SRINIVAS UNIVERSITY

COLLEGE OF COMPUTER SCIENCE & TECHNOLOGY

CITY CAMPUS, PANDESHWAR,
MANGALORE-575 001

BACKGROUND STUDY MATERIAL

SOFTWARE ENGINEERING

B.C.A - V SEMESTER

Compiled by
Faculty

SOFTWARE ENGINEERING

CONTENTS

UNIT-1

1. Introduction

The Software Problem
Software Is Expensive
Late, Costly and Unreliable
Problems of Change and Rework
Software Engineering Problem
Problem of Scale
Cost, Schedule and Quality
The Problem of Consistency
The Software Engineering Approach
Phased Development Process
Project Management and Metrics
Assignment 1

2. Software Processes

Software Process
Processes, projects and Products
Components of Software Process
Characteristics of a Software Process
Predictability
Support Maintainability and Testability
Early Defect Removal and Defect Prevention
Process Improvement
Software Development Process
A Process Step Specification
Waterfall Model
Prototype Model
Iterative Enhancement Model
Spiral Model

Project Management Process

Phases of Management Process

Metrics, Measurements and Models

Software Configuration Management Process [SCM]

Configuration Identification

Change Control

Status Accounting and Auditing

Process Management Process

Building estimation models

Process Improvement and Maturity

Assignment 2

UNIT-2

3 . Software Requirement Analysis and Specification

Software Requirement

Need for SRS

Requirement Process

Problem Analysis

Analysis Issues

Informal Approach

Structured Analysis

Prototyping

Requirement Specification

Characteristics of an SRS

Components of SRS

Specification Languages

Structure of a requirements document

Requirement Validation

Requirement Reviews

UNIT 3

Project design management

Design Principles

Problem Partitioning and Hierarchy

Abstraction

Module-Level Concepts

Coupling

Cohesion

Design Notation and Specification

Structure Charts

Specifications

Structured Design Methodology

Restate the Problem as a Data Flow Diagram

Identify the Most Abstract Input and Output Data Elements

First- Level Factoring

Factoring the Input, Output and Transform Branches

Design Heuristics

Design Validation/Verification

Design Reviews

Automated Cross-Checking

Assignment 4

UNIT 4

5. Detailed Design and Programming Management

Module Specifications

Specifying Functional Modules

Detailed Design

PDL

Logic/Algorithm Design

Verification

Design Walkthroughs

Critical Design Review

Consistency Checkers

6. Coding

Programming Practice

Top-Down and Bottom-Up

Structured Programming

Information Hiding

Programming Style

Internal Documentation

Verification

Code Reading

Static Analysis

Symbolic Execution

Proving Correctness

Code Inspections or Reviews

Unit Testing

Assignment 6

UNIT 5

7. Testing and Maintenance

Introduction

Testing Fundamentals

Test Oracles

Top-Down and Bottom-up Approaches

Test Cases and Test Criteria

Psychology of Testing

Type of Testing

Functional Testing

Structural Testing

Levels of Testing

Test Plan

Software Maintenance Activities

Definitions of Software Maintenance

Types of Software Maintenance

Corrective Maintenance

Preventive Maintenance

Adaptive Maintenance

Perfective Maintenance

SOFTWARE ENGINEERING

Total hours:40H

UNIT I

8 hrs.

Introduction

Session 1: Definition of Software, the Software Problem.

Session 2: Software Engineering Approach.

Session 3: Phased Development Process, Software Process.

Session 4: Characteristics of the software Process, Process step Verification.

Session 5: Waterfall Model

Session 6: Iterative Enhancement Model, Spiral model.

Session 7: Process Management Process, SCM.

Session 8: Process Management Process,

UNIT – II

Software Requirements Analysis and Specifications

8 hrs.

Session 9: Software Requirements, Need for SRS,

Session 10: Requirement process

Session 11: Problem Analysis, Analysis Issues,

Session 12: Informal Approach, Structured Analysis

Session 13: Prototyping, Requirements Specification, Characteristics of an SRS

Session 14: Components of an SRS, Specification Languages

Session 15: Structure of a Requirements Document

Session 16: Validation, Requirement Reviews

UNIT – III

Project Design Management

8 hrs.

Session 17: Definition of Design and Design principles

Session 18: Problem partitioning and Hierarchy

Session 19 Abstraction and Module level Concepts.

Session 20: Coupling, Cohesion, Design Notation and Specification.

Session 21: Structure Charts, Specifications, Structured Design Methodology.

Session 22: Restate the Problem as a Data Flow Diagram, Identify the Most Abstract Input and Output Data Elements

Session 23: First- Level Factoring, Factoring the Input, Output and Transform Branches

Session 24: Design Heuristics, Design Validation/Verification, Design Reviews , Automated Cross-Checking

UNIT –IV

Detailed Design

8 hrs.

Session 25: Module specification, Specifying functional module, Detailed design, PDL

Session 26: Logic/Algorithm Design, Verification, Design Walkthroughs

Session 27: Critical Design Reviews, Consistency checkers

Session 28: Programming Practice, Top-Down and Bottom-Up

Session 29: Structured Programming, Information Hiding Programming Style, Internal Documentation

Session 30: Verification, Code Reading, Static Analysis

Session 31: Symbolic Execution and Execution Tree

Session 32: Proving Correctness, Code Inspections or Reviews, Unit Testing

UNIT – V

Testing and Maintenance

8 hrs.

Session 33: Testing Fundamentals, Error, Fault, and Failure, Test Oracles

Session 34: Top-Down and Bottom –Up Approaches, Test Cases and Test Criteria

Session 35: Psychology of Testing, Functional Testing

Session 36: Equivalence class partitioning, Boundary value analysis

Session 37: Cause-effect graphing, Structural Testing,

Session 38: Control flow based criteria, Data flow based testing

Session 39: Preventive and Corrective Maintenance

Session 40: Conclusion.

TEXT BOOKS

1. Integrated Approach to Software Engineering by Jalote Pankaj.

Scheme of Evaluation:
The paper carries 100 marks out of which 50 marks will be allotted to external examination and 50 marks will be allotted to the internal assessment.
Internal assessment marks will be calculated as follows

1. Performance in 2 IA examinations will be converted out of 30 marks
2. Attendance 10 marks
3. Assignment 10 marks.
   Total 50 marks.

External examination marks will be as follows

1. 1 marks questions 10 out of 12 1 X 10 = 10 marks.
2. One full question out of 2 full questions in each unit carries 8 X 5 = 40 marks
   Total 50 marks.

In order to clear this paper minimum 50% marks must be scored both in internal and well as external examination.

****

EXAM:	**V SEM**	**PAPER:**	**18BCASD53**
SUBJECT:	**SOFTWARE ENGINEERING**	**CLASS:**	**BCA**
MAXIMUM:	**50**	**TIME:**	**2H**

WEIGHTAGE TO OBJECTIVES TABLE

SL.NO	OBJECTIVES	MARKS	% MARKS
1.	Knowledge (Remembering)	05	10
2.	Understanding	20	40
3.	Application	15	30
4.	Skill	10	20
Total		**50**	**100**

BLUE PRINT

EXAM:	**V SEM**	**PAPER:**	**18BCASD53**
SUBJECT:	**SOFTWARE ENGINEERING**	**CLASS:**	**BCA**
MAXIMUM:	**50**	**TIME:**	**2H**

Unit	REMEMBERING			UNDERSTAND			APPLICATION			SKILL			Total
	OT	SA	Unit wise Marks	OT	SA	Unit wise Marks	OT	SA	Unit wise Marks	OT	SA	Unit wise Marks	
1	1(1)	1(4)	5	1(1)	1(4)	5							10
2				2(1)	1(4)	6		1(4)	4				10
3					1(4)	4	2(1)	1(4)	6				10
4				1(1)	1(4)	5		1(4)	4	1(1)		1	10
5							1(1)		1	1(1)	2(4)	9	10
	05			20			15			10			50

UNIT-1
CHAPTER 1

INTRODUCTION TO S/W ENGINEERING

Introduction

The use of computers is growing very rapidly. Now computer systems are used in areas like business applications, scientific work, video games, aircraft control, missile control, hospital management, airline-reservation etc.

The Software Problem

Software is not only a collection of computer programs. There is a distinction between a *program* and *programming system's product*. A program is generally complete in itself and is used usually by the author of the program.

A programming system's product is used largely by people other than the developers of the system. The users may be from different backgrounds, so a proper user-interface should be provided. There is sufficient documentation to help the users to use the system.

IEEE defines **Software** as the collection of computer programs, procedures, rules and associated documentation and data. This definition clearly states that, software is not just programs, but includes all the associated documentation and data.

Note: IEEE stands for Institute of Electrical and Electronic Engineers

Software Is Expensive

The main reason for the high cost of the software is that, software development is still labor-intensive. In olden days, hardware was very costly. To purchase a computer lacks of rupees were required. Now a days hardware cost has been decreased dramatically. Now software can cost more than a million dollars, and can efficiently run on hardware that costs almost tens of thousands of dollars.

Late, Costly and Unreliable

There are many instances quoted about software projects that are behind the schedule and have heavy cost overruns. If the completion of a particular project is delayed by a year, the cost of the project may be double or still more. If the software is not completed in the scheduled period, then it will become very costly.

Unreliability means, the software does not do what it is supposed to do or does something it is not supposed to do. In software, failures occur due to bugs or errors that get introduced during the design and development process. Hence, even though the software may fail after operating correctly for sometime, the bug that causes the failure was there from the start. It only got executed at the time of failure.

Problem of Change and Rework

Once the software is delivered to the customer, it enters into maintenance phase. All systems need maintenance. Software needs to be maintained because there are often some residual errors remaining in the system that must be removed as they are discovered. These errors once discovered, need to be removed, leading to software getting changed. This is sometimes called as **corrective maintenance**.

Software often must be upgraded and enhanced to include more features and provide more services. This also requires modification of the software. If the operating environment of the software changes, then the software must be modified to the needs of the changed environment. The software must adapt some new qualities to fit to the new environment. The maintenance due to this is called **adaptive maintenance.**

Software Engineering Problem

Software Engineering is a systematic approach to the development, operation, maintenance and retirement of the software. There is another definition for s/w engineering, which states that "Software engineering is an application of science and mathematics by which the capabilities of computer equipments are made useful to man via computer programs, procedures and associated documentation".

Problem of Scale

A fundamental problem of software engineering is the problem of scale. Development of a very large system requires very different set of methods compared to developing a small system. In other words, the methods that are used for developing small systems generally do not scale up to large systems. For example: consider the problem of counting people in a room versus taking the census of a country. Both are counting problems but the methods used are totally different. A different set of methods have to be used for developing large software. Any large project involves the use of technology and project management. In small projects, informal methods for development and management can be used. However, for large projects both have to be much more formal. When dealing with small software project, the technology and project management requirement is low. However, when the scale changes to the larger systems, we have to follow formal methods. For example: if we have 50 bright programmers without formal management and development procedures and ask them to develop a large project, they will produce anything of no use.

Cost, Schedule and Quality

The cost of developing a system is the cost of resources used for the system, which in the case of software are, the manpower, hardware, software and other support resources. The manpower component is predominant as the software development is highly labor-intensive.

Schedule is an important factor in many projects. For some business systems, it is required to build a software with small cycle of time. The developing methods that produce high quality software is another fundamental goal of software engineering. We can view the quality of a software product having three dimensions: Product Operation, Product Transition and Product Revision.

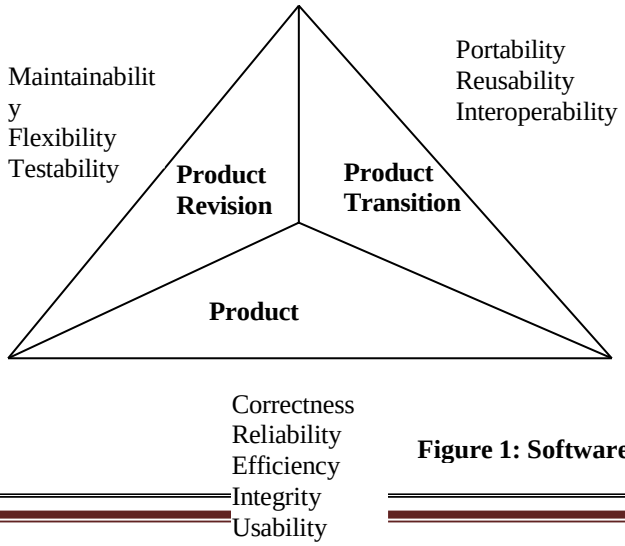


Figure 1: Software Quality Factors

The Product operation deals with the quality factors such as correctness reliability and efficiency. Product transition deals with quality factors such as portability, interoperability. Product revision deals with aspects related to modification of programs, including factors like maintainability and testability.

Correctness is the extent to which a program satisfies its specifications. **Reliability** is the property that defines how well the software meets its requirements. **Efficiency** is the factor in all issues rating to the execution of the software. It includes considerations such as response time, memory requirements and throughput. **Usability** is the effort required to learn and operate the software properly.

Maintainability is the effort required to locate and fix errors in the programs. **Testability** is the effort required to test and check that symbol or module performs correct operation or not. **Flexibility** is the effort required to modify an operational program (functionality).

Portability is the effort required to transfer the software from one hardware configuration to another. **Reusability** is the extent to which parts of software can be used in other related applications. **Inter-operability** is the effort required to couple the system with other systems.

The Problem of Consistency

For an organization there is another goal i.e. consistency. An organization involved in software development does not just want low cost and high quality for a project but it wants these consistently. Consistency of performance is an important factor for any organization; it allows an organization to predict the outcome of the project with reasonable accuracy and to improve its processes to produce higher-quality products. To achieve consistency, some standardized procedures must be followed.

Software Engineering Approach

The objectives of software engineering is to develop methods and procedures for software development that can scale-up for large systems and that can be used consistently to produce high quality software with low cost and small cycle time. The key objectives are high quality, low cost, small cycle time scalability and consistency. To achieve these objectives, design a proper software process and its control becomes the primary goal of software engineering. This process is called development process. The development process must be controlled properly. To do so we have project management which controls the development process to achieve the objectives.

Phased Development Process

A development process consists of various phases, each phase ending with a predefined output.

Software engineering must consist of these activities:

- Requirement specification for understanding and clearly stating the problem.
- Design for deciding a plan for the solution.
- Coding for implementing the planned solution.
- Testing for verifying the programs.

Requirement Analysis

Requirement analysis is done in order to understand the problem to be solved. In this phase, collect the requirement needed for the software project.

The goal of software requirement specification phase is to produce the **software requirement specification document**. The person who is responsible for requirement analysis is called as **analyst.** In problem analysis, the analyst has to understand the problem. Such analysis requires a thorough understanding of the existing system. This requires interaction with the client and end-users as well as studying the existing manuals and procedures. Once the problem is analyzed, the requirements must be specified in the requirement specification document.

Software Design

The purpose of design phase is to plan a solution for the problem specified by the requirement document. The output of this phase is the design document which is the blue-pint or plan for the solution and used later during implementation, testing and maintenance.

Design activity is divided into two phases- **System design** and **detailed design.** System design aims to identify the module that should be included in the system. During detailed design, the internal logic of each of the modules specified during system design is decided.

Coding

The goal of coding is to translate the design into code in a given programming language. The aim is to implement the design in the best possible manner. Testing and maintenance costs are much higher than the coding cost, therefore, the goal of should be to reduce testing and maintenance efforts. Hence the programs should be easy to read and understand.

Testing

After coding phase computer programs are available, which can be executed for testing purpose. Testing not only has to uncover errors introduced during coding, but also errors introduced during previous phases.

The starting point of testing is **unit testing.** Here each module is tested separately. After this, the module is integrated to form sub-systems and then to form the entire system. During integration of modules, **integration testing** is done to detect design errors. After the system is put together, **system testing** is performed. Here, the system is tested against the requirements to see whether all the requirements are met or not. Finally, **acceptance testing** is performed by giving user's real-world data to demonstrate to the user.

CHAPTER 2
SOFTWARE PROCESSES

Software Process

A process means, a particular method of doing something, generally involving several operations. In software engineering, the phrase software process refers to the method of developing the software. A software process is a set of activities together with proper ordering to build high-quality software with low cost and small cycle-time.

The process that deals with the technical and management issues of software development is called a software process. Clearly, many different types of activities need to be performed to develop software. As different types of activities are performed by different people, a software process consists of many components each consisting of many activities.

Processes, Projects and Products

A software process defines a method for developing software. A software project is a development project in which a software process is used. Software products are the outcomes of a software project. Each software development process starts with some needs and ends with some software that satisfies those needs. A software process specifies how the abstract set of activities that should be performed to go from user needs to final product.

The process specifies the activities at an abstract level that are not project specific. It is a generic set of activities and does not provide a detailed roadmap for a particular project. The process also specifies the order in which the activities of a project are carried out.

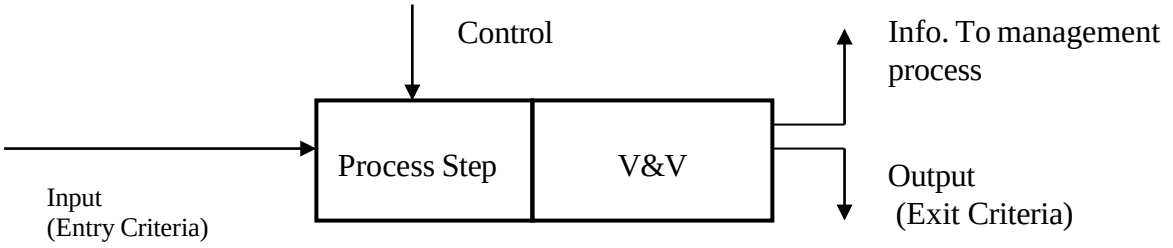
Software Development Process

Software development process focuses on the activities related to the production of the software. For example: design, coding, testing. A development process model specifies some activities that according to the model should be conducted. It also gives the order in which the activity to be performed.

A Process Step Specification

A production process is a sequence of steps. The output of one step will be the input to the next one. The process model will just specify the steps and their order. There are some practical issues such as when to initiate a step, when to terminate a step will not be given by any process model. There must be some verification and validation (V&V) at the end of each step in order to detect the defects. This implies that, there is an early defined output of a phase which can be verified by some means and can form the input to the next phase such products are called **work products.** [Example: Requirement document, design document, code prototype etc.] Having too many steps results in too many work products each requiring V&V can be very expensive. Due to this, the development process typically consists of a few steps producing few documents for V&V.

The major issues in development process are when to initiate and when to terminate a phase. This is done by specifying an entry criteria and exit criteria for a phase. The entry criteria specifies the conditions that the input to the phase should satisfy in order to initiate the activities of that phase. The output criteria specifies the conditions that the work product of current phase should satisfy in order to terminate the activities of the phase.



A step in a development process

Besides the entry and the exit criteria for the input and the output a development step needs to produce some information for the management process. To goal of management process is to control the development process.

Waterfall Model

Waterfall model is the simplest model which states that the phases are organized in a linear order. In this model, a project begins with feasibility analysis. On successfully demonstrating the

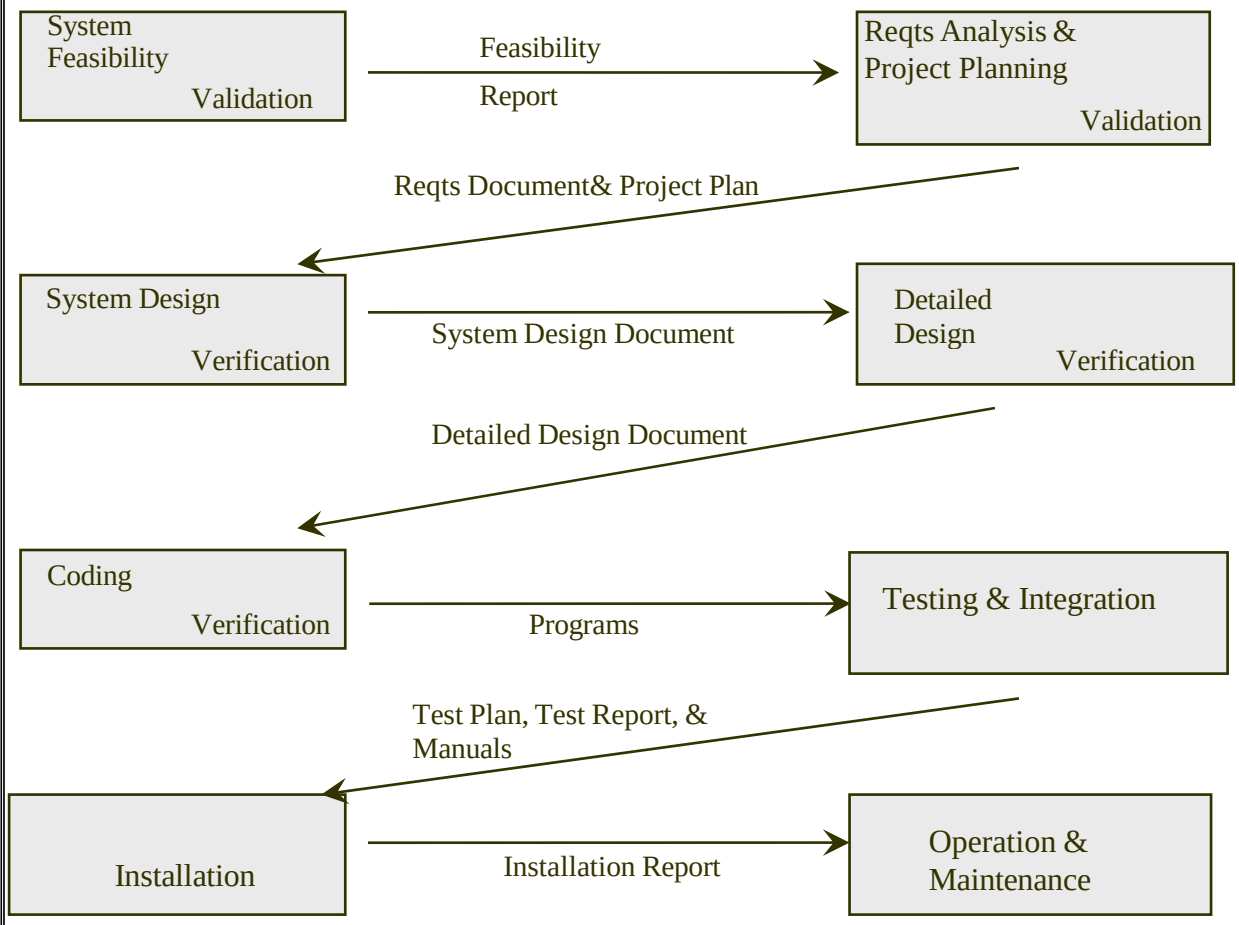
feasibility of a project, the requirement analysis and project planning begins. The design starts after the requirement analysis is complete and the coding begins after the design is complete, once the programming is complete, the code is integrated and testing is done. On successful completion of testing, the system is installed. After this, the regular operations and maintenance take place as shown in the figure (next page).

Each phase begins soon after the completion of the previous phase. Verification and validation activities are to be conducted to ensure that the output of a phase is consistent with the overall requirements of the system. At the end of every phase there will be an output. Outputs of earlier phases can be called as work products and they are in the form of documents like requirement document and design document. The output of the project is not just the final program along with the user manuals but also the requirement document, design document, project plan, test plan and test results.

Project Outputs of the Waterfall Model

- Requirement document
- Project plan
- System design document
- Detailed design document
- Test plan and test report
- Final code
- Software manuals
- Review report.

Reviews are formal meetings to uncover deficiencies in a product. The review reports are the outcomes of these reviews.



Waterfall Model

Limitations of Waterfall Model

1. Waterfall model assumes that requirements of a system can be frozen before the design begins. It is difficult to state all the requirements before starting a project.
2. Freezing the requirements usually requires choosing the hardware. A large project might take a few years to complete. If the hardware stated is selected early then due to the speed at which the hardware technology is changing, it will be very difficult to accommodate the technological changes.
3. Waterfall model stipulates that the requirements be completely specified before the rest of the development can proceed. In some situations, it might be desirable to produce a part of the system and then later enhance the system. This can't be done if waterfall model is used.
4. It is a document driven model which requires formal documents at the end of each phase. This approach is not suitable for interactive applications.
5. In an interesting analysis it is found that, the linear nature of the life cycle leads to "blocking states" in which some project team members have to wait for other team members to

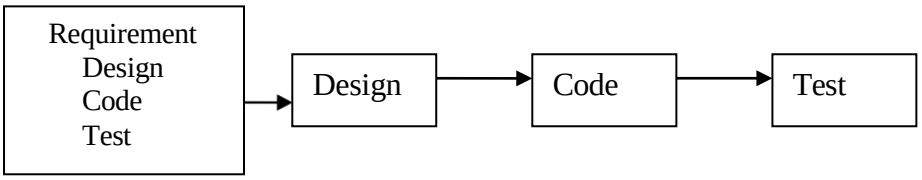
complete the dependent task. The time spent in waiting can exceed the time spent in productive work.

6. Client gets a feel about the software only at the end.

Prototype Model

The goal of prototyping is to overcome the limitations of waterfall model. Here a throwaway prototype is built to understand the requirements. This prototype is developed based on the currently known requirements. Development of the prototype undergoes design, coding and testing, but each of these phases is not done very thoroughly of formally. By using the prototype, the client can get actual feel of the system because the interaction with the prototype can enable the client to better understand the system. This results in more stable requirements that change less frequently. Prototyping is very much useful if there is no manual process or existing systems which help to determine the requirements.

Initially, primary version of the requirement specification document will be developed and the end-users and clients are allowed to use the prototype. Based on their experience with the prototype, they provide feedback to the developers regarding the prototype. They are allowed to suggest changes if any. Based on the feedback, the prototype is modified to incorporate the changes suggested. Again clients are allowed to use the prototype. This process is repeated until no further change is suggested.



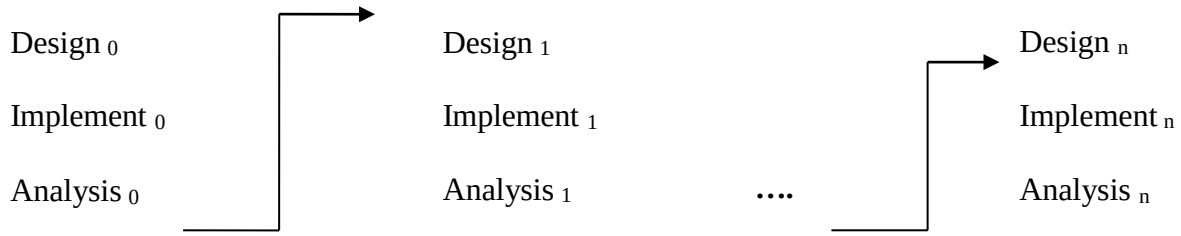
Requirement Analysis The Prototype Model

This model is helpful when the customer is not able to state all the requirements. Because the prototype is throwaway, only minimum documentation is needed during prototyping. For example design document and test plan etc. are not needed for the prototype.

Problems:

This model much depends on the efforts required to build and improve the prototype which in turn depends on computer aided prototyping tools. If the prototype is not efficient, too much effort will be put to design it.

Iterative Enhancement Model



Iterative Enhancement Model

This model tries to combine the benefits of both prototyping and waterfall model. The basic idea is, software should be developed in increments, and each increment adds some functional capability to the system. This process is continued until the full system is implemented. An advantage of this approach is that, it results in better testing because testing each increment is likely to be easier than testing the entire system. As prototyping, the increments provide feedback from the client, which will be useful for implementing the final system. It will be helpful for the client to state the final requirements.

Here a project control list is created. It contains all tasks to be performed to obtain the final implementation and the order in which each task is to be carried out. Each step consists of removing the next task from the list, designing, coding, testing and implementation and the analysis of the partial system obtained after the step and updating the list after analysis. These three phases are called design phase, implementation phase and analysis phase. The process is iterated until the project control list becomes empty. At this moment, the final implementation of the system will be available.

The first version contains some capability. Based on the feedback from the users and experience with the current version, a list of additional features is generated. And then more features are added to the next versions. This type of process model will be helpful only when the system development can be broken down into stages.

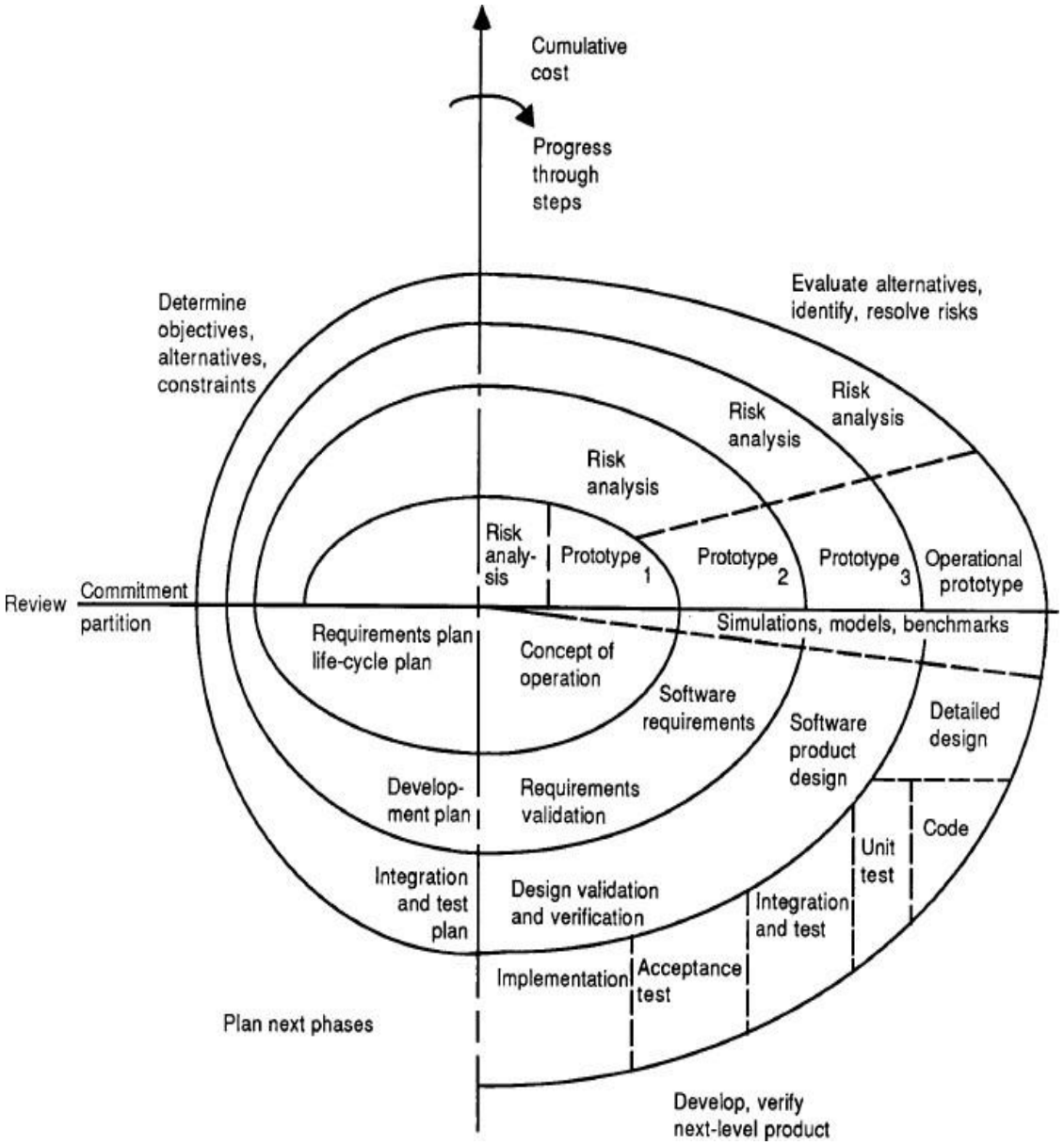
Disadvantage:

This approach will work only if successive increments can actually put into operation.

Spiral Model

As the name suggests, the activities of this model can be organized like a spiral that has many cycles as shown in the above figure. Each cycle in the spiral begins with the identification of objectives for that cycle; the different alternatives that are possible for achieving the objectives and the constraints that exist. This is the first quadrant of the cycle. The next step is to evaluate

different alternatives based on the objectives and constraints. The focus is based on the risks. Risks reflect the chances that some of the objectives of the project may not be met. Next step is to develop strategies that resolve the uncertainties and risks. This step may involve activities like prototyping.



The risk-driven nature of the spiral model allows it to suit for any applications. The important feature of spiral model is that, each cycle of spiral is completed by a review that covers all the products developed during that cycle; including plans for the next cycle. In a typical application of spiral model, one might start with an extra round-zero, in which the feasibility of the basic project objectives is studied. In round-one a concept of operation might be developed. The risks are typically whether or not the goals can be met within the constraints. In round-2, the top-

evel requirements are developed. In succeeding rounds the actual development may be done. In a project, where risks are high, this model is preferable.

Problems:

1) It is difficult to convince the customers that the evolutionary approach is controllable.

2) It demands considerable risk-assessment expertise and depends heavily on this expertise for the success.

3) If major risks are uncovered and managed, major problems may occur.

Software Configuration Management Process [SCM]

SCM is a process of identifying and defining the items in the system, controlling the change of these items throughout their life-cycle, recording and reporting the status of item and change request and verifying the completeness and correctness of these items. SCM is independent of development process. Development process handles normal changes such as change in code while the programmer is developing it or change in the requirement while the analyst is gathering the information. However it cannot handle changes like requirement changes while coding is being done. Approving the changes, evaluating the impact of change, decide what needs to be done to accommodate a change request etc. are the issues handled by SCM. SCM has beneficial effects on cost, schedule and quality of the product being developed.

It has three major components:

1. Software configuration identification
2. Change Control
3. Status accounting and auditing.

Configuration Identification:

When a change is done, it should be clear, *to what*, the change has been applied. This requires a **baseline** to be established. A baseline forms a reference point in the development of a system and is generally defined after the major phases in the development process. A software baseline represents the software in a most recent state. Some baselines are requirement baseline, design baseline and the product baseline or system baseline.

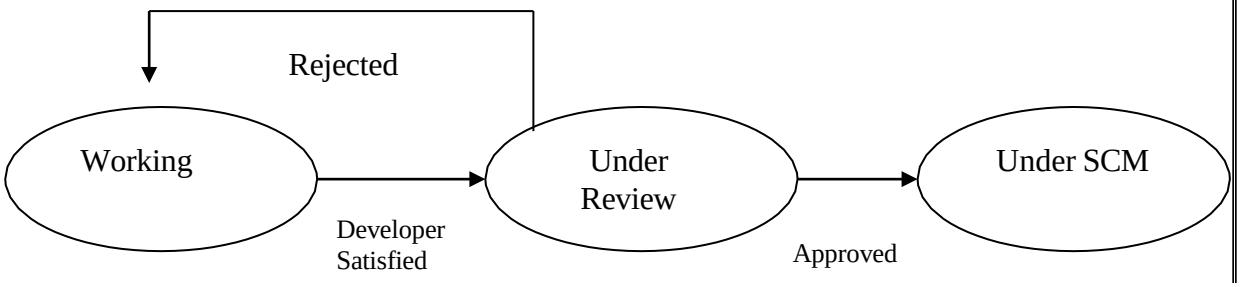
Though the goal of SCM is to control the establishment and changes to these baselines, treating each baseline as a single entity for the change is undesirable, because the change may be limited to a very small portion of the baseline. For this reason, a baseline can consist of many software configuration items. [SCI's] A baseline is a set of SCIs and their relations.

Because a baseline consists of SCIs and SCI is the basic unit for change control, the SCM process starts with identification of the configuration items. Once the SCI is identified, it is given a name and becomes the unit of change control.

Change Control:

Once the SCIs are identified, and their dependencies are understood, the change control procedures of SCM can be applied. The decisions regarding the change are generally taken by the configuration control board [CCB] headed by configuration manager [CM]

When a SCI is under development, it has considered being in working state. It is not under SCM and can be changed freely. Once the developer is satisfied with the SCI, then it is given to CM for review and the item enters to 'under review' state. The CM reviews the SCI and if it is approved, it enters into a library after which the item is formally under SCM. If the item is not approved, the item is given back to the developer. This cycle of a SCI is given in the figure below.



Once the SCI is in the library, it can not be modified, even without the permission of the CM. an SCI under SCM can be changed only if the change has been approved by the CM. A change is initiated by a change request (CR). The reason for change can be anything. The CM evaluates the CR primarily by considering the effect of change on the cost schedule and quality of the project and the benefits likely to come due to this change. Once the CR is accepted, project manager will take over the plan and then CR is implemented by the programmer.

Status Accounting and Auditing

The aim of status accounting is to answer the question like what is the status of the CR (approved/rejected), what is the average and effort for fixing a CR and what is the number of CR. For status accounting, the main source of information is CR. Auditing has a different role.

Quality Improvement Paradigm and GQM

It gives a general method for improving a process, essentially implying that what constitutes improvement of a process depends on the organization to which the process belongs and its objectives. The basic idea behind this approach is to understand the current process, set objectives for improvement, and then plan and execute the improvement actions. The QIP consists of six basic steps:

Part A (Multiple Choice Questions)
Remembering

1. IEEE defines _____as the collection of computer programs, procedures, rules and associated documentation and data. Identify the same

A. SOFTWARE Engineering
B. **software**
C. software model
D. SRS

2. IEEE defines _____is a systematic approach to the development, operation, maintenance and requirement of the software. Identify the same

A. **SOFTWARE Engineering**
B. requirement specification
C. coding
D. SRS

3. Quote from memory that the fundamental goal of software engineering is to produce _____

A. SRS
B. Review report
C. Cost effective design
D. <u>High quality software product</u>

4. Quote from the memory that _____is the effort required to couple system with other system

A. Maintainability
B. **<u>Inter-operability</u>**
C. Portability
D. Quality

5. Quote from memory that _____is the effort required to transfer the software from one hardware configuration to other.

A. Maintainability
B. Inter-operability
C. <u>Portability</u>
D. Reliability

6. State that _____is the effort required to locate and fix the errors in the program.

A. <u>Maintainability</u>
B. Inter-operability
C. Portability
D. Testing

7. State that _____is one of the quality attributes in product revision

A. Portability
B. Efficiency
C. Integrity
D. <u>Testability</u>

8.____is a risk driven model. Identify
A. Waterfall
B. **<u>Spiral</u>**
C. Iterative enhancement
D. Prototype

9. Blocking states is found in _____model. Recognize.
A. W**<u>aterfall</u>**
B. Spiral
C. Iterative enhancement
D. Prototype

10. Freezing of requirement before development process is a limitation in _____, identify
A. **<u>Waterfall</u>**
B. Spiral
C. Iterative enhancement
D. Prototype

Understanding

11. Throw away models are used in _____model of development
A. Waterfall
B. Spiral
C. Iterative enhancement
D. <u>Prototype</u>

12. In which model , software is built in increments
A. Waterfall
B. Spiral
C. <u>Iterative enhancement</u>
D. Prototype

13. Evaluating a software process and identifying the loop holes increases _____of a software
A. Predictability
B. Scalability
C. Cost
D. <u>Quality</u>

14. Development process of software has ____phases
A. Three

B. Four
C. Five
D. Two

15. Software process components are development process, management process and _____process
A. Monitoring
B. SCM
C. Termination analysis
D. Quality analysis

16. Planning, monitoring and control and termination analysis are part of_____
A. Development process
B. Quality analysis
C. SCM
D. Management process

17.___provide quantifiable measures used to measure different characteristics of software product.
A. **Software Metric**
B. Software Model
C. Measurement
D. Mass

18. Expand SCM
A. Software Configuration Model
B. Software Code Management
C. **Software Configuration Management**
D. Software Code Maintenance

19.___is independent of development process
A. design
B. management
C. **SCM**
D. testing

20. Terms like Change Request, Status accounting appear in_____
A. management process
B. development process
C. quality analysis process
D. SCM process

21. Baseline in SCM process has set of _____
A. work products
B. Change Request (CR)
C. **software configuration Item**

D. Change Control Procedures

22. CMM, QIP, GQM are part of___
A. SCM process
B. Process improvement and maturity model
C. status accounting
D. estimation models

23. QIP has __basic steps.
A. four
B. five
C. **six**
D. Three

24. Initial, repeatable, defined, managed and optimizing are levels of____model
A. QIP
B. GQM
C. **CMM**
D. SCI life cycle

25. In SCI life cycle, CR request is approved by_____
a) **Configuration Manager**
b) CCB
c) Project manager
d) developer

Part B (4 marks)

Remember

1. Define Software Engineering. Explain various problem faced in Software Engineering
2. Discover the Quality attributes of Software Engineering
3. Discover different phases of development process
4. State and explain the working of waterfall model with the help of a diagram.
5. List out the limitations of waterfall model

Understand

6. Explain the working of iterative enhancement model
7. Describe the spiral model with the help of diagram
8. Describe the SCM life cycle of an item
9. Explain various activities of Software configuration Management Process