NEERAJ®

M.C.S.-213
Software Engineering

Chapter Wise Reference Book
Including Many Solved Sample Papers

Based on

I.G.N.O.U.

& Various Central, State & Other Open Universities

By: *Anand Prakash Srivastava*

NEERAJ PUBLICATIONS
(Publishers of Educational Books)

Mob.: 8510009872, 8510009878 E-mail: info@neerajbooks.com
Website: www.neerajbooks.com

MRP ₹ 350/-

Content

SOFTWARE ENGINEERING

S.No.	*Chapterwise Reference Book*	*Page*

BLOCK-1: AN OVERVIEW OF SOFTWARE ENGINEERING

BLOCK-2: SOFTWARE PROJECT MANAGEMENT

■■

Sample Preview of the Solved Sample Question Papers

Published by:

NEERAJ PUBLICATIONS

www.neerajbooks.com

QUESTION PAPER

June – 2023

(Solved)

SOFTWARE ENGINEERING

M.C.S.-213

Time: 3 Hours] ***[Maximum Marks : 100***
(Weightage : 70%)

Note: *(i) Question No. 1 is compulsory. (ii) Attempt any three questions from the rest.*

Q. 1. ***(a)*** **Explain the following approaches for the development of mobile applications:**

(i) **Native Application Development**

Ans. Ref.: See Chapter-10, Page No. 118, 'Native Application Development'.

(ii) **Rapid Mobile Application Development (RMAD)**

Ans. Ref.: See Chapter-10, Page No. 119, 'Rapid Mobile Application Development'.

(iii) **Progressive Web Applications (PWAs)**

Also, mention any two advantages for each.

Ans. Ref.: See Chapter-10, Page No. 119, 'Progressive Web Applications".

(b) **Discuss the *(i)* requirements related optimization and *(ii)* architecture and design related optimizations in context of First Time Right (FTR) framework.**

Ans. Ref.: See Chapter-13, Page No. 145, 'Software Process Optimisation Through First Time Right Framework' and Page No. 146, 'Requirements Related Optimizations'.

(c) **In context to software project estimation, explain the following (highlighting their main tasks):**

(i) **Estimating the project-size**

(ii) **Estimating efforts**

(iii) **Estimating the schedule**

(iv) **Estimating the total cost**

Ans. Ref.: See Chapter-5, Page No. 60, 'Software Project Estimation'.

(d) **Discuss the following software engineering models:**

(i) **Waterfall model**

Ans. Ref.: See Chapter-1, Page No. 2, 'Waterfall model.

(ii) **Spiral model**

Ans. Ref.: See Chapter-1, Page No. 3, 'Spiral model'.

Q. 2. ***(a)*** **Discuss the Human Computer Interface (HCI) and User Experience (UX) and designing for mobility aspects of software design phase.**

Ans. Ref.: See Chapter-3, Page No. 34, 'Design of Human-Computer Interface' and Page No. 35, 'User Experience (UX) Design'.

(b) **Explain defect metrics and maintainability metrics for measurement of software quality.**

Ans. Ref.: See Chapter-4, Page No. 45, 'Measurement of Software Quality'.

Q. 3. ***(a)*** **Define cleanroom software engineering. List and explain the principles for the cleanroom based software development.**

Ans. Ref.: See Chapter-12, Page No. 135, 'Cleanroom Software Engineering', 'Cleanroom Software Engineering Principles and Strategy and Process Overview'.

(b) **Explain the following emerging trends in software engineering highlighting their salient features, tools, technologies, purpose of usage and advantages:**

(i) **Low Code and No Code platforms**

Ans. Ref.: See Chapter-14, Page No. 160, 'Low Code and No Code Platforms'.

(ii) **Containerization**

Ans. Containerization

1. Salient Features:

- Containerization is a virtualization method that allows applications to be packaged along with their dependencies, libraries, and runtime environment in containers.
- Containers provide lightweight, portable, and isolated execution environments for applications, ensuring consistency across different computing environments.
- Containerization uses technologies like Docker, Kubernetes, and container orchestration

QUESTION PAPER

December – 2022

(Solved)

SOFTWARE ENGINEERING

M.C.S.-213

Time: 3 Hours] ***[Maximum Marks : 100***
(Weightage : 70%)

Note: *(i) Question No. 1 is compulsory. (ii) Attempt any three questions from the rest.*

Q. 1. ***(a)*** **What are the different methods used for estimating cost and efforts required for completing a project successfully? Explain any one of the methods.**

Ans. Ref.: See Chapter-5, Page No. 60, 'Software Project Estimation' and Page No. 61, 'Cocomo Model'.

(b) **Explain McCall's Quality factors.**

Ans. McCall's Quality Factors, also known as McCall's Quality Model, is a software quality model developed by John McCall in the 1970s. It is one of the earliest and most influential models for evaluating software quality. McCall's model identifies 11 factors that contribute to the overall quality of a software product. These factors are grouped into three categories: product revision, product transition, and product operation.

1. Product Revision Factors:

Correctness: The degree to which the software meets its specified requirements and produces accurate results.

Reliability: The ability of the software to perform consistently and predictably under various conditions without failure.

Efficiency: The software's ability to utilize system resources effectively to achieve its intended functions in a timely manner.

Integrity: The ability of the software to protect data and maintain its consistency and accuracy over time.

2. Product Transition Factors:

Usability: The ease of learning, understanding, and using the software by its intended users.

Maintainability: The ease with which the software can be modified, updated, or repaired to meet changing requirements or fix issues.

Flexibility: The ability of the software to adapt and accommodate changes or enhancements without major redesign efforts.

3. Product Operation Factors:

Portability: The ease with which the software can be transferred or deployed across different platforms or environments.

Reusability: The extent to which software components or modules can be reused in different applications or contexts.

Interoperability: The ability of the software to interact and operate seamlessly with other systems, software, or components.

McCall's Quality Factors provide a structured framework for assessing and improving software quality throughout its lifecycle. By considering these factors, software developers and quality assurance professionals can identify areas of strength and weakness in a software product and implement measures to enhance its overall quality and performance.

(c) **Explain various steps involved in Debugging.**

Ans. Ref.: See Chapter-7, Page No. 89, Q. No. 3.

(d) **Briefly explain the process of Software Configuration Management.**

Ans. Ref.: See Chapter-11, Page No. 133, Q. No. 5.

Q. 2. ***(a)*** **Write a short note on 'Human Computer Interface (HCI)'.**

Ans. A Human-Computer Interface (HCI) refers to the point of interaction between a human user and a computer system. It encompasses all aspects of how users interact with computers and the design principles used to facilitate effective and efficient interactions. HCI plays a crucial role in ensuring that computer systems are user-friendly, intuitive, and accessible to a wide range of users.

Key elements of HCI include:

1. User Interface Design: This involves designing the graphical user interface (GUI) elements such as menus, buttons, icons, and widgets that users interact

Sample Preview of The Chapter

Published by:

NEERAJ PUBLICATIONS

www.neerajbooks.com

SOFTWARE ENGINEERING

Software Engineering and its Models

1

INTRODUCTION

The field of software engineering is connected with the advancement of software. Huge software needs precise advancement not at all like basic programmes, which can be created in disconnection and there may not be any orderly methodology being followed.

Over the most recent couple of many years, the computer business has gone through progressive changes in equipment. That is, processor innovation, memory innovation and incorporation of gadgets have changed quickly. As the product is expected to keep up with similarity with equipment, the intricacy of programming additionally has changed much in the new past. In 1970s, the projects were little, basic and executed on a straightforward uniprocessor framework.

Along these lines, the requirement for its use is acknowledged to design methods in their turn of events. The utilisation of designing way to deal with programming advancement lead to the development of the area of Software Engineering. The IEEE glossary of computer programming wording characterises the Software Engineering as:

(a) "The use of an orderly, restrained, quantifiable way to deal with the advancement, activity and upkeep of programming, or at least, the utilisation of designing to programming." *(b)* The study of approaches in *(a)*.

There is a contrast among programming and Software Engineering. Programming incorporates exercises like expense assessment, time assessment, planning, coding, documentation, upkeep, quality affirmation, testing of programming and so forth though programming incorporates just the coding part.

CHAPTER AT A GLANCE

EVOLUTION OF SOFTWARE ENGINEERING

Any application on computer goes through programming. As computer advances have changed hugely over the most recent fifty years, in like manner, the product improvement has gone through huge changes over the most recent couple of many years of 20th century. The programme improvement was reliant upon the programmer's abilities and no essential programming rehearses were available. In the mid 1980s, the size of programming and the application area of programming expanded.

The following summarises the evolution of software:

1960s: Infancy Machine Code.

1970s: Project Years of Higher Order Languages.

1980s: Project Years of Project Development.

1990s: Process and Production Era of Software Reuse.

Some of the common misunderstandings regarding software development are given below:

1. It is not difficult to correct mistakes.
2. By legitimate advancement of programming, it can work impeccably at first time.
3. Free true definition can be utilized at a beginning stage.
4. More labour can be added to accelerate the turn of events.

Software Standards

Various terms related with software engineering are routinely normalised by associations like IEEE (Institute of Electrical and Electronics Engineers), ANSI (American National Standards Institute), OMG (Object Management Group), CORBA (Common Object Request Broker Architecture).

IEEE routinely distributes programming advancement norms.

SOFTWARE DEVELOPMENT MODELS

Software is different from other engineering products in the following ways:

1. Engineering items once created can't be changed.
2. The other engineering items are apparent, however, the product as such isn't noticeable.
3. Software doesn't fail in the customary sense. The software is considered as failed if:
   (a) It doesn't work accurately.
   (b) It does not give the necessary number of featured.

NEERAJ PUBLICATIONS

www.neerajbooks.com

4. First and foremost, a software product must satisfy the needs of the end-user as well as that of the business.
5. Additionally, software development and maintenance should be affordable.
6. Software development should be completed within the specified time frame.
7. Many software products that serve various industries are available in the market today.

Importance of Software Engineering

Software engineering is important because specific software is needed in almost every industry, in every business and for every function. It becomes more important as time goes on – if something breaks within your application portfolio, a quick, efficient and effective fix needs to happen as soon as possible.

Thus, the organisation has to answer the following or similar queries of clients:

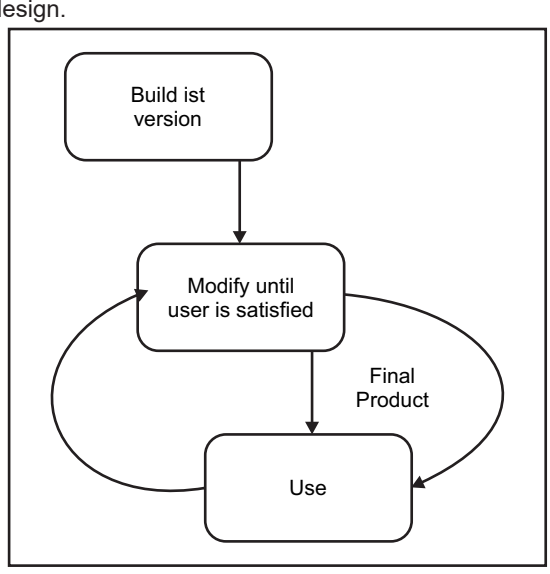
1. What is the best approach to design a software?
2. Why the cost of software is too high?
3. Why can't we find all errors?
4. Why is there always some gap between claimed performance and actual performance?

To answer every single such inquiry, programming improvement has taken on a precise methodology Software development shouldn't stay a work-manship. Logical reason for cost, span, gambles, abandons and so on are required.

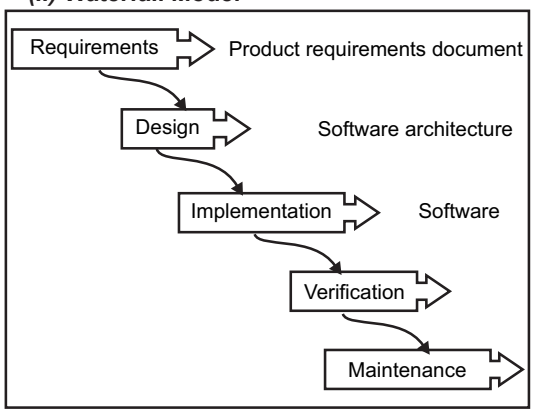
Various Development Models

The following are some of the models adopted to develop software:

***(i)* Build and Fix Model:** In the **build and fix model** (also referred to as an **ad hoc model**), the software is developed without any specification or design.



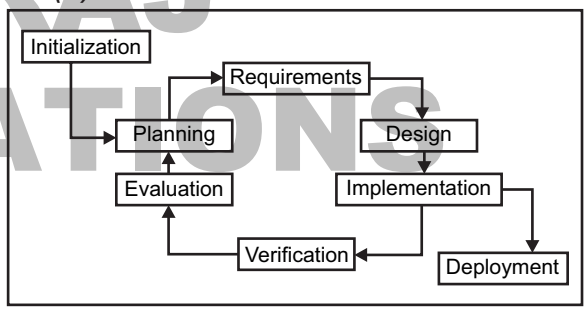
***(ii)* Waterfall Model**



The Waterfall Model is a linear sequential flow, often used with projects that have a defined set of requirements. Reflective of its name, the model's process flows steadily downwards through the phases of software implementation.

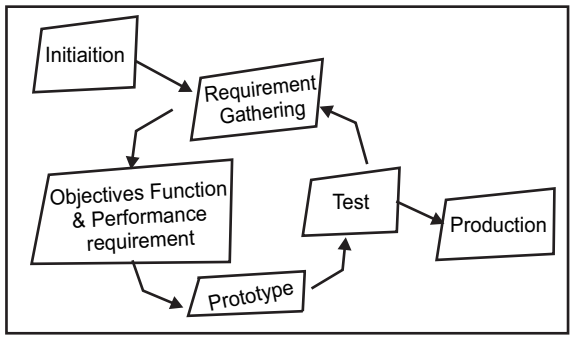
The Waterfall Model should be used with projects that do not anticipate unforeseen changes in mid-development.

***(iii)* Iterative Enhancement Model**



The Iterative Model relies on specifying and implementing individual parts of the software, rather than attempting to start with full specification requirements. Once a rough product is created within an iteration, it is then reviewed and improved in the next iteration and so on.

***(iv)* Prototype Model**



The Prototype Model relies on creating prototypes of the software applications or system software that are used to visualise various components of the software.

NEERAJ PUBLICATIONS

www.neerajbooks.com

www.neerajbooks.com

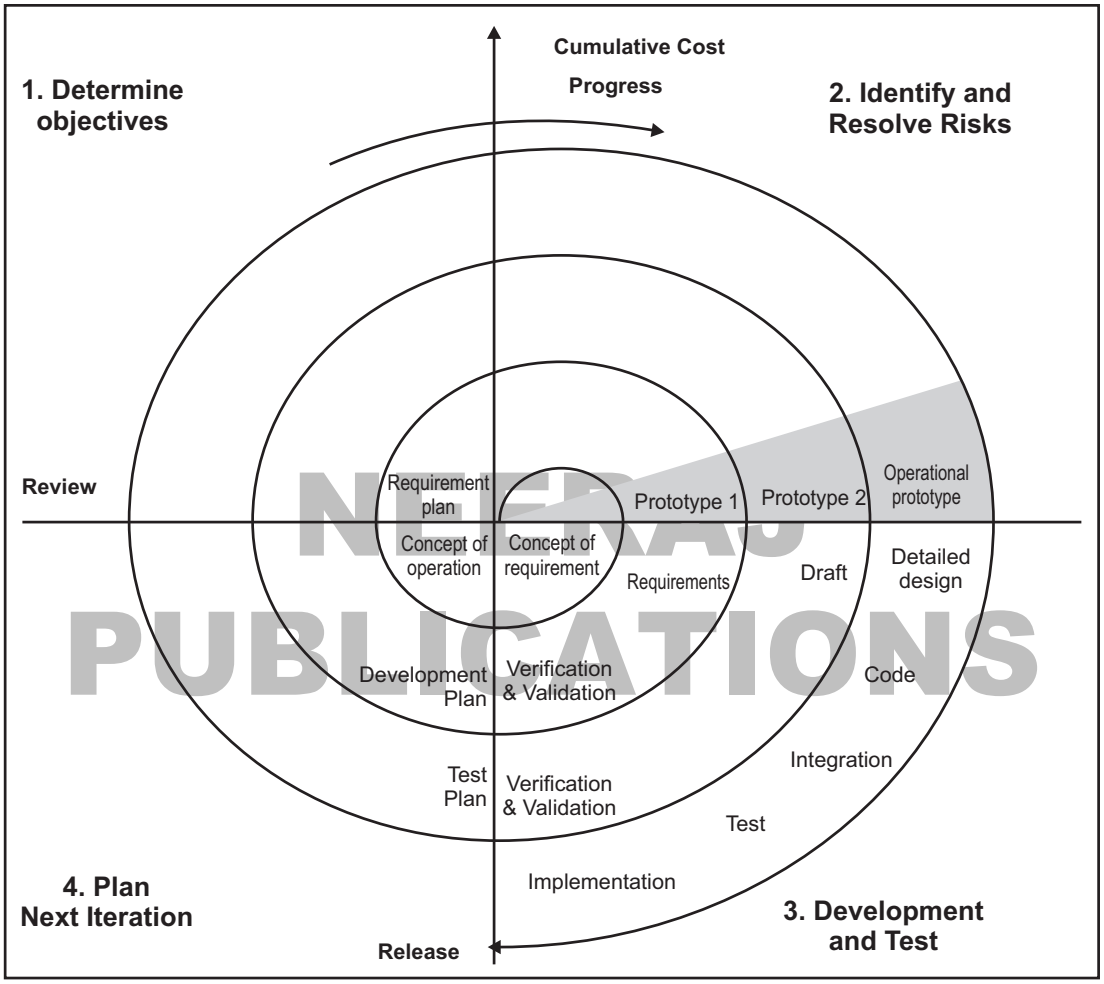
The model has the accompanying highlights:

(a) It helps in deciding client prerequisites all the more profoundly.

(b) At the hour of genuine item improvement, the client input is accessible.

(c) It thinks about any kinds of dangers at the underlying level.

***(v)* Spiral Model**



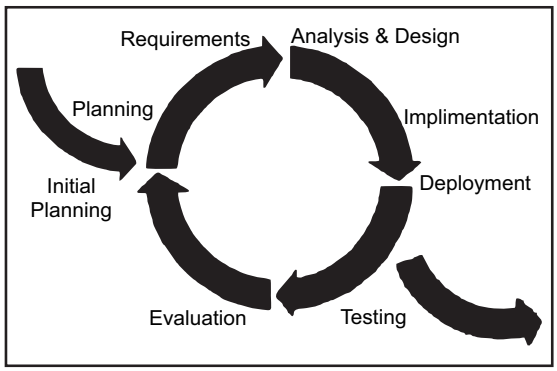
NEERAJ PUBLICATIONS

The Spiral Model combines elements of both, the Iterative and Waterfall development models in efforts to combine advantages of top-down and bottom-up production. The Spiral Model has four phases–Identification, Design, Construct/Build, Evaluation and Risk Analysis.

The following are the primary activities in this model:

- **Finalizing Objective:** The goals are set for the specific period of the project.
- **Risk Analysis:** The dangers are recognised to the conceivable degree. They are broke down also, important advances are taken.
- **Development:** Based on the dangers that are distinguished, an SDLC model is chosen, what's more, is followed.
- **Planning:** At this point, the work done this time is checked on.

***(vi)* RAD Model**



www.neerajbooks.com

www.neerajbooks.com

Short for Rapid Application Development, the RAD Model is a modification of the Incremental Model. When implementing this model, several components are developed simultaneously as if they were smaller, individual projects. The different components are then assembled into working prototypes.

CAPABILITY MATURITY MODELS

The Capability Maturity Model (CMM) of Software Engineering Institute (SEI) specifies an increasing series of levels of a software development organisation. The higher the level, the better the software development process hence, reaching each level is an expensive and time-consuming process.

Maturity Levels

It defines five maturity levels as described below. Different organisations are certified for different levels based on the processes they follow:

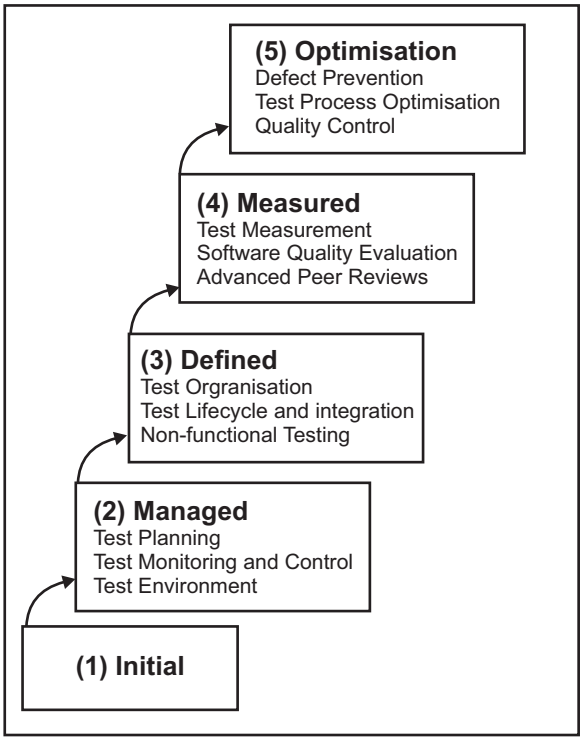
Level 1 (Initial): The software process is haracterized as inconsistent and occasionally even chaotic.

Level 2 (Repeatable): This level of Software Development Organisation has a basic and consistent project management process to track cost, schedule, and functionality.

Level 3 (Defined): The software process for both management and engineering activities are documented, standardised and integrated into a standard software process for the entire organisation.

Level 4 (Managed): At this level, organisation sets a quantitative quality goal for both software process and software maintenance.

Level 5 (Optimizing): The key characteristic of this level is focusing on continually improving process performance through both incremental and innovative technological improvements.



Key Process Areas

Each maturity level is composed of key process areas. Each key process area identifies a cluster of related activities that, when performed collectively, achieve a set of goals considered important for establishing process capability at that maturity level. For example, one of the key process areas for Level 2 is Software Project Planning.

Goals: The goals summarise the key practices of a key process area and can be used to determine whether an organisation or project has effectively implemented the key process area. The goals signify the scope, boundaries and intent of each key process area.

Commitments: The prerequisites that the association ought to meet to guarantee the guaranteed nature of item.

Abilities: The abilities an association has.

Activities: The particular undertakings expected to accomplish KPA work.

Techniques for shifting execution: It makes sense of how the KPAs can be confirmed.

18 KPAs are defined by SEI and associated with different maturity levels. These are described below:

Level 1 KPAs: There is no key process area at Level 1.

Level 2 KPAs:

1. Software Project Planning is to establish reasonable plans for performing the software engineering and for managing the software project.

2. Software Project Tracking and Oversight is to establish adequate visibility into actual progress so that management can take effective actions when the software project's performance deviates significantly from the software plans.

3. It combines the concerns of Requirements Management, Software Project Planning and Software Project Tracking and Oversight for basic management control, along with necessary coordination of Software Quality Assurance and Software Configuration Management, and applies this control to the subcontractor as appropriate.

4. Software Subcontract Management is to select qualified software subcontractors and manage them effectively.

5. Software Quality Assurance (SQA) is to provide management with appropriate visibility into the process being used by the software project and of the products being built.

6. Software Configuration Management (SCM) is to establish and maintain the integrity of the products of the software project throughout the project's software life cycle.

Level 3 KPAs:

1. Organisation Process Focus (OPF) is to establish the organisational responsibility for software process activities that improve the organisation's overall software process capability.

NEERAJ PUBLICATIONS

www.neerajbooks.com