('True')   # If it equals 9, then the addition worked and its a pass
else:
    print('Incorrect addition value') # Any other number ois a fail.
```

This is only testing one set of numbers though, so you want to automate this process to run lots of test data. You could run the unit test within a for loop similar to the image below.



```
# Code to run a single test
for i in range (0,1000,1):
    unit_test = addition(i, i+1)  # Our two inputs for a and b in addition function
    if unit_test == ?????:  # When the funtion returns sum it gets compared to whatever the output should be
        print('True')   # If they are equal, then the addition worked and its a pass
    else:
        print('Incorrect addition value') # Any other number ois a fail.
```

Benefits of unit testing:

- Find bugs early
- Testing can be automated
- Decrese the total testing cost because it can be automated
- Makes debugging easier as only newest code needs to be checked

Disadvantages

- Increase in initial development time
- Can introduce false confidence in the code

System testing

System testing during development involves integrating components to create a version of the system and then testing the full system. System testing checks that components are compatible, interact correctly, and transfer the right data at the right time across their interfaces.

1. During system testing, reusable components that have been separately developed and off-the-shelf systems may be integrated with newly developed components. The complete system is then tested.

2. Components developed by different team members or sub-teams may be integrated at this stage. System testing is a collective rather than an individual process. In some companies, system testing may involve a separate testing team with no involvement from designers and programmers.

All systems have emergent behaviour. This means that some system functionality and characteristics only become obvious when you put the components together. This interaction testing should discover those component bugs that are only revealed when a component is used by other components in the system. Automated system testing is usually more difficult than automated unit or component testing. Automated unit testing relies on predicting the outputs and then encoding these predictions in a program. The prediction is then compared with the result. However, the point of implementing a system may be to generate outputs that are large or cannot be easily predicted.

Function testing

FUNCTIONAL TESTING is a type of software testing that validates the software system against the functional requirements/specifications. The purpose of Functional tests is to test each function of the software application, by providing appropriate input, and verifying the output against the Functional requirements. These are statements of services the system should provide, how the system should react to particular inputs, and how the system should behave in particular situations. For example, what should the software do if someone enters an invalid user name or password. In some cases, the functional requirements may also explicitly state what the system should not do

Black box vs White Box testing

Software Testing can be majorly classified into two categories:

1. **Black Box Testing** is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. Only the external design and structure are tested.

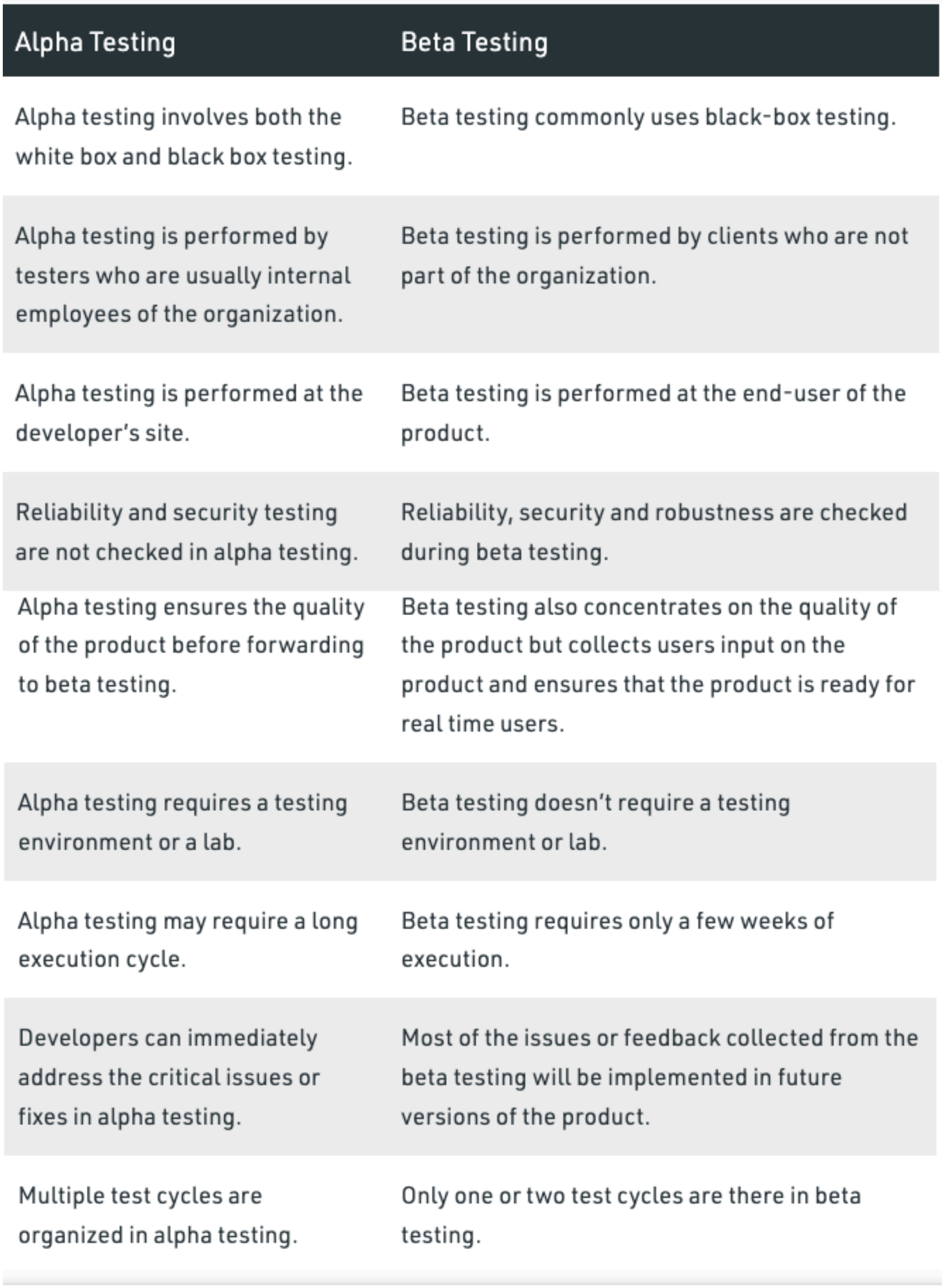
2. **White Box Testing** is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester. Implementation and impact of the code are tested.

S. No.	Black Box Testing	White Box Testing
1.	It is a way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it.	It is a way of testing the software in which the tester has knowledge about the internal structure or the code or the program of the software.
2.	Implementation of code is not needed for black box testing.	Code implementation is necessary for white box testing.
3.	It is mostly done by software testers.	It is mostly done by software developers.
4.	No knowledge of implementation is needed.	Knowledge of implementation is required.
5.	It can be referred to as outer or external software testing.	It is the inner or the internal software testing.
6.	It is a functional test of the software.	It is a structural test of the software.
7.	This testing can be initiated based on the requirement specifications document.	This type of testing of software is started after a detail design document.
8.	No knowledge of programming is required.	It is mandatory to have knowledge of programming.
9.	It is the behavior testing of the software.	It is the logic testing of the software.
10.	It is applicable to the higher levels of testing of software.	It is generally applicable to the lower levels of software testing.

Alpha vs Beta testing

Alpha Testing is a type of software testing performed to identify bugs before releasing the product to real users or to the public. Alpha Testing is one of the user acceptance tests.

Beta Testing is performed by real users of the software application in a real environment. Beta testing is one type of User Acceptance Testing.



Alpha Testing	Beta Testing
Alpha testing involves both the white box and black box testing.	Beta testing commonly uses black-box testing.
Alpha testing is performed by testers who are usually internal employees of the organization.	Beta testing is performed by clients who are not part of the organization.
Alpha testing is performed at the developer's site.	Beta testing is performed at the end-user of the product.
Reliability and security testing are not checked in alpha testing.	Reliability, security and robustness are checked during beta testing.
Alpha testing ensures the quality of the product before forwarding to beta testing.	Beta testing also concentrates on the quality of the product but collects users input on the product and ensures that the product is ready for real time users.
Alpha testing requires a testing environment or a lab.	Beta testing doesn't require a testing environment or lab.
Alpha testing may require a long execution cycle.	Beta testing requires only a few weeks of execution.
Developers can immediately address the critical issues or fixes in alpha testing.	Most of the issues or feedback collected from the beta testing will be implemented in future versions of the product.
Multiple test cycles are organized in alpha testing.	Only one or two test cycles are there in beta testing.

Test Cases

Test cases are specifications of the inputs to the test and the expected output from the system (the test results), plus a statement of what is being tested. Test data are the inputs that have been devised to test a system. Test data can sometimes be generated automatically, but automatic test case generation is impossible. People who understand what the system is supposed to do must be involved to specify the expected test results. However, test execution can be automated. The test results are automatically compared with the predicted results, so there is no need for a person to look for errors and anomalies in the test run.

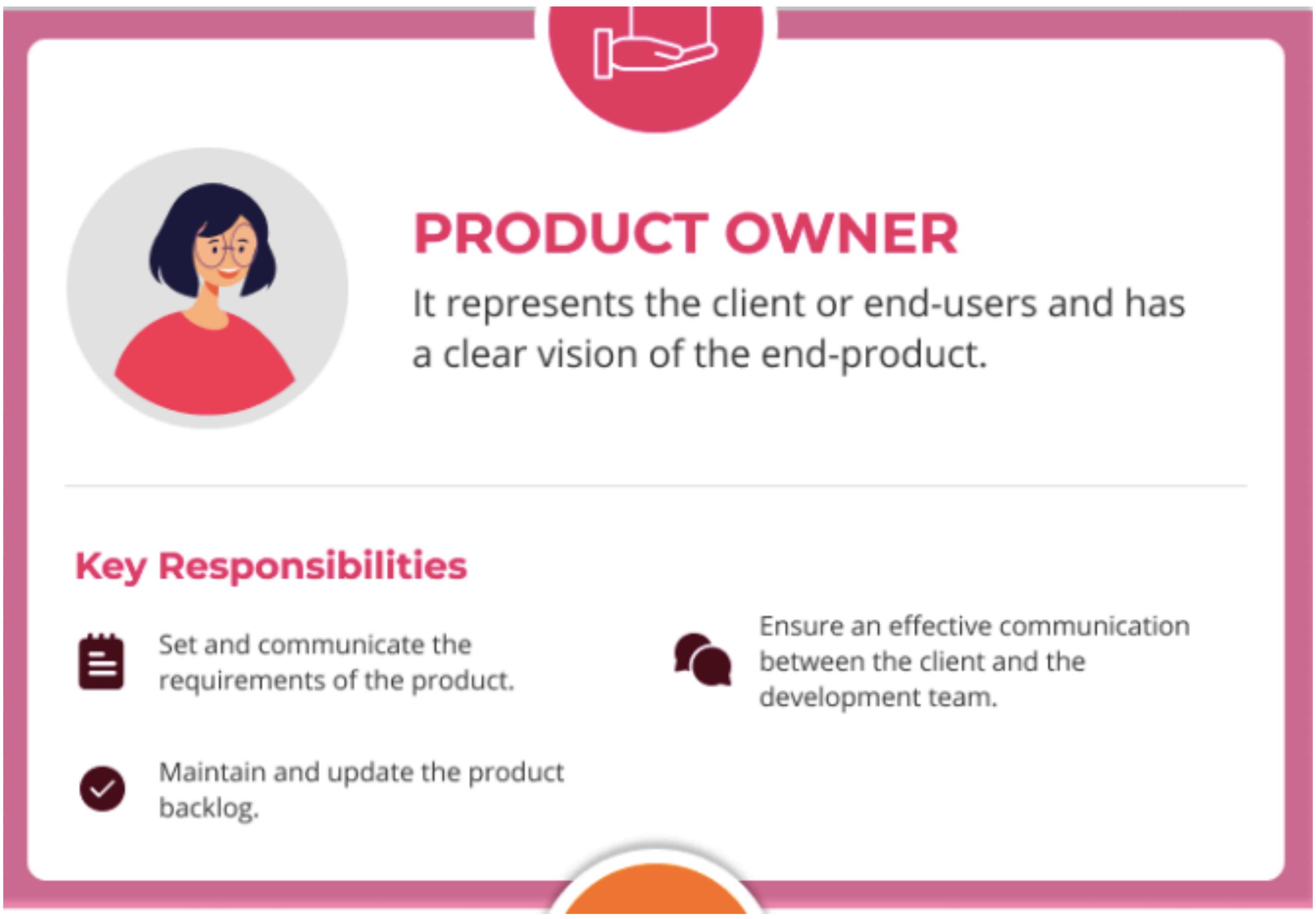
Universal design

1.15 List the principles of universal design (This is part of a larger requirement)

https://universaldesign.ie/what-is-universal-design/the-7-principles/

Roles in Software Development

1.18 Recognise the diverse roles and careers that use computing technologies



PRODUCT OWNER

It represents the client or end-users and has a clear vision of the end-product.

Key Responsibilities

- Set and communicate the requirements of the product.
- Ensure an effective communication between the client and the development team.
- Maintain and update the product backlog.

BUSINESS ANALYST

Translates business needs into requirements and ensures they are documented correctly.

Key Responsibilities

- Define the scope and prioritize it, and provide technological solutions
- Define, analyze, and manage technical and business requirements

TEAM LEAD

Acts as a mentor to help the team keep focused on the tasks, deliver work on time, and meet the project goals.

Key Responsibilities

- Guide the team towards successful project delivery.
- Prevent and solve any conflict or issue that may arise.

SOFTWARE ARCHITECT

Defines the overall architecture system and makes high-level design choices based on non-functional requirements.

Key Responsibilities

- Define the technical and functional architecture of the system.
- Develop the most critical components of the system.

DEVELOPERS

They are in charge of writing the code and developing the software products.

Key Responsibilities

- Develop the features laid out in the sprint.
- Update the status of the software project to the Project Manager.

QA TEAM

They provide support to projects to ensure the adequate practices are used during the development process.

Key Responsibilities

- Evaluate the execution of processes and production of deliverables.
- Identify and document deviations in the use of standards.

UX/UI DESIGNERS

They provide support to projects to ensure the adequate practices are used during the development process.

Key Responsibilities

- Analyze functional requirements intended for the users.
- Define the information architecture and navigation model.

TESTERS

They are responsible for ensuring that the software solution meets the requirements and complies with the quality standards.

Key Responsibilities

- Understand feature requirements.
- Create and execute test cases to detect bugs or defects.

PROJECT MANAGER

Is responsible for knowing the “who, what, where, when, and why” of the software plan.

Key Responsibilities

- Plan, schedule, budget, execute, and deliver the software project.
- Supervise the development team.