

# Chapter 1

## Process Life Cycle

### LEARNING OBJECTIVES

- Introduction
- Process vs program
- Software component and elements
- Information gathering
- Requirement analysis
- Feasibility study
- Data flow diagram
- Process specification
- Input/output design
- Software process life cycle
- Software process model

### INTRODUCTION

A system can be defined as an orderly grouping of interdependent components linked together according to a plan to achieve a specific objective.

**Example:** Telephone system, transportation system, accounting system, etc.

### PROCESS VERSUS PROGRAM

A software process gives all steps used to create a software application, from the customer's requirements to the finished product.

- The software process determines the organization and flexibility of the project.
- There are several different software processes and each describes their own solution to develop a valid software.
- Software programs are written programs or rules with associated documentation pertaining to the operation of a computer system.

### SOFTWARE COMPONENTS AND ELEMENTS

#### Software Component

It is a software element that can be independently deployed and composed without modification according to a composition standard.

- A component model implementation is the dedicated set of executable software elements required to support the execution of components.
- A component has clearly defined interfaces.

- An interface standard is the mandatory requirement enforced to enable software elements to directly interact with other software elements.
- An interface standard declares, when an interface comprises.

**Standard** An object or measure serving as a basis to which others should conform, by which the quality of others is judged.

#### Software Element

A sequence of abstract program statements that describe computations, which has to be performed by a machine.

#### Interface

It describes the behavior of a component that is obtained by considering only the interactions of that interface and by hiding all other interactions.

- An abstraction of the behavior consists of subset of the interactions of one component together with a set of constraints.

#### Interaction

It is defined as action between 2 or more software elements.

**Composition** It is a combination of 2 or more software components, the newly formed component, behaviour will be at a different level of abstraction.

The characteristics of new component is determined the components combined and the way in which they are combined.

## INFORMATION GATHERING

Complete and accurate information is essential in building computer-based systems. Information about the organization, the staff who uses the system and the workflow should be gathered.

Information about the organization's policies, goals, objectives and structure explains the kind of environment the computer-based system should produce.

Information about the people who run the present system, their job functions and information requirements, the relationships of their jobs to the existing system and the interpersonal network that holds the user groups together are required for determining the importance of the existing system for the organization and also for planning the proposed system.

Workflow focuses on what happens to the data through various points in a system and can be shown by a data flow diagram or a system flow chart.

Information can be gathered by studying documents, forms and files of existing system. Onsite observation of the system is also an effective method for gathering information. It is the process of recognizing and noting people, objects and occurrences to obtain information. Interview is one of the most often and oldest method for gathering information. Interview has the advantage of identifying relations or verifying information and also capture information face to face with the concerned person. Questionnaire is another method for gathering information and is an inexpensive mean for gathering data which can be tabulated and analyzed quickly. Visiting companies that have developed similar systems, reading journals and other computer related books, which specify how others have solved similar problems is also another means of information gathering.

## REQUIREMENT ANALYSIS

Requirement analysis results in specification of software's operational characteristics, indicates software's interface with other system elements and establishes constraints that the software must meet.

During requirement analysis, the primary focus should be on *what* not *how*. It should define what user interaction occurs in a particular circumstance, what objects does the system manipulate, what functions must the system perform, what behaviours does the system exhibit, what interfaces are defined and what constraints applied.

The requirement analysis model must achieve three primary objectives:

1. To describe what the customer requires.
2. To establish a basis for the creation of a software design
3. To define a set of requirements that can be validated once the software is built.

## Requirement Negotiation

Requirement negotiation is required to have a win-win result. The customer should get product which satisfies most of his/her needs, and software team should develop a product within a budget, working in real-time environment and within deadlines.

Boehm has defined negotiation activities at the beginning of each software process iteration.

- Identify the key stake holder's system or subsystem.
- Determine the 'win conditions' of the stake holder.
- Negotiate the 'win conditions' of stake holder and establish them into win-win condition.

## Requirement Elicitation

Requirement elicitation is gathering the requirements from stakeholders, customers, etc. The question and answer format is suitable for the first encounter with users and the remaining phases are replaced with requirement elicitation. As we can't get all the requirements by having questions and answers session, requirement elicitation practices should be implemented, which includes interviews, workshops, user scenarios, etc.

The approaches that are followed for eliciting the requirements are:

1. Collaborative requirement gathering
2. Quality function deployment
3. User scenarios
4. Elicitation work products

## Functional Requirements

Functional requirements are primary actions that must take place in software in accepting and processing the input and in processing and generating the output.

Functional requirements capture the intended behaviour of the system, which could be expressed as service task (or) functions of the system.

These are core functionalities of the system. It also includes exact sequence of operations, input validation, mapping of outputs to inputs and error handling and recovery. These requirements are implemented in system design.

## Non-functional Requirements

Non-functional requirements are expected requirements of a user. These define operational constraints based on the user characteristics.

Non-functional requirements are product, business and external-based. These requirements define how a software system has to be. It also defines the quality of product, type of reliability and usability of the system. Implementation requirements depend on organization, and delivery requirements are defined in the non-functional requirements.

Non-functional requirements are implemented in system-architecture.

## Measuring Requirements

The requirements are the major component of project. The metrics for the requirement activities are:

1. Product size
2. Requirement quality
3. Requirement status
4. Requirement change (request for changes)
5. Effort

**Product size** refers to the count of functional and non-functional requirements. It tracks, whether these requirements are implemented as a function of time.

**Requirement quality** refers to the inspection of specification of requirements, counting the defects, missing of requirements incompleteness, ambiguities, etc.

**Requirements status** is monitoring of requirements over time, gives out project status. The status could be proposed, approved, implemented, verified deferred, deleted, rejected.

**Requirement management** handles the addition, modify and deletion of requirement, track the change of requirement which affects multiple requirements of different level.

**Effort** is the time taken to record requirements activities which includes development and management of requirements.

## Types of Requirements

These are the services that a software system has to provide and constraints under which it must operate.

### User requirements

- Written for customers.
- These statements will be given in natural language and diagrams of the services that the system provides and operational constraints.

### System requirements

- Written as a contract between contractor and client.
- A structured document with detailed descriptions of the system services.

### Software specification

- Written for developers.
- A detailed description of software that can act as basis for a design (or) implementation.

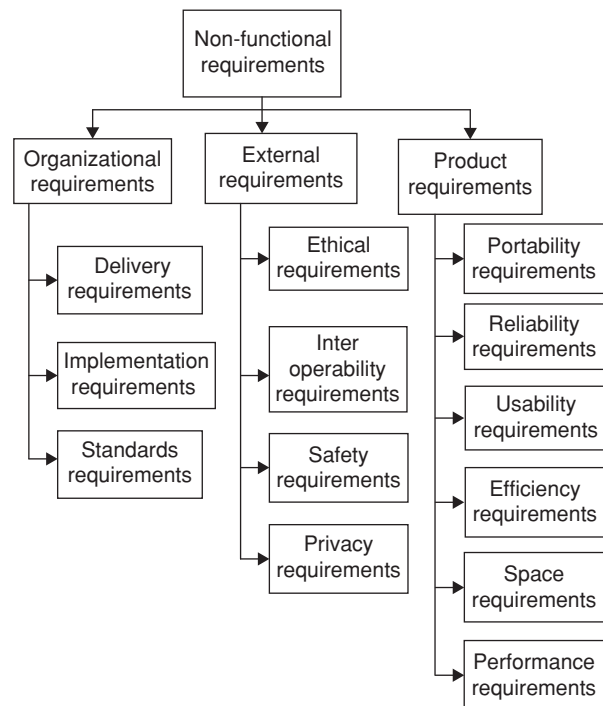
### Functional requirements

The system should provide statements of services, how the system should react to particular inputs and how the system should behave in particular situations.

## Non-functional requirements

Functions offered by the system such as timing constraints, constraints on the development process, standards, etc.

1. **Product requirements:** It specifies, that the delivered product must behave in a particular way.  
Example: Execution speed, reliability
2. **Organizational requirements:** These are consequences of organizational policies and procedures.  
Example: Process standards, implementation requirements
3. **External requirements:** These arise from factors which are external to the system and its development process.  
Example: Interoperability requirements



## Domain requirements

These come from the application domain of the system that reflects the characteristics of the domain.

- These could be functional or non-functional.

## FEASIBILITY ANALYSIS

Feasibility study is a test of a system proposal according to its workability, impact on the organization, ability to meet user needs, and effective use of resources.

The objective of a feasibility study is not to solve the problem but to acquire a sense of its scope. Costs and benefits are estimated with greater accuracy at this stage. Feasibility analysis helps to identify the best solution to the end user.

The key considerations involved in feasibility analysis are:

1. Economic feasibility
2. Technical feasibility
3. Behavioral feasibility

Economic analysis is the most frequently used method for evaluating the effectiveness of a candidate system. Also known as cost/benefit analysis, economic analysis determines the benefits and savings that are expected from the candidate system and compares them with the costs. If benefits outweigh costs, then decision is made to design and implement the system, else alterations are made if it has a chance of being approved.











Technical feasibility is concerned with hardware and software requirements to implement the system. Technical analysis centres around the existing computer system (hardware, software, etc.) identifies, to what extent it can support the proposed addition. Additional hardware and software (OS, databases) requirements are identified and checks whether financial considerations/constraints can accommodate these technical enhancements.

Behavioural analysis makes an estimate of how strong a user staff is likely to react towards the development of a computerized system. Computer installations usually changes employee job status, and also there may be transfer, training period, etc. Thus the introduction of a new system requires special effort to educate, sell and train the staff on new ways of conducting business.

## DATA FLOW DIAGRAMS (DFD)

DFD also called bubble chart, clarifies system requirements and identifies major transformations that will become programs in system design. It functionally decomposes the requirements specification down to the lowest level of detail.

The four DFD symbols are:

1.  (or)  Source/Destination of data
2.  (or)  Data flow
3.  (or)  (or)  Process
4.  (or)  (or)  Data store

The first symbol defines a source or destination of system data. The second symbol specifies data flow direction. It can be considered as a pipeline through which the information flows. The third symbol represents a process that transforms incoming data flows into outgoing data flows, and the fourth symbol is used to represent storage of data.

In short, DFD takes an input-process-output view of a system. That is, data objects flow into the software is transformed by processing elements and resultant data objects flow out of the software. Data objects are represented by labelled arrows and transformations are represented by circles (also called

bubbles). The DFD is presented in a hierarchical fashion. That is, the first data flow model sometimes called level 0 DFD or context diagram represents the system as a whole. Subsequent data flow diagrams refine the context diagram, providing increase in detail with each subsequent level.

## PROCESS SPECIFICATION (PSPEC)

The process specification (PSPEC) is used to describe all flow model processes that appear at the final level of refinement. The content of the process specification can include narrative text, a program design language (PDL) description of the process algorithm, mathematical equations, tables or UML activity diagrams.

By providing a PSPEC to accompany each transformation (bubble) in the flow model, a 'mini-spec' can be created that serves as a guide for design of the software component that will implement the transformation.

## INPUT/OUTPUT DESIGN

### Input Design

The most common cause of errors in data processing is inaccurate input data. Errors occurred during data entry can be controlled by input design.

Input design is the process of converting user-originated inputs to a computer-based format.

Input data is collected and organized into groups of similar data. The goal of designing input data is to make data entry easy, logical and free from errors.

Source data is captured initially on original paper or a source document. A source document should be logical and easy to understand. Each area in the form should be clearly identified and should specify to the user what to write and where to write.

Source documents may enter into the system from punch cards, diskettes, optical character recognition (OCR) reader, Magnetic ink character recognition (MICR) reader, barcode reader, etc. Touch screen or voice input can be used for online data entry, for example, ATM.

There are three major approaches for entering data into the computer—menus, formatted forms and prompts.

A menu is a selection list that simplifies computer data access or entry. The user can choose what to enter from a list of options. Though a menu limits a user choice of responses, it reduces the chances of errors in data entry.

A formatted form is a preprinted form or a template that requests the user to enter data in appropriate locations (fill-in the blank type form). The form is displayed on the screen and the user can fill information by positioning the cursor in appropriate text boxes.

In prompt, the system displays one inquiry at a time, asking the user for a response, for example, asking for user-id and password.

## Output Design

Computer output is the most important and direct source of information to the user. Efficient and intelligible output design will improve system's relationships with the user and helps in decision making.

The devices available for providing computer-based output are printer, CRT screen display, audio response (speaker), plotters, etc.

The task of output preparation is very critical, regaining skill and ability to align user requirements with the capabilities of the system in operation.

## SOFTWARE PROCESS LIFE CYCLE

A software process can be defined as a frame work of the activities, actions and tasks that are required to build quality software.

All these activities, actions and tasks reside within a frame work or model that defines their relationship with the process and with one another.

A generic process frame work for software engineering encompasses five activities:

**Communication** Proper communication and collaboration with the customer is made in this activity to understand the objectives for the project and also to gather requirements that help to define software features and functions.

**Planning** This activity develops a software project plan which defines the software engineering work by specifying the technical tasks to be conducted, the risks that may occur, the resources that will require, the work products to produce and the work schedule.

**Modelling** A software engineer creates models to better understand software requirements and the design that will achieve those requirements.

**Construction** This activity combines code generation and testing required to uncover errors in the code.

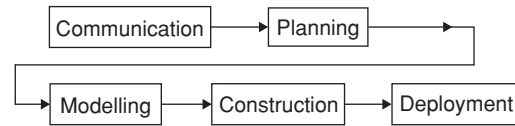
**Deployment** In this activity, the software (as a complete product or as a partial increment) is delivered to the customer. The customer evaluates the delivered product and provides feedback based on evaluation.

Another important aspect of the software process called process flow describes how the frame work activities and the actions and tasks that occur within each framework activity are organized with respect to sequence and time.

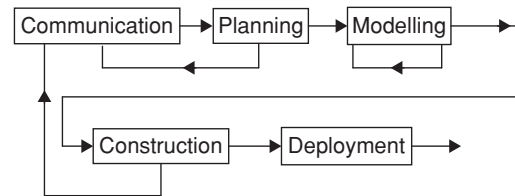
## Process Quality and Improvement

Quality refers to characteristic or attribute of something. Process quality factors are portability, usability, reusability, correctness and maintainability. The process quality is the implementation of the following steps firstly initiates the process and design the solutions, implement these solutions with the impact demonstration.

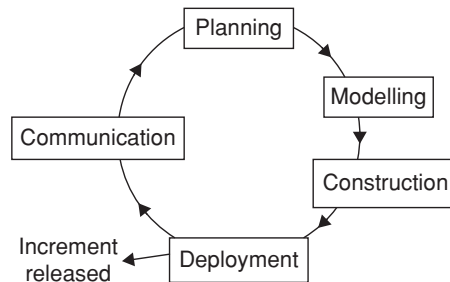
**Linear process flow** executes each of the five framework activities in sequence, beginning with communication and ends with deployment.



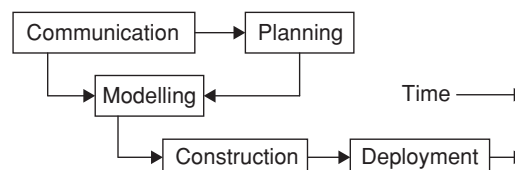
**Iterative process flow** repeats one or more of the activities before proceeding to the next activity.



**Evolutionary process flow** executes the activities in a circular manner. Each circuit through the five activities leads to a more complete version of the software.



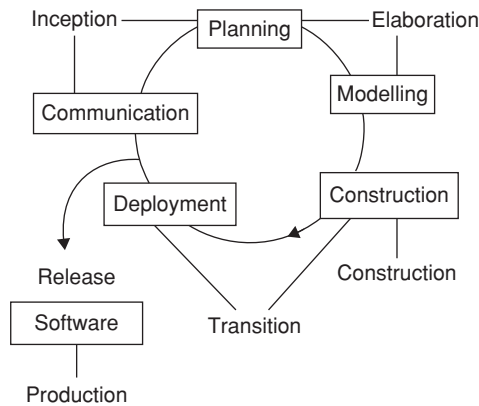
**Parallel process flow** executes one or more activities in parallel with other activities.



## The Unified Process

The unified process (UP) is an attempt to draw on the best features and characteristics of conventional software process models. It recognizes the importance of customer communication and streamlined methods for describing the customer's view of a system. It helps the architect focus on the right goals, such as understandability, reliance to future changes, and reuse. It suggests a process flow that is iterative and incremental, providing the evolutionary feel that is essential in modern software development.





**Figure 1** Phases of the unified process

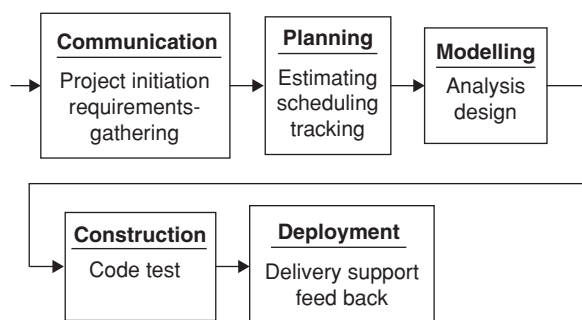
The unified process is an incremental model in which five phases are defined:

1. Inception phase: Encompasses both customer communication and planning activities and emphasizes the development and refinement of use cases as a primary model.
2. Elaboration Phase: Encompasses the customer communication and modelling activities focusing on the creation of analysis and design models with an emphasis on class definitions and architectural representations.
3. Construction phase: Refines and translates the design model into implemented software components.
4. Transition phase: Transfers the software from the developer to the end user for beta testing and acceptance.
5. Production phase: Ongoing monitoring and support are conducted. Defect reports and requests for changes are also submitted and reevaluated.

## SOFTWARE PROCESS MODELS

### The Waterfall Model

The waterfall model also called classic life cycle, follows a systematic sequential approach to software development.



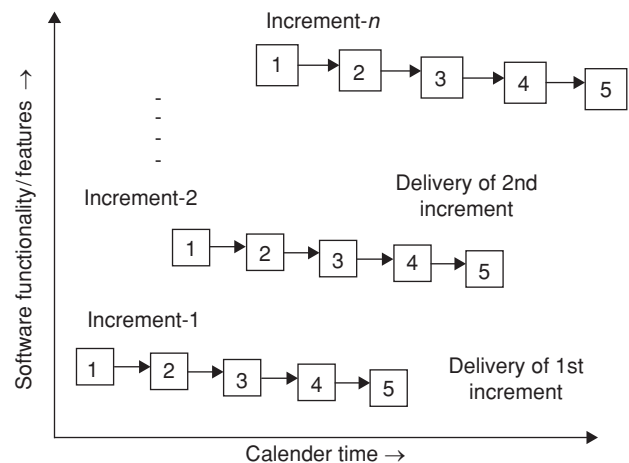
It begins with customer specification of requirements and progresses through planning, modelling, construction and deployment, culminating in on-going support of the completed software.

The waterfall model is the oldest paradigm for software engineering. The problems encountered when this model is applied are:

1. Real projects rarely follow the sequential flow that the model proposes.
2. This model requires the requirements explicitly which the customer cannot state all the requirements as it is difficult.
3. A working version of the program will not be available until late in the project time span. If a major blunder is undetected until the working program is reviewed, it can be disastrous.

### Incremental Process Model

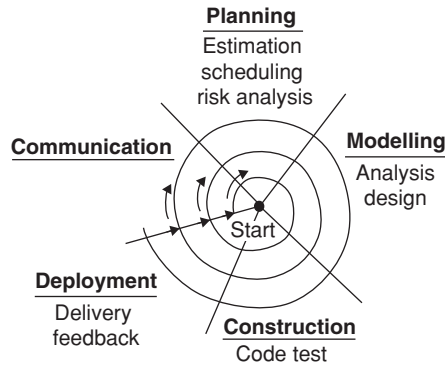
1. Communication
2. Planning
3. Modelling (analysis, design)
4. Construction (code, test)
5. Deployment (delivery, feedback)



Incremental development is particularly useful when staffing is unavailable for a complete implementation by the business deadline that has been established for the project.

### Spiral Model

Spiral model is an evolutionary software process model. Using spiral model, software is developed in a series of evolutionary releases. During early iterations, the release might be a model or prototype. Later iterations produce more complete versions of the system.



The spiral development model is a risk-driven process model generator that is used to guide multi stakeholder concurrent engineering of software intensive systems.

The spiral model is a realistic approach to the development of large scale systems and software. It uses prototyping as a risk reduction mechanism but, more importantly, enables the developer to apply the prototyping approach at any stage in the evolution of the product. At demands a direct consideration of technical risks at all stages of the project.

### Conceptual Modelling

Conceptual modelling refers to abstraction of a model which fits for the purpose. The purpose of this modelling is to make model valid credible, feasible and useful.

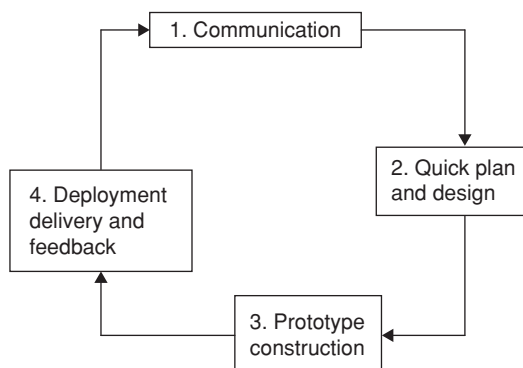
The main objective of conceptual modelling is improvising the understanding of an individual with respect to the system, an approach which will convey the system details among the stakeholders.

For the extraction of system specifications when a software is developed, some of the failures could occur in future due to lack of requirements [unclear requirements (or) changing requirements] This could be traced with the help of conceptual modelling.

### Prototyping Model

Prototyping model is used when the user is not sure about the addition of requirements in the product. It is also implemented when the developer is not sure about the algorithm efficiency, operation system adaptability, etc. Prototyping paradigm provides the approaches.

The prototyping model is implemented as follows:



Prototyping model starts with communication in which software objectives is defined requirement, identification are done. In quick design, all the software aspects are represented quick design leads to prototype construction.

This prototype model is deployed, in which requirements are evaluated and refined by customer.

The iteration is done until customer gets satisfied with the needs at the same time developer will come to know what are the needs to be done.

### Disadvantages

1. Developer may compromise at implementation, as prototyping works quickly. Un-ideal implementation issues may become an integral part of the system.
2. Customer just sees the working version of the software, he could not able to consider the quality of software and long-term maintenance.

Though there are some problems with prototyping, but it is effective paradigm for the software engineering when a software is developed using prototyping, both developer and customer should agree on the prototype.

It is more advantageous when the customer and user are not sure what they want it maintains a template of the older software.

### Role of metrics and measurement in software development

The software attributes that were present in process, project and product levels are called measurement.

The metric refers to the attributes that are included in the project.

A software engineer gets the measurements and develops the metrics.

Measurement is done in two ways:

1. Direct measure
2. Indirect measure

Direct measure includes the lines of code, (least, moderate, worst) execution speed and size of the memory.

Indirect measure is done with the help of functional points. It measures quality, maintainability, efficiency and reliability.

Metric is used to control the cost, schedule, project quality. It means metric provides information for the control of process development.

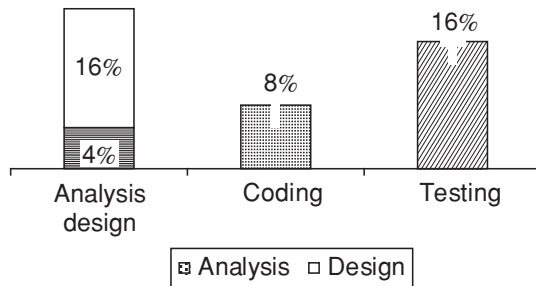
### Effort distribution with phases

Software development is done in phases. It includes analysis, design, coding and testing.

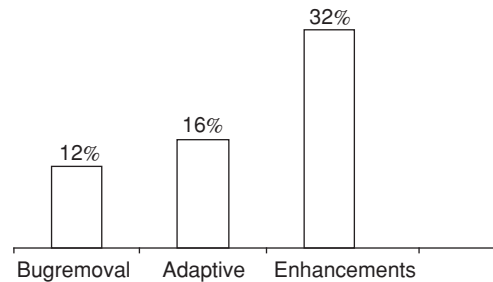
Design and testing plays major role in development, while coding is having least preference.

40% of the efforts were done on development and 60% of efforts are on the maintenance.

Distribution of the efforts on the development is shown below:



Maintenance includes removal of bug and corrective maintenance, adaptive maintenance and enhancement. Distribution of efforts in maintenance is shown below:



## EXERCISES

### Practice Problems I

**Directions for questions 1 to 15:** Select the correct alternative from the given choices.

- Which of the following statements is true?
  - The first step to the system study project is to announce the study project.
  - During the system study analysis determine manager's information needs by asking questions.
  - During the system study, flowcharts are drawn using general symbols.
  - All the above
- Which of the following statement(s) is true regarding the spiral model of software development?
  - In the spiral model of software development, the primary determinant in selecting activities in each interaction is risk.
  - The spiral model is a risk driven process model generator that is to guide multi-stakeholder. Concurrent engineering of software intensive systems.
  - Using the spiral model, software is developed in a series of evolutionary releases.
  - All the above
- Which of the following is a step in feasibility analysis?
  - Form a project team and appoint a project head.
  - Determine and evaluate performance and cost effectiveness of each candidate system.
  - Weigh system performance and cost data.
  - All the above
- Which of the following statement(s) is true?
  - The risk driven nature of the spiral model allows it to accommodate any mixture of specification oriented or some other approach.
  - Each cycle of spiral is completed by review which covers all the products developed during that cycle, including plans for the next cycle.
  - Spiral model works for development as well as enhancement project.
  - All the above
- Data flow diagram, regular expression and transition table can be combined to provide
  - decision table for functional specification of system software.
  - finite state automata for functional specification of system software.
  - event table for functional specification of system software.
  - None of these
- Which of the following statements are true about software configuration management tool?
  - It keeps track of the schedule based on the milestones reached.
  - It manages man power distribution by changing the structure of the project.
  - It maintains different versions of the configurable items.
  - All the above
- The cost incurred on a project was ₹250,000 and benefits were ₹30,000 per month. The payback period using simple pay back method is
 

(A) 8 months	(B) 8.3 months
(C) 12 months	(D) 1.2 months
- Which of the following phase has the maximum effort distribution?
 

(A) Testing	(B) Information gathering
(C) Requirement analysis	(D) Coding
- Which of the following statement is true regarding cost benefit analysis?
  - It evaluates tangible and non-tangible factors.
  - It estimates the hardware and software costs.
  - It compares the cost with the benefits of introducing a computer-based system.
  - All the statements are true.
- A project is considered economically feasible if the following factor holds good.
  - Return on investment (ROI)
  - Total cost of ownership (TCO)



- (C) Gross domestic product (GDP)  
(D) Net present value (NPV)

11. At the end of the feasibility study the system analyst  
(A) meets the users for a discussion.  
(B) gives system proposal to management.  
(C) gives a feasibility report to management.  
(D) gives a software requirement specification (SRS).
12. In a data flow diagram, data flows cannot take place between  
(A) two data stores  
(B) two external entities  
(C) a data store and an external entity  
(D) Both (A) and (B)
13. Consider the decision table shown below. It is

	R <sub>1</sub>	R <sub>2</sub>	R <sub>3</sub>	R <sub>4</sub>
C1	Y	Y	Y	N
C2	Y	N	Y	Y
C3			Y	Y
A1	X		X	
A2		X		X

- (A) an ambiguous decision table.  
(B) a complete decision table.  
(C) an incomplete decision table.  
(D) Both (A) and (B)
14. Which of the following requirement specifications can be validated?  
(S1): If the system fails during any operation, there should not be any loss of data.  
(S2): Checking the hardware compatibility.  
(S3): Defining a data interface.  
(S4): Specification of response time for various functions.  
(A) S1 and S2  
(B) S2, S3 and S4  
(C) S1, S3 and S4  
(D) S1 and S4
15. Which of the following are true?  
(i) A DFD should have loops.  
(ii) A DFD should not have crossing lines.  
(iii) Leveled DFD is easier to understand.  
(iv) Context diagrams are not used in DFDs.  
(A) (ii) and (i)  
(B) (i) and (iv)  
(C) (ii) and (iii)  
(D) (iii) and (iv)

## Practice Problems 2

**Directions for questions 1 to 15:** Select the correct alternative from the given choices.

- Questionnaire consists of  
(A) Forms  
(B) Documents  
(C) Qualitative data  
(D) Quantitative data
- The method to obtain qualitative information is  
(A) Background information  
(B) Questionnaires  
(C) Interviewing technique  
(D) Journals and reports on similar systems
- Which among the following is a functional requirement?  
(A) Description of all input data and their sources  
(B) Capacity requirements  
(C) Operating system available on the system  
(D) Maintaining a log of activities
- The advantage of use case during requirement analysis phase is, it  
(A) focuses on external behaviour only.  
(B) focuses on internal behaviour only.  
(C) focuses on additional behaviour.  
(D) focuses on internal and external behaviour.
- Operational feasibility refers to  
(A) technology needed is available and if available whether it is usable  
(B) the proposed solution can fit in with existing operations  
(C) the money spent is recovered by savings  
(D) superior quality of products
- Software engineering is the application of  
(A) Systematic approach of the development  
(B) Quantifiable approach of the development  
(C) Discipline approach of the development  
(D) All of these
- The data flow model of an application mainly shows:  
(A) The underlying data and the relationship among them  
(B) Processing requirement and the flow of data  
(C) Decision and control information  
(D) Communication network structure
- DFD completeness is  
(A) The process of discovering discrepancies between two or more sets of DFDs or discrepancies within a single DFD.  
(B) The extent to which all necessary components of a data flow diagram have been included and fully decomposed.  
(C) The conversation of inputs and outputs to a DFD process when that process is decomposed to a lower level.  
(D) An iterative process of breaking the description of a system down into a finer and finer details, which creates a set of charts in which one on a given chart is explained in greater detail on another chart.
- The requirement analysis is performed in  
(A) System design phase  
(B) System development phase  
(C) System analysis phase  
(D) System investigation phase

10. In data flow diagram, an originator or receiver of data is usually designed by  
 (A) square box (B) circle  
 (C) rectangle (D) arrow
11. A feasibility document should contain all the following except  
 (A) project name  
 (B) problem description  
 (C) feasible alternative  
 (D) data flow diagrams
12. SRS document is \_\_\_\_\_ between customers and developers.  
 (A) legal contract  
 (B) standard  
 (C) request proposal  
 (D) None of the above
13. According to Brooks, adding more people to an already late software project makes it  
 (A) late  
 (B) fast  
 (C) does not impact schedule  
 (D) None of the above
14. The following is a quality metric:  
 (A) Correctness  
 (B) Maintainability  
 (C) Usability  
 (D) All of the above
15. Feasibility study should focus on  
 (A) Technical feasibility  
 (B) Economic feasibility  
 (C) Operational feasibility  
 (D) All of the above

### PREVIOUS YEARS' QUESTIONS

1. What is the appropriate pairing of items in the two columns listing various activities encountered in a software life cycle? [2010]

P	Requirements capture	1	Module development and integration
Q	Design	2	Domain analysis
R	Implementation	3	Structural and behavioural modelling
S	Maintenance	4	Performance tuning

- (A) P-3, Q-2, R-4, S-1  
 (B) P-2, Q-3, R-1, S-4  
 (C) P-3, Q-2, R-1, S-4  
 (D) P-2, Q-3, R-4, S-1

2. Which one of the following is NOT desired in a good SRS document? [2011]  
 (A) Functional requirements  
 (B) Non-functional requirements  
 (C) Goals of implementation  
 (D) Algorithms for software implementation

### ANSWER KEYS

#### EXERCISES

##### Practice Problems 1

1. D    2. D    3. D    4. D    5. B    6. C    7. B    8. A    9. B    10. A  
 11. C    12. D    13. C    14. B    15. C

##### Practice Problems 2

1. D    2. C    3. A    4. A    5. B    6. D    7. B    8. B    9. C    10. A  
 11. D    12. A    13. A    14. D    15. D

##### Previous Years' Questions

1. B    2. D