BACHELOR OF COMPUTER APPLICATIONS

SEMESTER 5

DCA3103

SOFTWARE ENGINEERING

Unit 2

Software Design Process

Table of Contents

SL No	Topic	Fig No / Table / Graph	SAQ / Activity	Page No
1	Introduction			3
	1.1 Learning Objectives			
2	Software Design			4-5
	2.1 The Objectives of Software Design			
3	Software Development Life Cycle (SDLC)	1		5-7
4	Software Development Models			8-16
	4.1 Waterfall Model	2		
	4.2 Iterative Model:	3		
	4.3 Incremental Model	4		
	4.4 Spiral Model	5		
	4.5 Concurrent Development Model	6		
5	Summary			16-17
6	Self-Assessment Questions		1	18
7	Self-Assessment Answers			19
8	Terminal Questions			19
9	Terminal Answers			20

1. INTRODUCTION

In the process of designing software, the architecture and design of software systems are planned, conceptualised, and created. It is the procedure of outlining the needs for the programme, investigating the nature of the issue, and coming up with a solution that satisfies the needs. The foundation of software development is software design, which also serves as the starting point for the complete software lifecycle. Gathering requirements, analysis and modelling, architectural design, detailed design, and implementation are a few of the stages that commonly make up the software design process. The output of one stage often feeds into the output of the next, making each stage essential. Iterative means that designs are created and improved upon until the intended outcome is reached, and this is how the software design process is normally conducted.

1.1 Learning Objectives:

At the end of the topic, students will be able to:

- *Recall the fundamental principles of software design.*
- *Explain the importance of software design in the software development lifecycle.*
- *Apply the principles of software design to solve real-world software problems.*
- *Analyse the strengths and weaknesses of different software design approaches.*
- *Compare and contrast different design patterns for a given software problem.*

2. SOFTWARE DESIGN

The process of defining a software system's architecture, components, modules, interfaces, and data to meet predetermined requirements is known as software design. A software system that has been well designed will be modular, maintainable, and meet the required quality standards for performance, reliability, and scalability.

2.1 The Objectives of Software Design:

The objectives of Software Design are:

- It involves identifying the software requirements, researching them, and then developing a solution to satisfy them.
- Capturing the core features of the software solution in a high-level abstraction that can be further developed and put into practice.
- The design should ensure that the software system meets the functional requirements specified in the software requirements specification and also non-functional requirements like performance, scalability, reliability, maintainability, and security.
- The design should promote modular design principles to allow for easy modification and reuse of components in future software projects.
- The design intended to simplify altering, updating, and fixing bugs while also allowing the software system's development and maintenance.
- The design should make it possible to test the software system effectively to make sure it meets the criteria.
- The design should be able to adapt to changing the requirements whenever it is required.
- The design should maximise the use of resources like time, money, and staff.

2.2 Importance of Software Design

Software design is an important aspect of software development and is essential to a software project's success. It involves turning user needs into a precise design that can be utilised to create the programme. A well-designed software system may be simpler to create, test, and maintain, as well as more flexible to meet shifting needs. Improved software performance, higher quality, and shorter development cycles and costs can all result from good software design.

Designing software is crucial for several reasons.

1. Firstly, it makes sure the software satisfies the user's wants and demands. The development team can more effectively understand the needs of the system and produce a software solution that satisfies those needs by establishing a thorough design.
2. Second, software design contributes to raising the level of software quality. Because a well-designed software system is simpler to test and debug, the finished result may contain fewer mistakes and flaws.
3. The final benefit of software design is that it speeds up development and lowers costs. Developers may more easily maintain and expand the software over time by creating a system that is modular, adaptable, and scalable, which lowers development costs and accelerates time-to-market.

3. SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

Software Development Life Cycle (SDLC) is a process followed by software engineering teams to design, develop, test, deploy, and maintain software systems. It is a systematic approach to software development that provides a framework for the development team to create high-quality software in a consistent and repeatable manner that consists of several phases, and each phase has its specific activities and deliverables.

The seven phases of SDLC include:

1. Planning Phase
2. Requirement Gathering and Analysis Phase
3. Design Phase
4. Development Phase
5. Testing Phase
6. Implementation Phase, and
7. Maintenance Phase

Fig 1: The Seven Phases of DSLC

1. **Planning Phase:**

Project planning is the first phase of the SDLC, where the project goals, objectives, scope, and constraints are defined. In this phase, the team determines the needs of the new product and estimates the cost. This is the first stage of the SDLC, which is all about "What do we want?".

2. **Requirement Gathering and Analysis Phase:**

The software requirements are identified, examined, and documented during the requirements-collecting process. The development team gathers customer requirements using various techniques, such as surveys and interviews. The team then records the necessary functional and non-functional information.

3. **Design Phase:**

In this phase, the architecture, modules, interfaces, and data models of the software system are all designed. A high-level and low-level design specification is created by the team based on the needs discovered in the previous stage.

4. **Implementation Phase:**

The programme is created, coded, and tested during the implementation phase. The development team uses a variety of testing methodologies to make that the programme is operating as intended, and the code adheres to the design standards.

5. **Testing Phase:**

The software is tested in the testing phase about the requirements found during the requirements gathering phase. To make sure the programme satisfies user needs, the development team use a variety of testing approaches, including unit testing, integration testing, system testing, and acceptance testing.

6. **Deployment Phase:**

The deployment process involves setting up the system, installing the programme on the target environment, and distributing it to the users. The development team makes sure the software is properly installed and operating.

7. **Maintenance Phase:**

The development team keeps track of the software's performance during the maintenance phase, searches for flaws, and resolves them. The team also makes software changes to meet changing customer demands and needs.

4. SOFTWARE DEVELOPMENT MODELS

Software development teams use frameworks called software development models to direct the process of creating new software. These models offer an organised method for developing software, which makes it simpler to plan and calculate project costs and timetables. They also support ensuring that the finished software solution satisfies the needs and expectations of the client.

The Software Development Models include the Waterfall model, Iteration model, Incremental model, Spiral model and Concurrent development model.

4.1 Waterfall Model:

The waterfall model is a common life cycle model. It is also known as the linear-sequential life cycle model. This life cycle is simple and easy to understand.

Each phase in this model commences only if the previous phase has been entirely completed. This life cycle model is used where the requirements are known. If the software development tool is well known, then the waterfall life cycle model is used. Figure 2.1 shows a waterfall life cycle model.

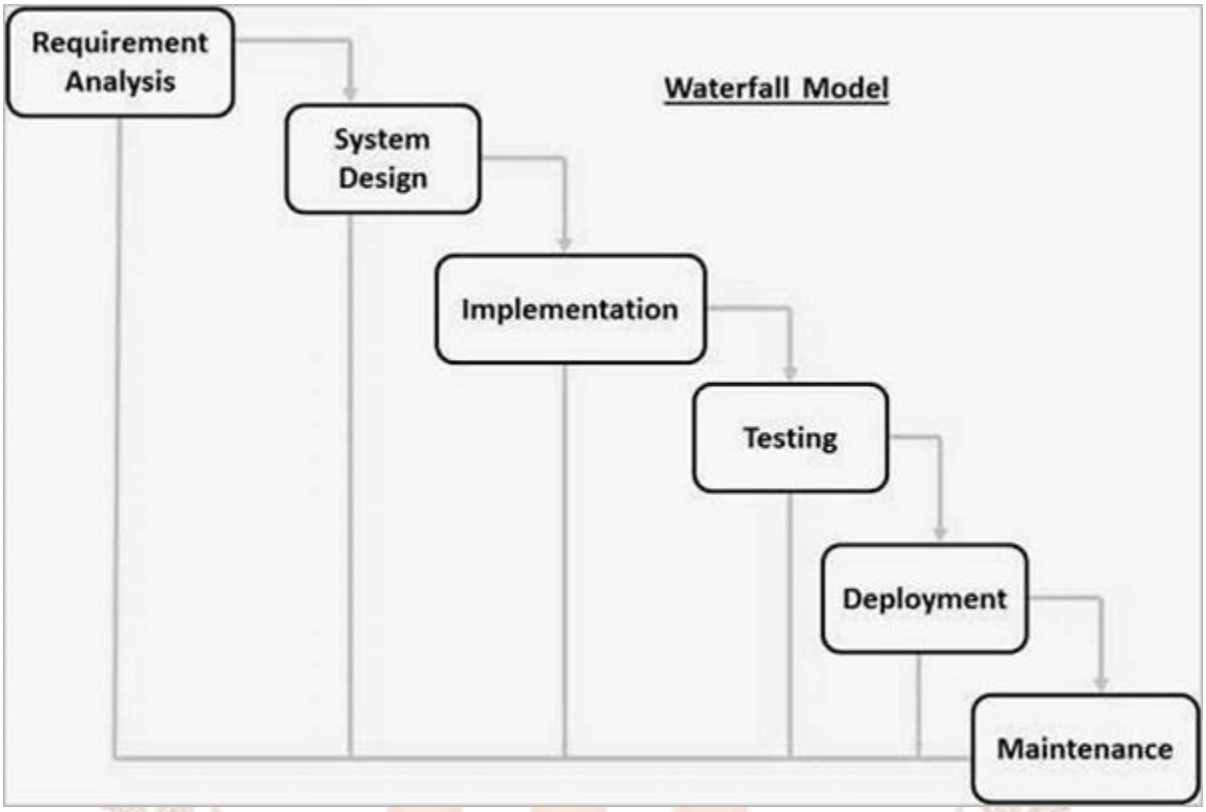

Fig 2.1: Waterfall Life Cycle Model

As shown in Figure 2.1, the project begins with requirement analysis planning.

- The project planning is done in this phase.
- After completing the requirement analysis, the design of the project begins.
- Once the design is completed, coding of the project begins. After completing the coding, the code is integrated and testing is done.
- The system is installed after the testing is completed. The final phase of the life cycle is the operation and maintenance of the system.

The advantages and disadvantages of the waterfall life cycle model are:

Advantages:

- This model is very simple and easy to use.
- Easy to manage because each phase has specific deliverables and a review process.
- Suitable for small projects where requirements are known clearly.
- The phases are processed and completed one at a time.

Disadvantages:

- The scope is adjusted during the life cycle which can adversely impact the project.
- The risk in this model is very high.
- This model is not suitable for long-term projects.
- The software is developed at the later stage of the life cycle.

4.2 Iterative Model:

In the Iterative Model, the software development process is broken down into small cycles or iterations, with each iteration consisting of the requirements gathering, design, implementation, and testing phases.

After the completion of each iteration, feedback is gathered from stakeholders and incorporated into the next iteration. This process continues until the final software product is completed as shown in Figure 2.2. Here, the deliverables from each iteration serve as feedback for the next iteration, ensuring that the software product is continuously improved and refined throughout the development process.

The Iterative Model is a flexible method of software development that permits modifications and alterations at any point in the process. It works well for projects with complicated needs that are constantly changing or ambiguous.

The Iterative Model also encourages stakeholder engagement and collaboration, leading to a more customer-centric approach to software development.

The Iterative Model is commonly used in Agile techniques like Scrum and Kanban.

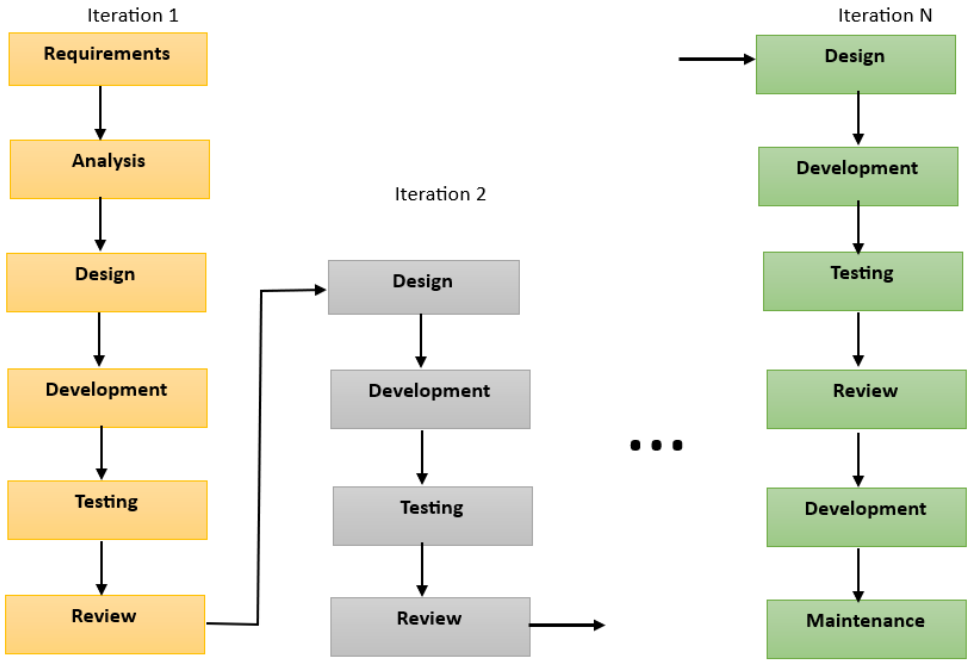

Fig 2.2: Iterative Model of Software Development

The advantages and disadvantages of the Iterative Model:

Advantages:

- Iterative model provides ***flexibility*** in the development process as it can modify and improve requirements and solutions over the course of the development cycle
- Using an iterative approach, working software can be delivered in small portions as an ***incremental delivery***, which can be helpful for stakeholders who need to see development progress as it happens.
- The iterative approach promotes ***stakeholder engagement*** and collaboration as the feedback is gathered and incorporated into each iteration.
- The iterative approach can help ***reduce risk*** in software development by identifying and addressing issues early on in the process.

Disadvantages:

- Managing an iterative approach can be more difficult as it requires more team coordination and communication than other software development models.
- The iterative approach may require more time and resources than other models due to the need for multiple iterations and feedback cycles.

- Since requirements and solutions may be modified and adjusted throughout the development process, the iterative approach is prone to scope creep, which could result in alterations to the project's scope.

4.3 Incremental Model

- The incremental model has the same phases that are in the waterfall model. But it is iterative in nature. The incremental model has various phases like analysis, design, coding and testing.
- The incremental model delivers a series of releases to the customer. These releases are called increments. More and more functionality is associated with each increment.
- The first increment is called the core product. In this release, the basic requirements are implemented and then in subsequent increments, new requirements are added. The word-processing software package can be considered as an example of an incremental model. In the first increment, only the document processing facilities are available. In the second increment, more sophisticated document-producing and processing facilities, and file management functionalities are given. In the next increment spelling and grammar checking facilities can be given. Thus, in the incremental model progressive functionalities are obtained with each release. Figure 2.3 shows an incremental life cycle model.

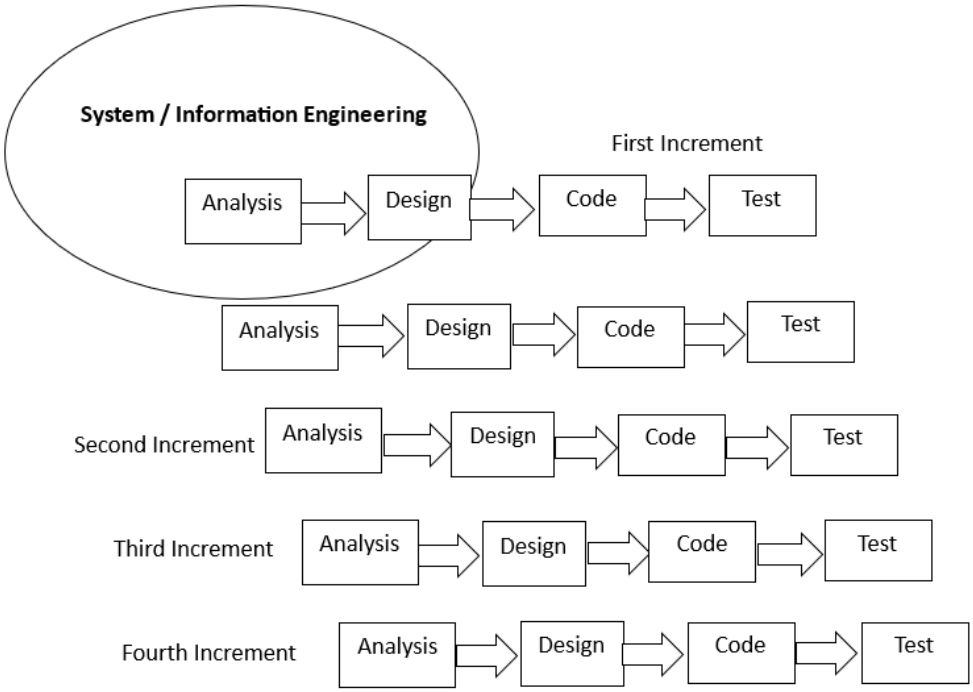

Fig 2.3: Incremental Life Cycle Model

The advantages and disadvantages of the Incremental life Cycle Model:

Advantages:

- The incremental model can be adopted when there is a smaller number of people involved in the project.
- With each increment technical risks can be managed.
- For a very small-time span, at least the core product can be delivered to the customer.
- Early feedback is generated, because implementation occurs rapidly for a small subset of the software.

Disadvantages:

- At the management and technical level planning is required.
- It becomes invalid when clients do not accept phased deliverables.

4.4 Spiral Model

The spiral model applies to projects where new technologies are used. The spiral model mainly focuses on risk analysis. The spiral life cycle model has four phases. They are - planning, risk analysis, engineering and evaluation. Software projects repeatedly pass through all these phases in iterations. The project life cycle moves in a spiral in this model. The baseline spiral starts in the planning phase and ends in the evaluation phase. Figure 2.4 shows a spiral life cycle model.

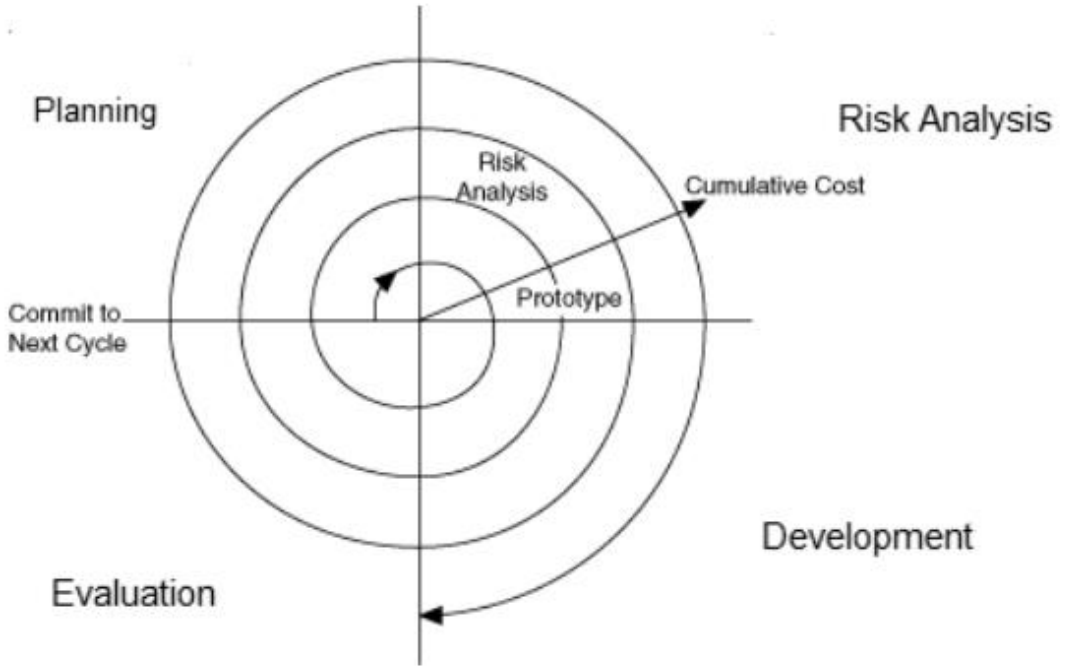

Fig 2.4: Spiral Life Cycle Model

As shown in Figure 2.4, the software project passes through all the phases in the life cycle. Each phase has some specific activity to perform.

- The requirements for the project are gathered in the planning phase.
- The risks and alternative solutions are identified in the risk analysis phase.
- At the end of the risk analysis phase, a prototype is produced.
- In the engineering phase, the software is developed and the testing is done at the end of this phase.
- In the evaluation phase the customer evaluates the project before it continues to the next spiral.

The advantages and disadvantages of the spiral model life cycle are:

Advantages:

- Risk analysis is too high in this life cycle model.
- Used for large and critical projects.
- Software is developed in the early stage of the life cycle.

Disadvantages:

- This model is costly to use.
- Success of the project depends on the risk analysis phase.
- Not suitable for small projects

4.5 Concurrent Development Model:

The concurrent development model is sometimes called concurrent engineering. using a concurrent development model, a software team can represent the concurrent and iterative components of any of the process models.

For example, the modelling activity for the spiral model is accomplished by invoking three software engineering techniques like prototyping, analysis, and design.

One software engineering activity within a modelling activity is shown schematically in Figure 2.5 using a concurrent modelling approach. Modelling may be done in any of the states mentioned at any given time. Similarly, the other tasks, actions, or activities can be represented in an analogous. However, they all take place in different states at the same time in software engineering.

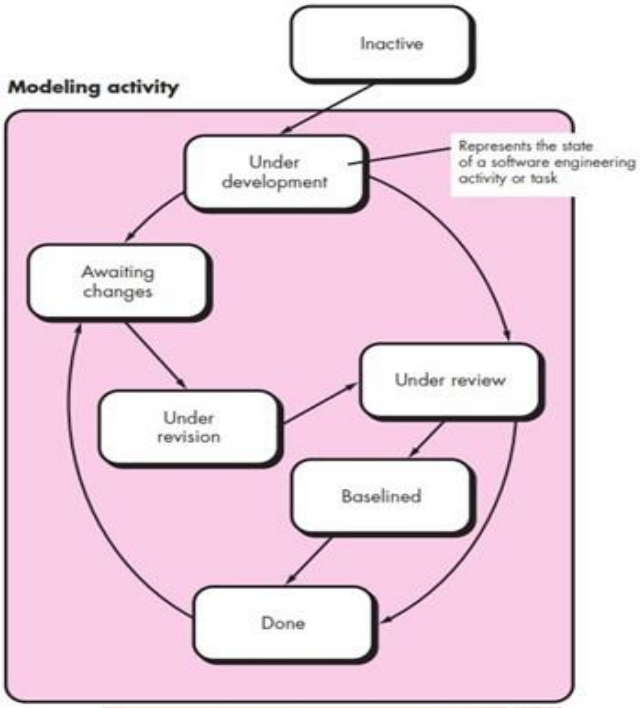

Fig 2.5: One element of the Concurrent Process Model

4. SUMMARY

In the software development life cycle, the software design process is a crucial step that makes sure the software system is high-quality and fits the requirements. An outline of the phases involved in software development is provided by the software development life cycle (SDLC). The SDLC typically includes the following stages:

- Requirements gathering and analysis
- Design
- Implementation
- Testing
- Deployment
- Maintenance

Each stage of the SDLC is finished in a particular sequence under the Waterfall model, which uses a linear sequential method. It is a well-established, strict method that works best for tasks where needs are clear and changes are not expected.

The Iterative model is an incremental and iterative approach in which the software is created over the course of several iterations. Every iteration involves every step of the SDLC, including gathering requirements, designing, implementing, testing, and deploying. An Incremental model is an iterative approach where the software is developed in incremental stages. Each stage adds a new feature to the software, with each subsequent stage building on the previous one.

The Spiral model is a risk-driven approach that emphasizes identifying and mitigating risks throughout the SDLC. It involves multiple iterations of the SDLC, with each iteration adding more functionality to the software.

A Concurrent model is an approach where multiple stages of the SDLC are executed concurrently. It is suited for large and complex projects that involve multiple teams working on different components of the software simultaneously.

The choice of software process model depends on the specific needs and requirements of the project. Each model has its advantages and disadvantages, and it is important to select the appropriate model based on the project requirements and constraints.

5. SELF-ASSESSMENT QUESTIONS

SELF-ASSESSMENT QUESTIONS – 1

1. The process of converting user requirements into a suitable form _____
2. The design should ensure that the software system meets the _____ specified in software requirements specification and also non-functional requirements
3. The benefit of software design is that it _____ development and _____ costs
4. _____ is a process followed by software engineering teams to design, develop, test, deploy, and maintain software systems.
5. A _____ design specification is created by the team based on the needs discovered in the previous stage.
6. Software development teams use frameworks called _____ to direct the process of creating new software.
7. Waterfall model is also known as the _____
8. In the_____, the software development process is broken down into small cycles or iterations, with each iteration consisting of the requirements gathering, design, implementation, and testing phases.
9. The Iterative Model is commonly used in Agile techniques like_____.
10. The iterative approach promotes _____ as the feedback is gathered and incorporated into each iteration.
11. The spiral model mainly focuses on _____
12. An Incremental model is an _____ where the software is developed in incremental stages.

6. SELF-ASSESSMENT ANSWERS

1. Software Design
2. Functional Requirements
3. Speeds Up, Lower
4. Software Development Life Cycle (SDLC)
5. High-Level and Low-Level
6. Software Development Models
7. Linear-Sequential Life Cycle Model
8. Iterative Model
9. Scrum and Kanban
10. Stakeholder Engagement and Collaboration
11. Risk Analysis
12. Iterative Approach

7. TERMINAL QUESTIONS

1. What is a software design and why is it important in software engineering?
2. Elucidate the different phases of the software development life cycle?
3. compare and contrast different software development life cycle models and evaluate their suitability for a given project?
4. Analyse the impact of design decisions on different phases of the waterfall model?
5. Explicate the advantages and disadvantages of the waterfall model.
6. Elucidate the different phases of the Spiral Model.
7. Briefly explain the Incremental and Iterative model with a neat diagram.
8. Explain the advantages and disadvantages of different software development models.
9. Explain the Objectives of Software Design.

8. TERMINAL ANSWERS

1. Refer to section 2.1
2. Refer to section 2.2
3. Refer to section 2.3
4. Refer to section 2.3.1
5. Refer to section 2.3.1
6. Refer to section 2.3.4
7. Refer sections 2.3.2 & 2.3.3
8. Refer to section 2.3
9. Refer to section 2.1.1