1 Introduction

Q.5.3.1.1 State important characteristics of a software.
Answer: A software has following important characteristics:
∗ Software is engineered, not manufactured.
∗ Software does not wear out.
∗ Most of the softwares are custom built rather than being assembled from components.

Q.5.3.1.2 Explain the role of system analyst in traditional business.
Answer: A systems analyst is an information technology (IT) professional who specializes in analyzing, designing and implementing information systems. Systems analysts assess the suitability of information systems in terms of their intended outcomes and liaise with end users, software vendors and programmers in order to achieve these outcomes (Wikipedia).
A systems analyst may underate the following tasks:
→ Identify, understand and plan for organizational and human impacts of planned systems, and ensure that new technical requirements are properly integrated with existing processes and skill sets.
→ Plan a system flow from the ground up.
Interact with internal users and customers to learn and document requirements that are then used to produce business requirements documents.
→ Write technical requirements from a critical phase.
→ Interact with software architect to understand software limitations.
→ Help programmers during system development by providing use cases, flowcharts, UML diagrams and other concepts.
→ Document requirements or contribute to user manuals.
→ Whenever a development process is conducted, the system analyst is responsible for designing components and providing that information to the developer.

Q.5.3.1.3 Classify software into different categories.
Answer: There are various ways one could classify software. Based on nature of application of a software, some important categories are application software, embedded software, web application, artificial intelligence software and system software.

Q.5.3.1.4 What are the factors to be considered in the system model construction?
Answer: Some important factors that are to be considered for system modeling are assumption, simplification, limitation, preferences and constraints.

Q.5.3.1.5 What is an embedded system?
Answer: An embedded system is a computer system with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints. It is embedded as part of a complete device often including hardware and mechanical parts. Today, embedded systems control many devices that are in common use.

Q.5.3.1.6 Which of the following are the fundamental process activities?
i. software development, ii. software validation, iii. software evolution, iv. software specification , v. all of these
Answer: v

Q.5.3.1.7 Define software process.
Answer: Software process is defined as the structured set of activities that are required to develop a software system.

Q.5.3.1.8 —– are software systems which are designed to support routine activities in the software process such as editing and designing diagram, checking diagram consistency and keeping track of program tests which have been running.
Answer: CASE tools

Q.5.3.1.9 Discuss essential properties of a system.
Answer: There are two essential system properties:
Functional property: Functional properties that appear when all the parts of a system work together to achieve certain objectives.
Non-Functional property: Non-functional properties such as reliability, performance, safety and security. These relate to the behaviour of the system in its operational environment.

Q.5.3.1.10 Present a comparative study between ISO 9000 and CMM process model standards.
Answer: ISO stands for international standard organization. CMM

stands for capability maturity model. Major differences between these two quality standards are given below:
i. ISO 9000 applies to any type of industry. CMM is specially developed for software industry.
ii. ISO 9000 addresses corporate business process. CMM focuses on the software engineering activities.
iii. ISO 9000 specifies minimum requirement. CMM gets nto technical aspect of software engineering.
iv. ISO 9000 provides pass or fail criteria. CMM provides grade for process maturity. CMM has 5 levels: initial, repeatable, defined, managed and optimization.
v. ISO 9000 does not specify sequence of steps required to establish a quality system. CMM reconnects the mechanism for step by step progress through its successive maturity levels.

Q.5.3.1.11 Discuss major challenges faced in engineering a software.
Answer: Some key challenges are discussed here.
Heterogeneity: A new software may have to interact with different types of systems. Developing different interfaces with different softwares is a major issue.
Delivery: The time of delivery of a complete software is a critical issue. Also, it needs to confirm all quality requirements. It is a challenge of shortening delivery times for large and complex systems without compromising system quality.
Trust: People need to believe that the software is working fine. We need to develop techniques that demonstrate that software can be trusted by its users.

Q.5.3.1.12 Identify attributes of a good software.
Answer: The specific attributes that a good software will have depend on the application. Some general characteristics of a good software are mentioned below:
Maintainability: Software should be developed in such a way that the maintenance of the software bs to be easy. Software should be able to adopt with the changing needs of the customers.
Dependability: We should be able to trust the software in many senses such as reliability, security and safety. Software should not cause physical or economic damage in the event of system failure.
Efficiency: Efficient software shows good behaviour in terms of responsiveness, processing time, memory utilisation, and other relevant factors.

2 Different Systems

Q.5.3.2.1 How do you judge the reliability of a system?
Answer: The reliability of a system comes from the following influences:
Hardware reliability: The hardware components should be good enough in order to function the system for a long time.
Software reliability: How likely is it that a software component will produce an incorrect output? Software should be tested thoroughly to enhance reliability.
Operator reliability: Operator also is an important component in the entire system. Mistakes made by an operator may produce wrong result.

Q.5.3.2.2 What do you mean by system requirements?
Answer: Before we design the system, we need to understand the requirements of the proposed system. There are three main types of requirement.
1. *Abstract functional requirements*: It specifies the basic functions that the system must provide. All these functions are defined at an abstract level.
2. *System properties*: Here we specify non-functional emergent system properties such as availability, performance and safety.
3. *Characteristics that the system must not exhibit*: Sometimes it is required to specify what the system must not do.

Q.5.3.2.3 What is system design? Discuss steps involved in it.
Answer: The objective of system design is to provide functionalities in the system. It requires allocating tasks to the components. Inportant tasks to achieve the objectives are mentined below:
i. *Partitioning system*: One needs to analyse the requirements and organise them into groups.
ii. *Formation of sub-systems*: We identify sub-systems that can individually or collectively meet the requirements.
iii. *Assign requirements to sub-systems*: We map now requirements to the sub- systems. In many cases, there is no clean match between requirements partitions and identified sub-systems.
iv. *Specifying sub-system functionality*: We specify the specific functions provided by each sub-system.
v. *Defining interfaces*: We then define the interfaces that are provided

and required by each sub-system.

Q.5.3.2.4 What do mean by a legacy system?
Answer: A legacy system, in the context of computing, refers to outdated computer systems, programming languages or application software that are used instead of available upgraded versions.
Legacy systems also may be associated with terminology or processes that are no longer applicable to current contexts. Thus, it creates confusion. In theory, it would be great to be able to have an immediate access to use the most advanced technology. But in reality, most organizations have legacy systems to some extent. A legacy system may be problematic, due to compatibility issues, obsoletion or lack of security support.

Q.5.3.2.5 Present a layered model of a legacy system.
Answer: A layerd legacy system may be viewed as a 3-layer system, where the topmost layer consists of business processes. Application layer is kept in the middle. We place support software as the bottom layer.

Q.5.3.2.6 Discuss methods of improving reliability of a system.
Answer: We discuss here three approaches to enhance reliability of a system.
Fault avoidance: Some development techniques are used that either minimise the pos- sibility of mistakes and/or that trap mistakes before they result in the introduction of system faults.
Fault detection and removal: We apply verification and validation techniques that increase the chances that faults will be detected and removed before the system is used.
Fault tolerance: We use some techniques that ensure that faults in a system do not result in system errors or that ensure that system errors do not result in system failures.

Q.5.3.2.7 Elaborate the idea of systems engineering. Discuss different phases of systems engineering process.
Answer: Systems engineering is defined collectively as the activity of specifying, designing, implementing, validating, deploying and maintaining systems. Systems engineers are not just concerned with software development but also with hardware and the system's interactions with users and its environment.
Phases of software engineering are: i. Requirements definition, ii. System design, iii. System modelling, iv. Sub-system development, v. Sys-

tem integration, vi. System installation, vii. System evolution, viii. System decommissioning.
We provide here some ideas about each of these phases.
i. *Requirements definition*: It specifies what the system should do, i.e., its functions. It also includes its essential and desirable system properties.
ii. *System design*: It is concerned with how the system functionality is to be provided by the components of the system.
iii. *System modelling*: It refers to a set of components and relationships between these components.
iv. *Sub-system development*: The sub-systems identified during system design phase are implemented.
v. *System integration*: In this phase, the developed sub-systems are put together to make up a complete system.
vi. *System installation*: The developed system is deployed for use at the client's place.
vii. *System evolution*: During the life of a system, it is changed to correct errors in the original system requirements and to implement new requirements that have emerged.
viii. *System decommissioning*: The system is taken out of service after the end of its useful operational lifetime.

Q.5.3.2.8 What is system dependability? Discuss different dimensions of dependability.
Answer: Dependability of a system means systems trustworthiness. Trustworthiness essentially means the degree of user confidence that the system will operate as they expect and that the system will not fail in normal use. It has many dimensions such as availability, reliability, safety, and security. We elaborate these dimensions in the following.
Availability: It refers to the ability of the system to deliver services when requested.
Reliability: It means that the system is able to deliver services as specified.
Safety: It is the ability of the system to operate without catastrophic failure.
Security: The ability of the system to protect itself against accidental or deliberate intrusion.

3 Software Processes

Q.5.3.3.1 Which of the following is an example of software process model? i. A data flow or activity model, ii. A work flow model, iii. i and ii, iv. None of the above
Answer: iii

Q.5.3.3.2 Name some fundamental activities in software processes.
Answer: There are different types of software. Hence, there exist different processes. Most of the software processes have the following activities:
Software specification: The functionality of the software and constraints on its operation together define the software specification. We need to define it before engineering the software.
Software design and implementation: This is basically the production phase of a software.
Software validation: The software must be checked to ensure that it does what the customer wants. *Software evolution*: The software must be changed as the requirements of the customer change.

Q.5.3.3.3 Name a few software process models.
Answer: Some popular process models are mentioned below:
The waterfall model: This takes the fundamental process activities of specifi- cation, development, validation and evolution and represents them as separate proess phases such as requirements specification, software design, implementation, testing and so on.
Evolutionary development: This approach interleaves the activities of specification, development and validation. An initial system is rapidly developed from abstract specifications. This is then refined with customer input to produce a system that satisfies the customer's needs.
Component-based software engineering: This approach is based on the existence of a significant number of reusable components. The system development process focuses on integrating these components into a system rather than developing them from scratch.

Q.5.3.3.4 Give the reasons for the failure of water fall model.
Answer: Some reasons for the failure of water fall model are stated below:

- There are difficulties for customers to state all the requirements explicitly.
- Customers need more patience as the working product reaches only at the deployment phase.
- Real project rarely follow sequential flow. Iterations are made indirect manner.

Q.5.3.3.5 What are the advantages of water fall model?
Answer: Some advantages of water fall model are given below.
- It is simple.
- Lining up resources with appropriate skills becomes easy.

Q.5.3.3.6 Name a few evolutionary process models. Write some characteristics of prototyping model.
Answer: Some examples of evolutionary process models are prototyping model, spiral model, concurrent development model and incremental model.
Prototyping model is depicted by Fig. 3.

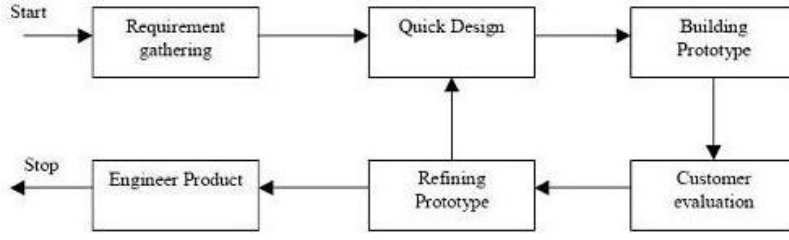

Figure 3: Prototyping model

Important characteristics of prototyping model are given here.
- Prototype is defined as first or preliminary form using available information.
- Prototype model is a set of general objectives for software.
- It does not identify the requirements like detailed input, output.
- It is a software engineering model of limited functionality.
- In this model, working programs are quickly produced.

Q.5.3.3.7 Compare and contrast spiral model and concurrent development model.
Answer: Spiral model (see Fig. 4) has the following characteristics.

- Spiral model is a risk driven process model.
- It is used for generating the software projects.
- In spiral model, an alternate solution is provided if the risk is found in the risk analysis.
- It is a combination of prototype and sequential model or waterfall model.
- In one iteration all activities are done for a large project.

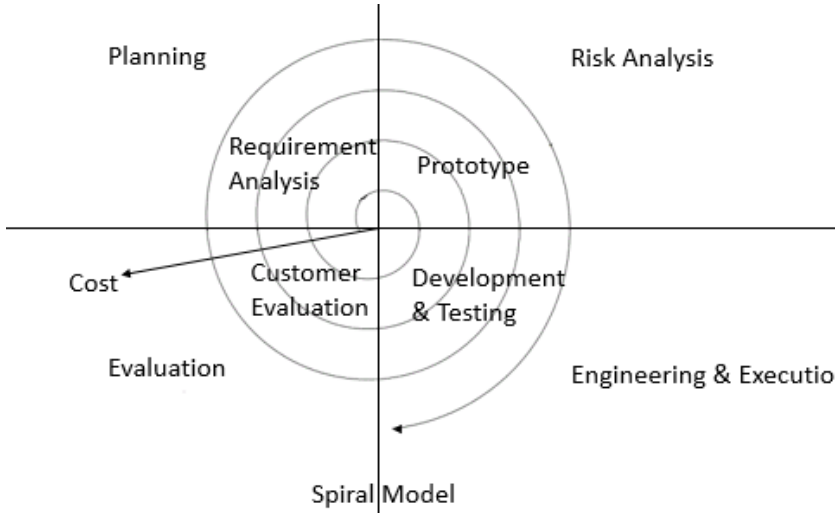

Figure 4: Spiral model

Concurrent development model (see Fig. 5) comes with the following characteristics.
- The concurrent development model, sometimes called concurrent engineering.
- It allows a software team to represent iterative and concurrent elements of any of the process model.