Section 12.1 - Program Development Lifecycle

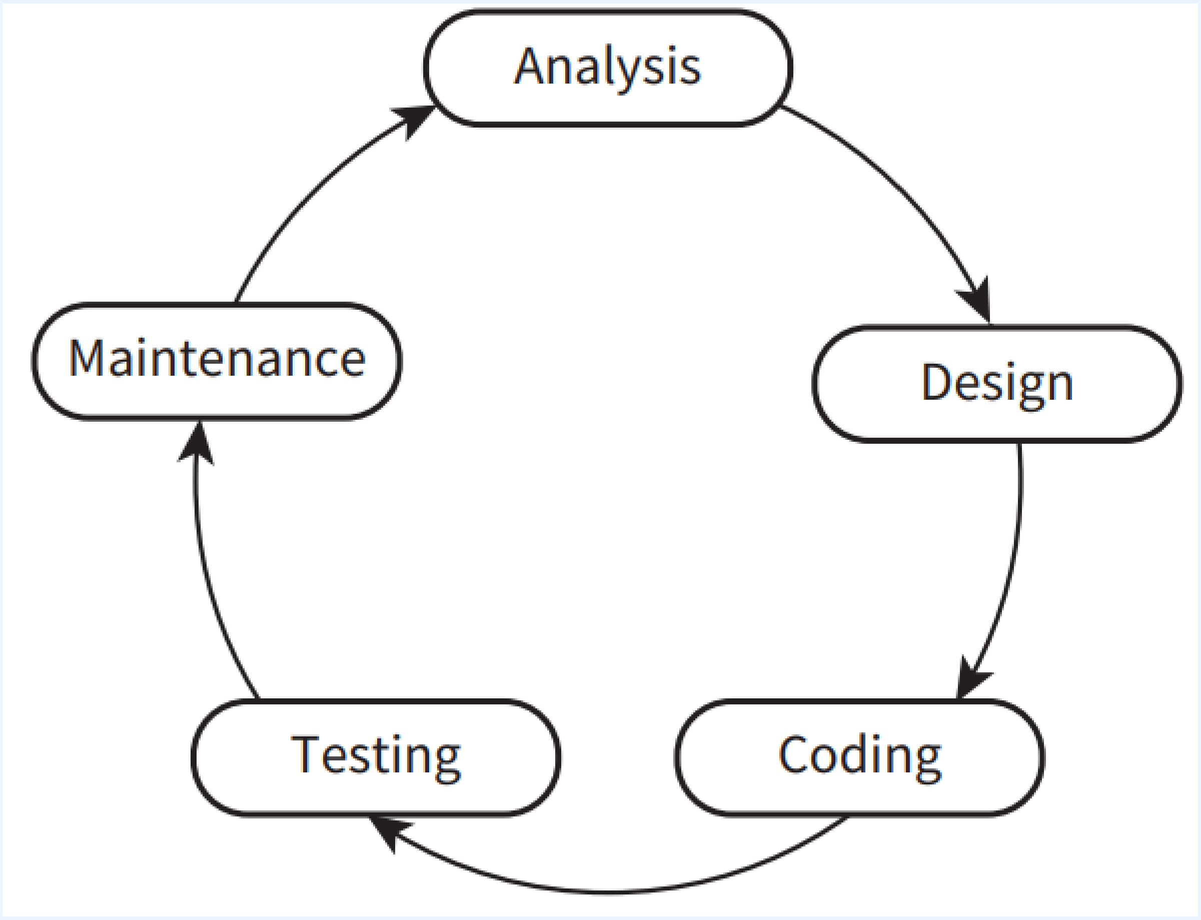
Layer 7: Application

Syllabus Content Section 12: Software Development

S12.1.1 Show understanding of the purpose of a development life cycle

---

When large software systems are required to solve big problems, these stages are more formal, especially when more people are involved in the development. Before a solution can be designed, the problem needs to be analysed. When the program works and is being used, issues might arise that require changes. This is known as maintenance.



S12.1.2 Show understanding of the need for different development life cycles depending on the program being developed

- Including, waterfall, iterative, rapid application development (RAD)

> Waterfall

Each stage must finish before moving onto the next stage

`Requirement Gathering and analysis` – All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

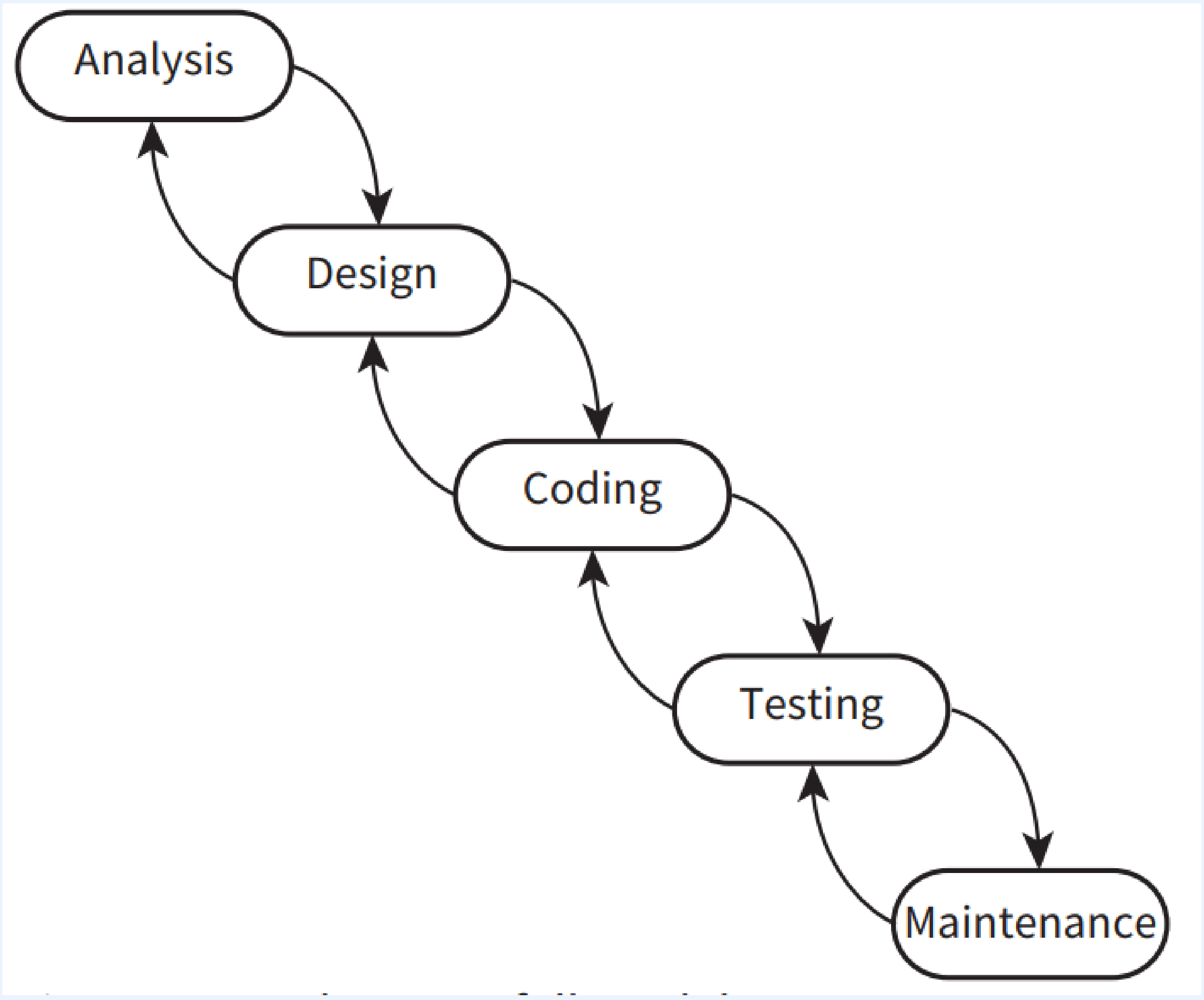
`System Design` – The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

`Implementation` – With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

`Integration and Testing` – All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

`Deployment of system` – Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

`Maintenance` – There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.



> interative

This is when you need to make a really detailed plan at the start, so you know what you want.

Then you begin to make your software.
Then you take that part and make it better
Then now you take the better part and make it even better
And you keep improving it, over and over and over again

> rapid application development(RAD)

RAD is a software development methodology that uses minimal planning. Instead it uses prototyping. A prototype is a working model of part of the solution.
In the RAD model, the modules are developed in parallel as prototypes and are integrated to make the complete product for faster product delivery. There is no detailed preplanning. Changes are made during the development process. The analysis, design, code and test phases are incorporated into a series of short, iterative development cycles.

S12.1.3 Describe the principles, benefits and drawbacks of each type of life cycle

---

> waterfall

- Advantages
  - Simple and easy to understand and use
  - Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
  - Phases are processed and completed one at a time.
  - Works well for smaller projects where requirements are very well understood.
  - Clearly defined stages.
  - Well understood milestones.
  - Easy to arrange tasks.
  - Process and results are well documented.
- Disadvantages
  - No working software is produced until late during the life cycle.
  - High amounts of risk and uncertainty.
  - Not a good model for complex and object-oriented projects.
  - Poor model for long and ongoing projects.
  - Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
  - It is difficult to measure progress within stages.
  - Cannot accommodate changing requirements.
  - Adjusting scope during the life cycle can end a project.
  - Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

> iterative

- Advantages
  - Some working functionality can be developed quickly and early in the life cycle.

    - Results are obtained early and periodically.
    - Parallel development can be planned.
    - Progress can be measured.
    - Less costly to change the scope/requirements.
    - Testing and debugging during smaller iteration is easy.
    - Risks are identified and resolved during iteration; and each iteration is an easily managed milestone.
    - Easier to manage risk - High risk part is done first.
    - With every increment, operational product is delivered.
    - Issues, challenges and risks identified from each increment can be utilized/applied to the next increment.
    - Risk analysis is better.
    - It supports changing requirements.
    - Initial Operating time is less.
    - Better suited for large and mission-critical projects.
    - During the life cycle, software is produced early which facilitates customer evaluation and feedback.
- Disadvantages
    - More resources may be required.
    - Although cost of change is lesser, but it is not very suitable for changing requirements.
    - More management attention is required.
    - System architecture or design issues may arise because not all requirements are gathered in the beginning of the entire life cycle.
    - Defining increments may require definition of the complete system.
    - Not suitable for smaller projects.
    - Management complexity is more.
    - End of project may not be known which is a risk.
    - Highly skilled resources are required for risk analysis.
    - Projects progress is highly dependent upon the risk analysis phase.

> rapid application development (RAD)

- Adventage
    - Changing requirements can be accommodated.
    - Progress can be measured.
    - Iteration time can be short with use of powerful RAD tools.
    - Productivity with fewer people in a short time.
    - Reduced development time.
    - Increases reusability of components.
    - Quick initial reviews occur.
    - Encourages customer feedback.
    - Integration from very beginning solves a lot of integration issues.
- Disadventage
    - Dependency on technically strong team members for identifying business requirements.
    - Only system that can be modularized can be built using RAD.
    - Requires highly skilled developers/designers.

- High dependency on Modelling skills.
- Inapplicable to cheaper projects as cost of Modelling and automated code generation is very high.
- Management complexity is more.
- Suitable for systems that are component based and scalable.
- Requires user involvement throughout the life cycle.
- Suitable for project requiring shorter development times.

S12.1.4 Show understanding of the analysis, design, coding, testing and maintenance stages in the program development life cycle

1. Analysis - Find out what the current method / current system is
2. Design - Design your new system on paper (flowcharts, drawings etc....)
3. Development - Make your new system
4. Testing - Check to see if your system works
5. Implementation - Give your system to the person who wanted it
6. Documentation - Make a user guide for your system
7. Evaluation - Check to see if your system does what it should do