OCR Computer Science A Level

1.2.3 Software Development

Concise Notes

Specification:

1.2.3 a)

- **Programming methodologies**
  - Waterfall lifecycle
  - Agile methodologies
  - Extreme programming
  - Spiral model
  - Rapid application development

1.2.3 b)

- **Merits, drawbacks and uses of programming methodologies**

1.2.3 c)

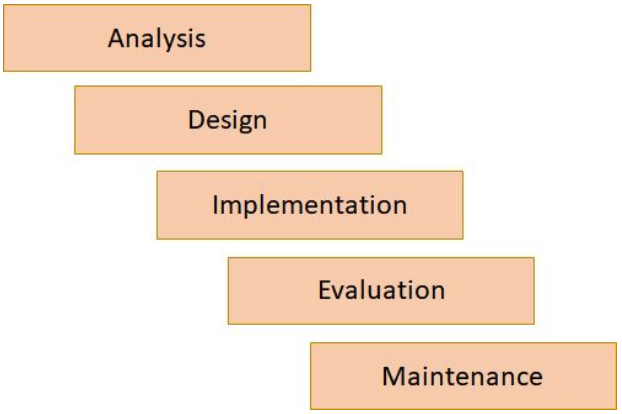
- **Writing and following algorithms**

Programming Methodologies

- Software development life cycles (SDLCs) all consist of the following stages:
    - Analysis
    - Design
    - Development
    - Testing
        - Alpha testing
        - Beta testing
        - White box testing
        - Black box testing
    - Implementation
    - Evaluation
    - Maintenance

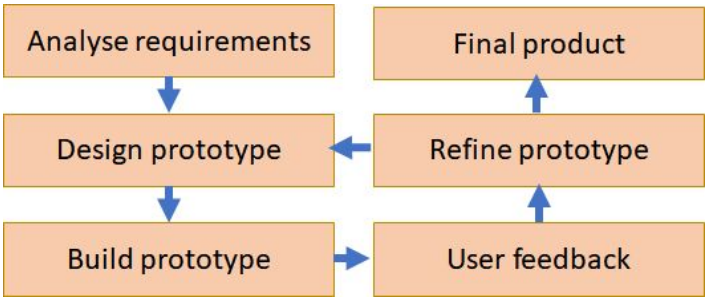
Waterfall lifecycle

- Stages are completed in sequence, from start to finish
- Clear structure makes this a model that is easy to follow
- To make a change, programmers must revisit all stages in between
- Low user involvement



Agile methodologies

- Collection of methodologies which aim to improve the flexibility of SDLCs
- Adapt quickly to changes in user requirements.
- Different sections of the program are developed in parallel so can be at different stages of development simultaneously



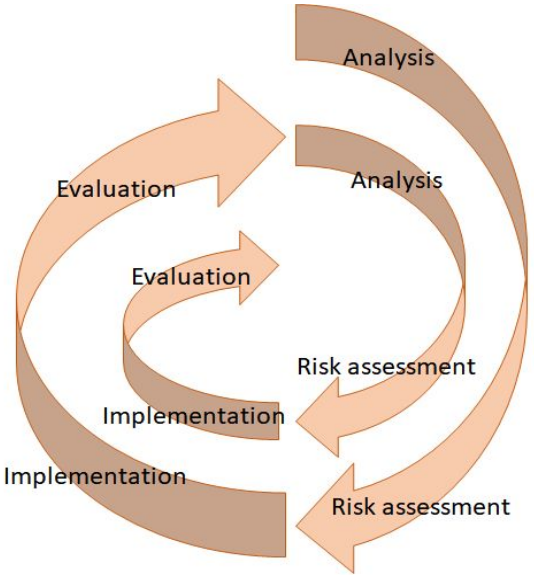
- Working prototype is delivered early on and improved in an iterative manner
- Less of a focus on documentation
- User satisfaction is prioritised

Extreme programming

- Example of an agile model
- Development team is a pair of programmers and a representative end-user
- 'User stories' are used to determine system requirements
- Produces high-quality code and highly-usable software
- Programmers work no longer than forty hours a week
- Hard to produce high quality documentation

Spiral model

- Used to manage risk-heavy projects
- Has four key stages:
  - Analysing system requirements
  - Pinpointing and mitigating risks
  - Development, testing and implementation
  - Evaluating to inform the next iteration
- Project terminated if too risky
- Specialist risk-assessors must be hired which is expensive



Rapid application development

- Iterative methodology which uses partially functioning prototypes
- User requirements are gathered using focus group
- 'Incomplete' version of the solution is given to the user to trial
- User feedback is used to generate next, improved prototype
- Final prototype matches user requirements fully
- Used where user requirements are incomplete or unclear at the start
- Code may be inefficient

Writing and following algorithms

- Algorithm = A set of instructions used to solve a problem
- All good algorithms have certain key qualities:
  - Inputs must be clearly defined - what is valid and what is invalid?
  - Must always produce a valid output for any defined input
  - Must be able to deal with invalid inputs
  - Must always reach a stopping condition
  - Must be well-documented for reference
  - Must be well-commented so modifications can easily be made

Merits, drawbacks and uses of programming methodologies

	Merits	Drawbacks	Uses
Waterfall	- Straightforward to manage - Clearly documented	- Lack of flexibility - No risk analysis - Limited user involvement	Static, low-risk projects which need little user input, such as a piece of general-purpose software
Agile	- Produces high quality code - Flexible to changing requirements - Regular user input	- Poor documentation - Requires consistent interaction between user and programmer	Small to medium projects with unclear initial requirements.
Extreme Programming	- Produces high quality code - Constant user involvement means high usability	- High cost of two people working on one project - Teamwork is essential - End-user may not be able to be present	Small to medium projects with unclear initial requirements requiring excellent usability.
Spiral	- Thorough risk-analysis and mitigation - Caters to changing user needs - Produces prototypes throughout	- Expensive to hire risk assessors - Lack of focus on code efficiency - High costs due to constant prototyping	Large, risk-intensive projects with a high budget.
Rapid Application Development	- Caters to changing user requirements - Highly usable finished product - Focus on core features, reducing development time	- Poorer quality documentation - Fast pace and late changes may reduce code quality	Small to medium, low-budget projects with short time-frames.