Higher

Computing Science

Software Design and Development

Development Methodologies, Analysis & Design

NAME:_____

Contents

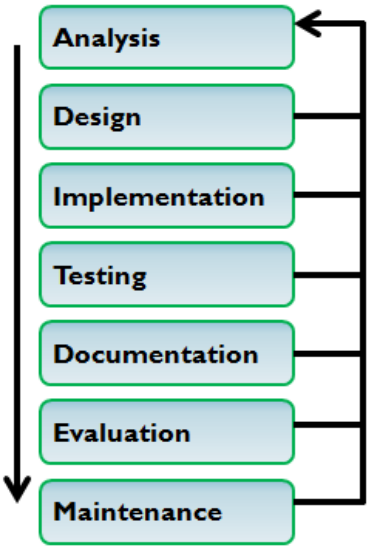
DEVELOPMENT METHODOLOGIES

ITERATIVE DEVELOPMENT PROCESS

In an iterative development process, software is developed in a sequence of stages.

Each stage takes information from the previous stage and provides information to the next.

There are seven stages to this software development process.



Software development is an **iterative process**.

At any point, it may be necessary to **revisit earlier stage**s.

This could be in order to make improvements due to new information or the presence of errors.

Key Features

- Client heavily involved in the initial analysis stage and at the end of development, but not consulted during development.
- Teams of analysts, programmers, testers and documenters work independently on each phase of development with limited communication.
- A lot of time spent at the beginning of the process creating a detailed project specification.
- Follows a strict plan, with progress measured against timescales set at the beginning of the project.
- A predictive methodology, focusing on analysing and planning the future in detail and catering for known risks.
- Testing is carried out when the full implementation phase of the project is complete.

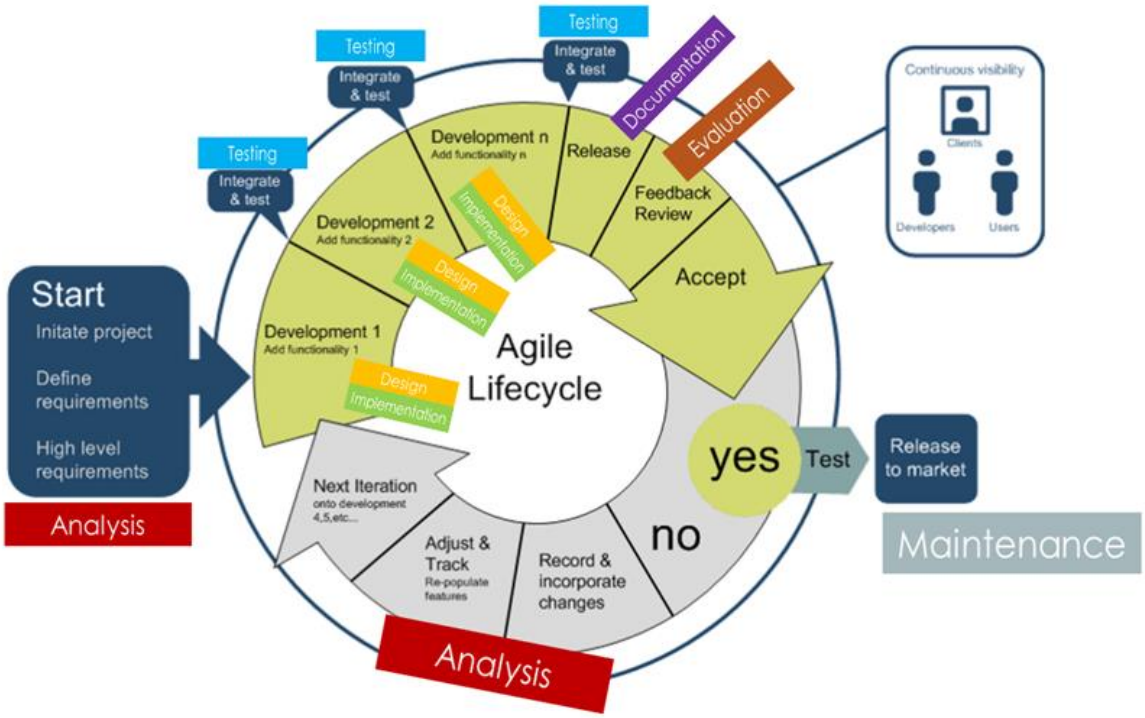
AGILE METHODOLOGIES

Agile development emphasises real time, **face-to-face communication** involving all the people necessary to finish the software.

Very little written documentation is produced.

Software is developed in short iterations, each one like a miniature software project of its own.

The purpose of a single iteration is not to produce the final completed solution, but to add additional functionality that produces working software. After each iteration the project priorities are then re-evaluated.



Iterative development attempts to produce software by assuming a perfect understanding of the client's requirements from the start. In reality however, it rarely delivers what the client wants as the client often doesn't know exactly what they want until they see it.

Agile methodologies embrace short **iterations** (Sprints) where small teams work to develop working software that builds on the previous iteration.

During each iteration, working software is produced following which the requirements for the next iteration can be evaluated.

Sprints

- A sprint is a planned delivery schedule for an aspect of the system. Within a sprint the principles of analysis, design, implementation and testing are used.
- Prototyping is also likely, particularly during the early phase of a sprint.
- Sprints are carried out for each area of development, so rather than having a rigid set of steps to follow for the development of the entire system, several steps are repeated in one sprint and then carried out again in the next sprint.

Agile Disadvantages

- The main drawback of agile methods is that following a sequence of sprints and engaging in near daily communication is very time consuming.
- The emphasis on team work and communication in a face to face manner means that long term, large scales projects are often unrealistic.
- Agile methods tend to suit small scale development better than large scale development.

Key Features

- Client is involved throughout the process, giving constant feedback on prototypes of the software during development.
- Feedback is acted upon, quickly ensuring the software evolves throughout the project.
- Teams of developers communicate and collaborate, rather than teams of experts operating in isolation.
- Agile focuses on reducing documentation. It spends time on small cycles of coding, testing and adapting to change.
- Progress is measured by the time it takes to produce prototypes or working components of the software. Focus is on delivering software as quickly as possible.
- An adaptive methodology, focusing on adapting quickly to changing realities. When the needs of a project change, an adaptive team changes as well.
- There is no recognised testing phase, as testing is carried out in conjunction with programming.

ITERATIVE VS AGILE

	Iterative (Waterfall Model)	Agile
Client Interaction	The client is heavily involved in the initial analysis stage and at the end of development, when evaluating if the software meets their needs and matches the agreed specification.	The client is involved throughout the process, giving constant feedback on prototypes of the software during development. This feedback is acted upon, quickly ensuring the software evolves throughout the project. Changing goals during the development can be positive in terms of final client satisfaction with the product.
Teamwork	Teams of analysts, programmers, testers and documenters work independently on each phase of development. Teams mainly work in isolation with some communication required between each phase.	Teams of developers communicate and collaborate, rather than teams of experts operating in isolation. During a project, fast, face-to-face communication between individuals with different skills is an important factor in progressing the project quickly.
Documentation	A detailed project specification is created at the beginning of a project. Significant time is spent during the project on design, program commentary and test plans.	While modelling solutions remains important, creating large documents that are never updated or referred to again upon completion of the project are not. Agile focuses on reducing documentation. It spends time on small cycles of coding, testing and adapting to change. Any documentation produced (for example internal commentary in code) should focus purely on progressing the project.

Measurement of progress	Follows a strict plan, with progress measured against timescales set at the beginning of the project.	Breaks a project down into a series of short development goals (often called “sprints”). This involves cross-functional teams working on: planning, analysis, design, coding, unit testing, and acceptance testing. Progress is measured by the time it takes to produce prototypes or working components of the software. Agile focuses on delivering software as quickly as possible.
Adaptive vs predictive	A predictive methodology, focusing on analysing and planning the future in detail and catering for known risks. Predictive methods rely on effective early phase analysis and if this goes very wrong, the project may have difficulty changing direction. Predictive teams often institute a change control board to ensure they consider only the most valuable changes.	An adaptive methodology, focusing on adapting quickly to changing realities. When the needs of a project change, an adaptive team changes as well. An adaptive team has difficulty describing exactly what they will do next week but could report on which features they plan for next month. The further away a date is, the vaguer an adaptive method is about what will happen on that date.
Testing	Testing is carried out when the implementation phase of the project is complete.	There is no recognised testing phase, as testing is carried out in conjunction with programming.

PRACTISE QUESTIONS

Question 1 (2019 Qu 5)

Describe the role of the client when developing software using agile methodologies. (2)

Question 2 (SQP Qu 2)

A developer and their client are based in different time zones in the world. Explain the impact that this can have when using an agile methodology compared to an iterative one. (1)

ANALYSIS STAGE

This is the start of the software development process and defines the extent of the software task. This is called the software specification. It is often the basis of a legal contract between the client (customer) and the software company writing the software.

ANALYSIS TASKS

Your analysis should include the following:

- **Purpose**: a general description of the purpose of the software.
- **Scope**: a list of the deliverables that the project will hand over to the client and/or end-user, eg design, completed program, test plan, test results and evaluation report. It can also include any time limits for the project.
- **Boundaries**: the limits that help to define what is in the project and what is not. It can also clarify any assumptions made by the software developers regarding the client's requirements.
- **Functional requirements**: the features and functions that must be delivered by the system in terms of inputs, processes and outputs.

INPUTS, PROCESSES, OUTPUTS

- **Inputs** are data items that must be entered by the user. Information we have to ask the user for. This is the data that the program will take in.
- **Processes** are the things the program will do with the data items. Calculations, formatting etc. are processes.
- **Outputs** are the data items that will be displayed by our program. This will usually be the result of what the program is supposed to do.

Example:

Purpose

The purpose of this program is to take 20 pupil names, their prelim marks and their assignment marks from a file. Calculate the percentage, and then find and display the name and percentage of the pupil with the highest percentage.

Scope

This development involves creating a modular program. The deliverables include:

- detailed design of the program structure
- test plan with completed test data table
- working program
- results of testing
- evaluation report

This development work must be completed within 4 hours.

Boundaries

The program will read the pupil data (name, prelim mark and assignment mark) for 20 pupils from a sequential file. The data is accurate, so there is no need to implement input validation.

The pupil with the top mark will be the pupil who has the highest percentage. The only output needed is the name and percentage of the pupil with the highest percentage.

Functional Requirements

These are defined in terms of the inputs, processes and outputs detailed below. All inputs are imported from a sequential file and all outputs displayed on the screen. The program is activated by double clicking on the file icon and then selecting “Run” from the menu. Each process should be a separate procedure or function that is called from the main program.

Inputs: Pupil name
Prelim mark
Assignment mark

Processes: Calculate the percentage for each pupil
Find the name and percentage of the pupil with the highest percentage

Outputs: Name of the pupil with the highest percentage
The highest percentage

PRATISE QUESTIONS

Question 1 (2019 Qu 11a)

A car manufacturer includes an event data recorder in their cars. This device begins recording when the car's sensors detect a sudden change in speed. The data captured can be analysed when required. A sample of data is shown below.

Event data	Sample
speed (mph)	58.2
accelerator (%)	42
brake (%)	0
seatbelt on	True

When triggered by a sensor this data is sampled 10 times per second for 20 seconds. Software is to be developed that can analyse the data captured from a car's event data recorder.

During the analysis stage boundaries are identified. State two boundaries for this task. (2)

DESIGN

TOP-DOWN DESIGN / STEPWISE REFINEMENT

Top-down design involves identifying an overall problem and breaking it down into smaller sub-problems (main steps).

The process of stepwise refinement is then used to break the sub-problems down until each one is small enough that they are manageable.

Top-down design emphasises planning and a complete understanding of the system.

No coding should take place until a sufficient level of detail has been reached in the design.

Each sub-problem is coded as a module however this delays testing of the functional units until significant design is complete.

DESIGN TECHNIQUES

Pseudocode

When using pseudocode to design efficient solutions to a problem, you must include the following:

- **Top level design** — the major steps of the design. In the example below, numbered from 1 to 4.
- **Data flow** — shows the information that must flow In or Out from the sub-programs. In the example below, written to the right of the top level design.
- **Refinements** — break down the design from the top level when required. In the example below, numbered as a sub-number of the top level.

Example:

The following design is for a program that will read the name, prelim mark and coursework mark for a class of 20 pupils from a file. It will calculate a percentage from each of their prelim marks and coursework marks added together. It will then display the name of the pupil with the highest percentage and their percentage.

Main Steps	**Data Flow**
1 Get results	(**OUT**: pupil name(), prelim mark(), course mark())
2 Calculate percentages	**(IN**: prelim mark(), course mark() **OUT**: percentage())
3 Find position of pupil with top mark	(**IN**: percentage() **OUT**: top position)
4 Display pupil with top mark	(**IN**: pupil name(), top position)

Refinements

1.1 Open marks file
1.2 Start fixed loop for each pupil
1.3 Get pupil name()
1.4 Get prelim mark()
1.5 Get course mark()
1.6 End fixed loop
1.7 Close marks file

2.1 Start fixed loop for each pupil
2.2 percentage() equals (prelim mark() + course mark()) divided by 1.5
2.3 End fixed loop

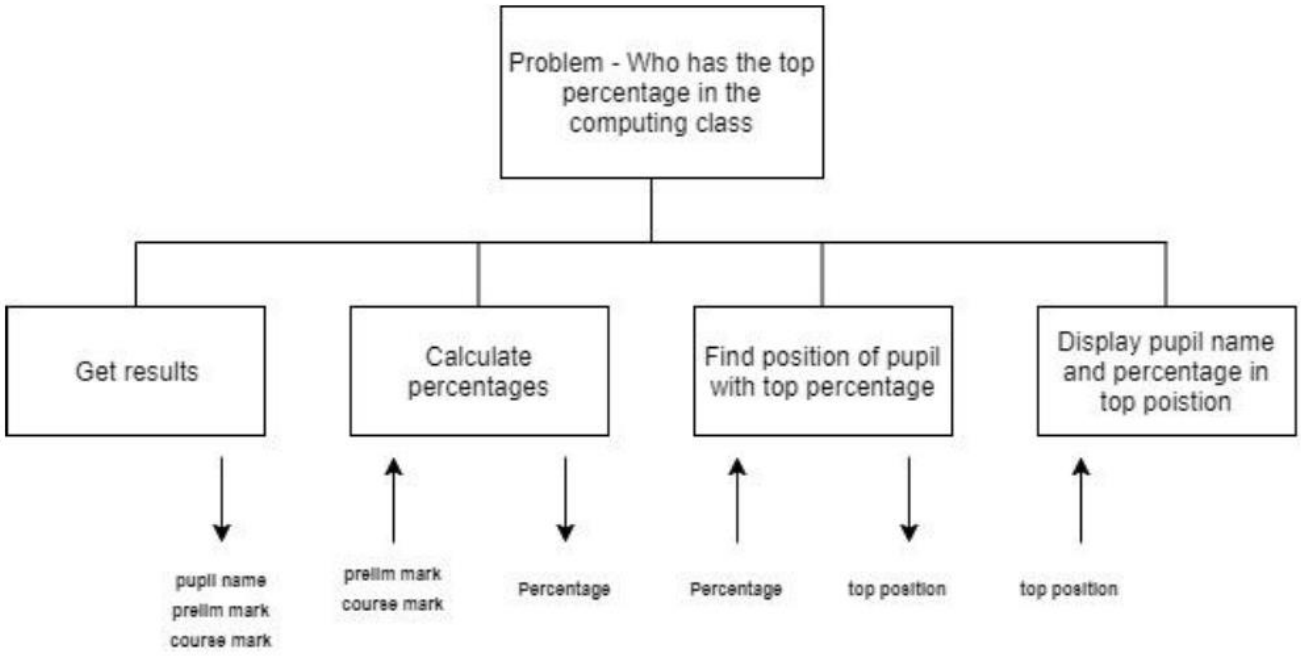
3.1 top position equals first position
3.2 Start fixed loop from second pupil
3.3 If percentage() is greater than current top percentage Then
3.4 set position as new top position
3.5 End If
3.6 End fixed loop

4.1 Display “Top pupil is”, pupil name(top position), “with”, percentage(top position), “percent”

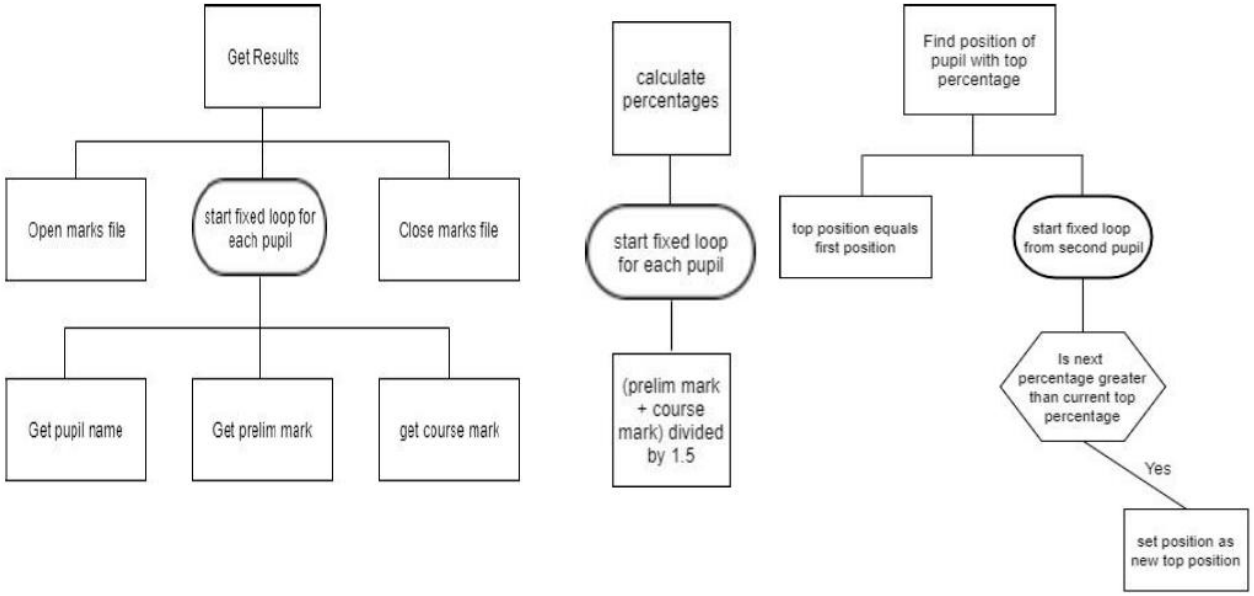
Structure Diagrams

The following structure diagram solves the same problem as the pseudocode:

- **Top level design** — the major steps of the design.
- **Data flow** — shows the information that must flow In or Out from the sub-programs. In the example below, written underneath the top level design with an arrow showing whether they are in or out.



- **Refinements** — break down the design from the top level into smaller steps. They can be shown separately from the top level design or below the top level design.



WIREFRAME (USER INTERFACE DESIGN)

The design of the user interface (the visual layout that allows the user to interact with the programming code) can be represented using a **wireframe** diagram.

A wireframe diagram is a visual representation of how the user interface will look and it will show the position of different elements such as text, graphics, navigation etc. It is also used as a visual representation to demonstrate the input and output of a program.

A wireframe diagram can be a detailed sketch or detailed image as shown below.

The wireframe diagram should clearly show the program input and output.