

①

65303

(2½ hours)

Total Marks: 75

- N. B.: (1) All questions are compulsory.  
 (2) Make suitable assumptions wherever necessary and state the assumptions made.  
 (3) Answers to the same question must be written together.  
 (4) Numbers to the right indicate marks.  
 (5) Draw neat labeled diagrams wherever necessary.  
 (6) Use of Non-programmable calculators is allowed.

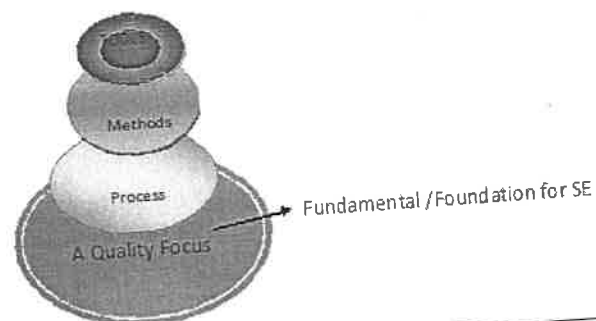
1.	Attempt <u>any three</u> of the following:	15
a.	<p>What is Software? Explain the characteristics of Software?</p> <p><b>Ans:</b>            Software system or software is:            (i) Instructions/Computer programs that when executed provide desired function &amp; performance.            (ii) Data structures that enable the programs to adequately manipulate information.            (iii) Documents that describe the operation and use of the programs.            A software system consists of several separate programs, configuration files, which are used to setup these programs, system documentation, which describes the structure of the system &amp; user documentation which explains how to use the system &amp; website for users to download recent product information.</p> <p><b>Characteristics:</b>            Since, software is a logical system (whereas, hardware is a physical system), its characteristics are different from hardware:            1. Software is developed or engineered; it is not manufactured in the classical sense.            2. Software doesn't "wear out".            3. Most of the software continues to be custom built or customizable.</p>	
b.	<p>Explain Software Development Life Cycle (SDLC) with the help of diagram</p> <p><b>Ans:</b>            It is a sequence of activities that leads to the production of a software product.  <b>Communication:</b> Very first step where user contact the service provider i.e, software organisation and initiate the request for a desired software product in writing and tries to negotiate the terms.  <b>Requirement Gathering:</b> A team of software developer holds discussion with various stakeholders from problem domain and bring out as much as information as possible on the requirements.  <b>Feasibility Study:</b> After requirement gathering, with the help of many algorithms, team analyses if a software can be designed to fulfill all requirements of the user and analyse if the product is financially, practically and technologically feasible for the organization to take up.  <b>System Analysis</b> - A Planning Phase: Developer decide a road map of their plan and try to bring up the best software model suitable for project.  <b>Software Design:-</b> All knowledge of requirement and analysis are taken together to plan-up design the software product.  <b>Coding:</b> Programming Phase- This step is also known as programming phase. The implementation of software design starts in terms of writing program code in the suitable programming language and developing error-free executable programming efficiently.  <b>Testing:</b> Software testing is done while coding, by the tester that is developing team members.</p>	

2

**Integration:** Software is integrated with libraries, databases and other programs.  
**Implementation:** We install software on user machine. Software is tested for portability, adaptability integration.  
**Operation And Maintenance:** This phase confirms the software operations In terms of more efficiency and less errors. If required, the user are trained on, are aided with the documentation on how to operate the software and how to keep the software operational

c. Define Software Engineering and its layer with the help of diagram.

**Ans:**  
 Software Engineering (SE) is the establishment & use of sound engineering principles to obtain economic, reliable and efficient software. Software Engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation & maintenance of software. Software Engineering is layered technology as shown in Figure 1.6.



d. Write a short note on any one:

### 1. RAD model

Rapid Action Development is an incremental software development process model that emphasizes an extremely short development cycle. The RAD model is a high-speed adaptation of the linear sequential model in which rapid development is achieved by using component-based construction. If requirements are well understood and project scope is constrained, the RAD model enables a development team to create a fully functional system within 60-90 days. Used primarily for information system applications, the RAD approach encompasses the following phases:

**Business modeling:** The information flow among business functions is modeled so as to understand the following:

- i) The information that drives the business process
- ii) The information generated
- iii) The source and destination of the information generated
- iv) The processes that affect this information

**Data modeling:** The information flow defined, as a part of the business-modeling phase is refined into a set of data objects that are needed to support the business. The attributes of each object are identified and the relationships between these objects are defined.

**Process modeling:** The data objects defined in the previous phase are transformed to achieve the information flow necessary to implement a business function. Processing descriptions are created for data manipulation.

**Application generation:** RAD assumes the use of fourth generation techniques. Rather than using third generation languages, the RAD process works to reuse existing programming components whenever possible or create reusable components.

In all cases, automated tools are used to facilitate construction.  
**Testing and turnover:** Since RAD emphasizes reuse, most of the components have already been tested. This reduces overall testing time. However, new components

3

must be tested and all interfaces must be fully exercised. In general, if a business function can be modularized in a way that enables each function to be completed in less than three months, it is a candidate for RAD. Each major function can be addressed by a separate RAD team and then integrated to form a whole.

#### Advantages:

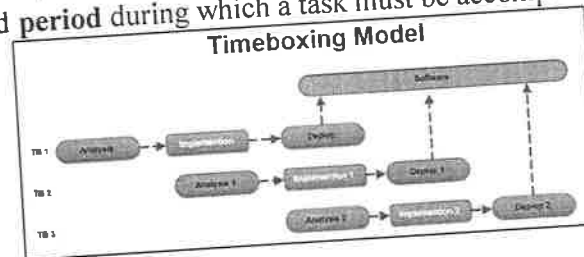
- Modularized approach to development
- Creation and use of reusable components
- Drastic reduction in development time

#### Disadvantages:

- For large projects, sufficient human resources are needed to create the right number of RAD teams.
- Not all types of applications are appropriate for RAD. If a system cannot be modularized, building the necessary components for RAD will be difficult.
- Not appropriate when the technical risks are high. For example, when an application makes heavy use of new technology or when the software requires a high degree of interoperability with existing programs.

## 2. TimeBox Model

Timeboxing is an approach to task and time management that sets rigid constraints on how long a given task or project can take to complete. Extensions are not permitted. The term comes from agile software development, in which a **time box** is defined **period** during which a task must be accomplished.



In time boxing model, development is done iteratively as in the iterative enhancement model. However, in time boxing model, each iteration is done in a timebox of fixed duration. The functionality to be developed is adjusted to fit the duration of the timebox. Moreover, each timebox is divided into a sequence of fixed stages where each stage performs a clearly defined task (analysis, implementation, and deploy) that can be done independently. This model also requires that the time duration of each stage is approximately equal so that pipelining concept is employed to have the reduction in development time and product releases. There is a dedicated team for each stage so that the work can be done in pipelining. Thus, stages should be chosen in such a way that each stage perform some logical unit of work that becomes the input for next stage.

In addition to the advantages of iterative model, time boxing model has some other advantages too.

Various advantages and disadvantages associated with timeboxing model are listed in Table.

Advantages	Disadvantages
Speeds up the development process and shortens the delivery time Well suited to develop projects with a number of features in short time period.	Project management becomes more complex. Not suited to projects in which entire development work cannot be divided into multiple iterations of almost, equal duration.



e.

What are Functional and non-functional requirements of software?

**Ans:**

Functional requirements are the statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations. Functional requirements are the product capabilities, or things that a product must do for its users. Functional requirements define how software behaves to meet user needs.

Non-Functional Requirements are the constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc. Non-Functional Requirements in Software Engineering presents a systematic approach to 'building quality into' software systems.

f.

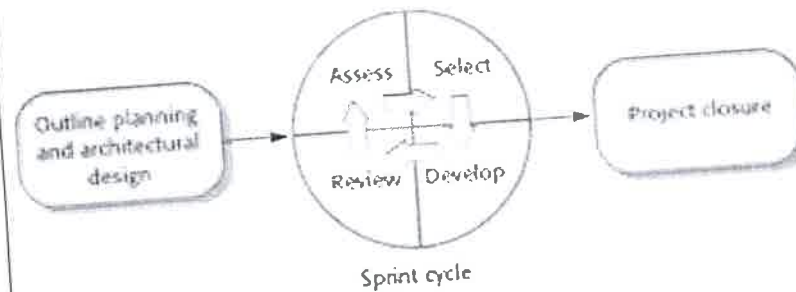
Explain the three phases in SCRUM for Agile Project Management.

**Ans:**

**Agile project management** requires a different approach, which is adapted to incremental development and the particular strengths of agile methods.

The Scrum approach is a general agile method but its **focus is on managing iterative development** rather than specific agile practices. There are three phases in Scrum.

- The **initial phase** is an **outline planning phase** where you establish the general objectives for the project and design the software architecture.
- This is followed by a **series of sprint cycles**, where each cycle **develops an increment** of the system.
- The project **closure phase** wraps up the project, completes required documentation such as system help frames and user manuals and assesses the lessons learned from the project.



### The Sprint cycle

- Sprints are **fixed length**, normally **2-4 weeks**. They correspond to the development of a release of the system in XP.
- The starting point for planning is the **product backlog**, which is the list of work to be done on the project.
- The **selection phase** involves all of the project team who work with the customer to select the features and functionality to be developed during the sprint.
  - ✧ Once these are agreed, the team organize themselves to develop the software. During this stage the team is isolated from the customer and the organization, with all communications channelled through the so-called 'Scrum master'.
  - ✧ The role of the Scrum master is to protect the development team from external distractions.
  - ✧ At the end of the sprint, the work done is reviewed and presented to stakeholders. The next sprint cycle then begins.

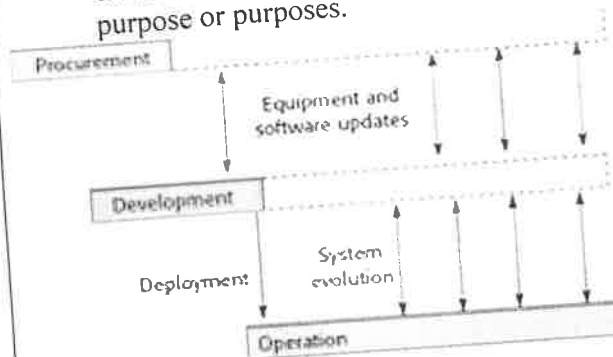
5

15

2. Attempt any three of the following:  
a. Describe the different stages of System Engineering process?

Ans:

- ✧ Procuring, specifying, designing, implementing, validating, deploying and maintaining socio-technical systems.
- ✧ Concerned with the services provided by the system, constraints on its construction and operation and the ways in which it is used to fulfil its purpose or purposes.



- ✧ Procurement (acquisition)
  - The purpose of the system is established, high-level system requirements are defined, decisions are made on how functionality is distributed and the system components are purchased.
- ✧ Development
  - The system is developed – requirements are defined in detail, the system is implemented and tested and operational processes are defined.
- ✧ Operation
  - The system is deployed and put into use. Changes are made as new requirements emerge. Eventually, the system is decommissioned.

b. Explain the Importance of System dependability and the causes of failure in system dependability.

Ans:

- ✧ For many computer-based systems, the most important system property is the dependability of the system.
- ✧ The dependability of a system reflects the user's degree of trust in that system. It reflects the extent of the user's confidence that it will operate as users expect and that it will not 'fail' in normal use.
- ✧ Dependability covers the related systems attributes of reliability, availability and security. These are all inter-dependent.

#### Importance of dependability

- ✧ System failures may have widespread effects with large numbers of people affected by the failure.
- ✧ Systems that are not dependable and are unreliable, unsafe or insecure may be rejected by their users.
- ✧ The costs of system failure may be very high if the failure leads to economic losses or physical damage.
- ✧ Undependable systems may cause information loss with a high consequent recovery cost.

#### Causes of failure

- ✧ Hardware failure
  - Hardware fails because of design and manufacturing errors or because components have reached the end of their natural life.

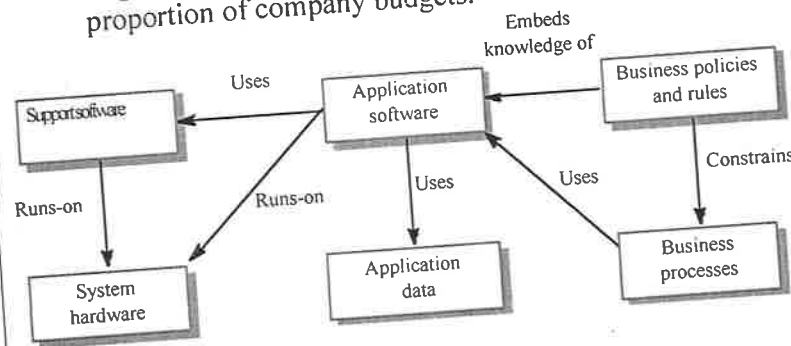
6

- ❖ Software failure
  - Software fails due to errors in its specification, design or implementation.
- ❖ Operational failure
  - Human operators make mistakes. Now perhaps the largest single cause of system failures in socio-technical systems.

c. Explain the legacy system in Socio Technical system that continues to provide essential services.

Ans:

- Socio-technical systems that have been developed using old or obsolete technology.
- Crucial to the operation of a business and it is often too risky to discard these systems
  - Bank customer accounting system;
  - Aircraft maintenance system.
- Legacy systems constrain new business processes and consume a high proportion of company budgets.



- Hardware - may be obsolete mainframe hardware.
- Support software - may rely on support software from suppliers who are no longer in business.
- Application software - may be written in obsolete programming languages.
- Application data - often incomplete and inconsistent.
- Business processes - may be constrained by software structure and functionality.
- Business policies and rules - may be implicit and embedded in the system software.

d. Explain the process or the steps of Requirements Engineering Briefly?

Ans:

It is a four step process, which includes –

- Feasibility Study
- Requirement Gathering
- Software Requirement Specification
- Software Requirement Validation

**Feasibility study**

- When the client approaches the organization for getting the desired product developed, it comes up with rough idea about what all functions the software must perform and which all features are expected from the software.

The analysts does a detailed study about whether the desired system and its functionality are feasible to develop.

7

- This feasibility study is focused towards goal of the organization. This study analyzes whether the software product can be practically materialized in terms of implementation, contribution of project to organization, cost constraints and as per values and objectives of the organization. It explores technical aspects of the project and product such as usability, maintainability, productivity and integration ability.

### Requirement Gathering

- If the feasibility report is positive towards undertaking the project, next phase starts with gathering requirements from the user.
- Analysts and engineers communicate with the client and end-users to know their ideas on what the software should provide and which features they want the software to include.

### Software Requirement Specification

- SRS is a document created by system analyst after the requirements are collected from various stakeholders.
- SRS defines how the intended software will interact with hardware, external interfaces, speed of operation, response time of system, portability of software across various platforms, maintainability, speed of recovery after crashing, Security, Quality, Limitations etc.
- The requirements received from client are written in natural language. It is the responsibility of system analyst to document the requirements in technical language so that they can be comprehended and useful by the software development team.

### Software Requirement Validation

After requirement specifications are developed, the requirements mentioned in this document are validated. User might ask for illegal, impractical solution or experts may interpret the requirements incorrectly. This results in huge increase in cost if not nipped in the bud.

Requirements can be checked against following conditions -

- If they can be practically implemented
- If they are valid and as per functionality and domain of software
- If there are any ambiguities
- If they are complete
- If they can be demonstrated

e. Explain USE Case diagram with  
**Online Shopping** : Web Customer actor uses some web site to make purchases online. Top levels USE CASE are **View Items, Make Purchase and Client Register**. View Items use case could be used by customer as top level use case if customer only wants to find and see some products. This use case could also be used as a part of Make Purchase use case. Client Register use case allows customer to register on the web site, for example to get some coupons or be invited to private sales.

f. Explain briefly Legacy system categories and its assessment with the help of example?

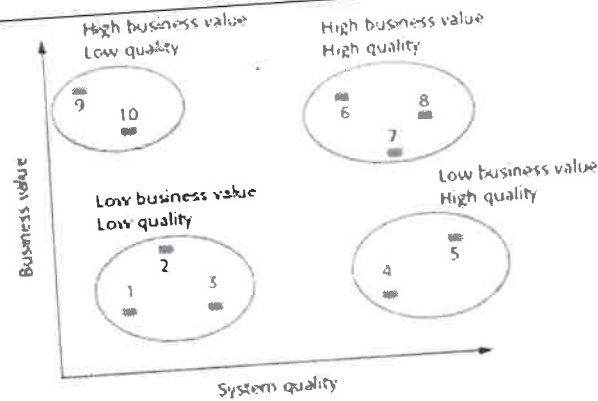
**Ans:**

- ✧ Organisations that rely on legacy systems must choose a strategy for evolving these systems
  - Scrap the system completely and modify business processes so that it is no longer required;
  - Continue maintaining the system
  - Transform the system by re-engineering to improve its maintainability;
  - Replace the system with a new system.
- ✧ The strategy chosen should depend on the system quality and its business value.

**An example of a legacy system assessment**



8



### Legacy system categories

- ✧ Low quality, low business value
  - These systems should be scrapped.
- ✧ Low-quality, high-business value
  - These make an important business contribution but are expensive to maintain. Should be re-engineered or replaced if a suitable system is available.
- ✧ High-quality, low-business value
  - Replace with COTS, scrap completely or maintain.
- ✧ High-quality, high business value
  - Continue in operation using normal system maintenance.

15

### 3. Attempt any three of the following:

a. Define Architectural design and explain the functions of architectural design?

Ans:

Architectural Design - The architectural design is the highest abstract version of the system. It identifies the software as a system with many components interacting with each other. At this level, the designers get the idea of proposed solution domain.

IEEE defines architectural design as 'the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system.'

An architectural design performs the following functions.

1. It defines an abstraction level at which the designers can specify the functional and performance behaviour of the system.
2. It acts as a guideline for enhancing the system (when ever required) by describing those features of the system that can be modified easily without affecting the system integrity.
3. It evaluates all top-level designs.
4. It develops and documents top-level design for the external and internal interfaces.
5. It develops preliminary versions of user documentation.
6. It defines and documents preliminary test requirements and the schedule for software integration. Architectural design is of crucial importance in software engineering during which the essential requirements like reliability, cost, and performance are dealt with. Though the architectural design is the responsibility of developers, some other people like user representatives, systems engineers, hardware engineers, and operations personnel are also involved. All these stakeholders must also be consulted while reviewing the architectural design in order to minimize the risks and errors.

b. Explain User Interface design process(UID)?

Ans:

User interfaces should be designed to match the skills, experience and expectations of its anticipated users. System users often judge a system by its interface rather than its functionality. A poorly designed interface can cause a user to make catastrophic



9

errors. Poor user interface design is the reason why so many software systems are never used.

User Interface design is an iterative process involving close liaisons between users and designers.

The three core activities in this process are:

- User analysis. Understand what the users will do with the system;
- System prototyping. Develop a series of prototypes for experiment;
- Interface evaluation. Experiment with these prototypes with users.

c. Explain Software Project Management briefly?

Ans:

Software project management is the mean by which an orderly control process can be imposed on the software development process to ensure software quality along with the monitoring of the process and then controlling the wrong activities. Software project management is a crucial difficult task in case where  
(i) software execution is late (ii) over budget (iii) software fails to meet the user requirements.

The basic project management process activities:

- 1- Proposal Writing
- 2- Project Planning and Scheduling
- 3- Project Costing
- 4- Project monitoring and reviews
- 5- Personal Selection and Evaluation
- 6- Report Writing and Presentations
- 7- Quality Management
- 8- Configuration Management

1-Proposal Writing:

It includes:

- Description of main objectives of the project
- How the objectives will be carried out and fulfilled
- Cost & schedule estimates

2- Project Planning and Scheduling: Planning includes:

- Identification of activities
- Milestones (reports ,manual) for the management
- Deliverables for the customer

Project scheduling includes:

- Division of project into separate activities
- Time Judgment for completion of each Activity

3- Project Costing :

It Includes:

- Estimation of the total cost of the projects

4- Project Monitoring and Reviews :

Monitoring is a continuing activity and includes:

- The progress of the project is compared regularly with the planned time schedule & costing (could be done with daily informal discussions or formal meetings

Review includes:

- The review of overall progress of the technical development of the project is done regularly.

5- Personal Selection Evaluation :

It includes:

- Selection of skilled and experienced staff for the project
- Regular evaluation of the performance of staff.

19

- Inexperienced staff may be trained.

6- Report writing Presentation :

- Report of project is briefly documented to present before the client & contractor.

7- Quality Management

It includes:

- Quality Assurance

- Quality planning

- Quality control

8- Configuration Management :

Configuration is a set of activities designed to control change by identifying the work products that are likely to change, establishing relationships among them, defining mechanisms for managing different version of these products, controlling & the changes imposed auditing & reporting on the changes made.

It includes:

- Identification of the work products.

- Managing the products and controlling, auditing, reporting the changes.

d. Briefly explain the Risk identification and the types of risk in the process of Risk Management?

Ans:

Risk identification is a systematic attempt to specify threats to the project plan. By identifying known and predictable risks, the project manager takes a first step towards avoiding them when possible and controlling them when necessary. There are two distinct types of risks:

i) **Generic risks:** These are a potential threat to every software project.

ii) **Product-specific risks:** These can be identified only by a clear understanding of the

technology, the people and the environment that are specific to the project at hand. One method to identify risks is to create a **risk item checklist**. The checklist can be used for risk identification and focuses on some subset of known and predictable risks in the following general categories:

☐ ☐ **Product size:** Risks associated with the overall size of the software to be built or modified.

☐ ☐ **Business impact:** Risks associated with constraints imposed by the management or the market.

☐ ☐ **Customer characteristics:** Risks associated with the sophistication of the customer and the developers ability to communicate with the customer in a timely manner.

☐ ☐ **Process definition:** Risks associated with the degree to which the software process has been defined.

☐ ☐ **Development environment:** Risks associated with the availability and the quality of the tools used to build the product.

☐ ☐ **Technology:** Risks associated with the complexity of the system to be built and the newness of the technology that is packaged by the system.

☐ ☐ **Staff size and experience:** Risks associated with the overall technical and project experience of the software engineers who will do the work.

The risk item checklist can be organized in different ways. Questions relevant to each of the topics can be answered for each software project. The answers to these questions allow the planner to estimate the impact of risk. A different risk item checklist format simply lists characteristics that are relevant to each general subcategory. Finally, a set of risk components and drivers are listed along with their

4

probability of occurrence.

**Risk components** are defined in the following manner:

□□ **Performance risk** - The degree of uncertainty that the product will meet its requirement and be fit for its intended use.

□□ **Cost risk** - The degree of uncertainty that the project budget will be maintained

□□ **Support risk** - The degree of uncertainty that the resultant software will be easy to correct, adapt and enhance.

□□ **Schedule risk** - The degree of uncertainty that the project schedule will be maintained and the product will be delivered on time.

e.

Explain the functions of Quality Assurance and its Standards?

**Ans:**

Quality assurance is establishing organizational procedures and standards for quality. Software Quality Assurance (SQA) Activities:

- Prepares an SQA plan for a project
- Participates in the development of the project and software process description
- Reviews software engineering activities to verify compliance with the defined software process
- Audits designated software work products to verify compliance with those defined as part of the software process
- Ensures that deviations in software work and work products are documented and handled according to a documented procedure
- Records any noncompliance and reports to senior management
- Coordinates the control and management of change
- Helps to collect and analyze software metrics

Standards:

- Standards are the key to effective quality management
- They may be international, national, organizational or project standards
- Product standards define characteristics that all components should exhibit e.g. a common programming style
- Process standards define how the software process should be enacted Importance of standards.
- Encapsulation of best practice- avoids repetition of past mistakes
- Framework for quality assurance process - it involves checking standard compliance
- Provide continuity - new staff can understand the organisation by understand the standards applied

Certification Standards:

1. ISO 9000
2. ISO 9001

f.

Describe why it is important to measure the software metrics?

**Ans:**

Software measurement:

Software measurement is concerned with deriving a numeric value for an attribute of a software product or process. This allows for objective comparisons between techniques and processes

Metrics:

- Any type of measurement which relates to a software system, process or related documentation
- Allow the software and the software process to be quantified
- Measures of the software process or product
- May be used to predict product attributes or to control the software process

**Use of measurements**



02

- ✧ To assign a value to system quality attributes
  - By measuring the characteristics of system components, such as their cyclomatic complexity, and then aggregating these measurements, you can assess system quality attributes, such as maintainability.
- ✧ To identify the system components whose quality is sub-standard
 

Measurements can identify individual components with characteristics that deviate from the norm. For example, you can measure components to discover those with the highest complexity. These are most likely to contain bugs because the complexity makes them harder to understand

15

4. Attempt any three of the following:

a. Explain System testing Process.

**Ans:**

System Testing is the assurance of proper working or execution of the developed system from user's perspective using formal procedure.

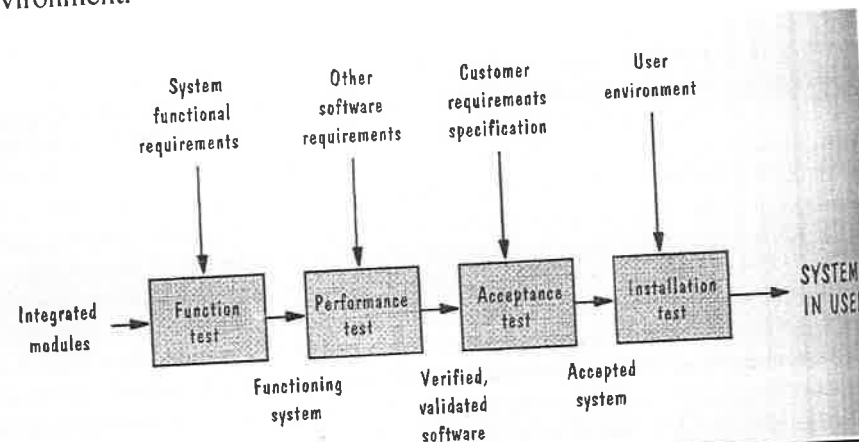
System Testing Process can be defined in the following steps:

a. **Function Testing:** the system must perform functions specified in the requirements.

b. **Performance Testing:** the system must satisfy security, precision, load and speed constraints specified in the requirements.

c. **Acceptance Testing:** customers try the system to make sure that the system built is the system they had requested.

d. **Installation Testing:** the software is deployed and tested in the production environment.



**System testing:-**

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.

b. Explain briefly Verification and Validation (V & V) Process?

**Ans:**

Verification and Validation Verification is a process of evaluating the intermediary work products of a software development lifecycle to check if we are in the right track of creating the final product. Validation is the process of evaluating the final product to check whether the software meets the business needs.

Verification Validation Evaluates the intermediary products to check whether it meets the specific requirements of the phase. Evaluates the final product to check whether it meets the business needs. Checks whether the product is built as per the specified requirement and design specification. It determines whether the software is fit for use and satisfies the business need. Checks —Are we building the product

13

	<p>right? Checks —Are we building the right product? This is done without executing the software. Is done with executing the software. Involves all the static testing techniques. Includes all the dynamic testing techniques.</p> <p>The objectives of Verification and Validation activities are as follows:</p> <ul style="list-style-type: none"> <li>• Facilitates early detection and correction of errors.</li> <li>• Encourages and enhances the management intervention and inside into process and product risks.</li> <li>• Provide supportive measures towards the software lifecycle process, to enhance compliance with schedule and budget requirements.</li> </ul>	
c.	<p>List and describe the static Analysis check points involved in Automated Static Analysis?</p> <p><b>Ans:</b> Automated static analysis involves the static analyzers. Static analyzers are software tools for source text processing. They parse the program text and try to discover potentially erroneous conditions and bring these to the attention of the Verification and Validation team. They are very effective and helpful for inspections.</p> <p>Static testing is a form of software testing where the software isn't used. This contrasts with dynamic testing. It is generally not detailed testing, but checks mainly for the sanity of the code, algorithm, or document. It is primarily syntax checking of the code and/or manually reviewing the code or document to find errors. This type of testing can be used by the developer who wrote the code, in isolation. Code reviews, inspections and walkthroughs are also used.</p> <p>From the black box testing point of view, static testing involves reviewing requirements and specifications. This is done with an eye toward completeness or appropriateness for the task at hand. This is the verification portion of Verification and Validation.</p> <p>Even static testing can be automated. A static testing test suite consists of programs to be analyzed by an interpreter or a compiler that asserts the programs syntactic validity.</p>	
d.	<p>Write a short note on Size oriented Metrics of Software Measurement and Find the effort for the project, assume that 310 FP are estimated in total, and average productivity based on past projects is 5.5 FP/person-month.</p> <p><b>Ans:</b> Size-oriented measures are computed by normalizing direct measures of the software engineering process (e.g. effort or defects) over the product size, measured in lines of code. Size-oriented metrics are relatively easy to collect, but can present problems when component-based or visual programming methods are applied.</p> <p><b>Size-oriented Metrics</b> Attempt to quantify software projects by using the size of the project to normalize other quality measures</p> <p>Possible data to collect:</p> <ul style="list-style-type: none"> <li>number of lines of code</li> <li>number of person-months to complete</li> <li>cost of the project</li> <li>number of pages of documentation</li> <li>number of errors corrected before release</li> <li>number of bugs found post release</li> </ul> <p>Assume that 310 FP are estimated in total, and average productivity based on past projects is 5.5 FP/person-month, then the effort for the project is: <b>Effort = 310/5 = 56 person-months</b></p>	
e.	Explain any one type of metrics to estimate the Software productivity?	

(14)

- Function Points
- Object Point

Function points :

- Function Points measure software size by quantifying the functionality provided to the user based solely on logical design and functional specifications
- Function point analysis is a method of quantifying the size and complexity of a software system in terms of the functions that the system delivers to the user
- It is independent of the computer language, development methodology, technology or capability of the project team used to develop the application

Working from the project design specifications, the following system functions are measured (counted):

- Inputs
- Outputs
- Files
- Inquires
- Interfaces

Steps involved:

Determine Type of Count (3 Types)

- Enhancement (Project) Function Point Count
- Application Function Point Count • Development (Project) Function Point Count
- Identify Counting Scope and Application Boundary
- Count Data Functions
- Count Transactional Functions
- Determine Unadjusted Function Point Count
- Determine Value Adjustment Factor
- Calculate Adjusted Function Point Count

Object points:

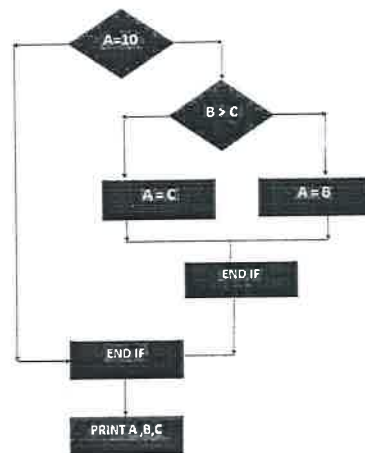
- Object points are an alternative function-related measure to function points when 4GLs or similar languages are used for development
- Object points are NOT the same as object classes
- The number of object points in a program is a weighted estimate of
  - The number of separate screens that are displayed
  - The number of reports that are produced by the system
  - The number of 3GL modules that must be developed to supplement the 4GL code
- Object points are easier to estimate from a specification than function points as they are simply concerned with screens, reports and 3GL modules
- They can therefore be estimated at an early point in the development process. At this stage, it is very difficult to estimate the number of lines of code in a system

f. Calculate Cyclomatic complexity using the control flow diagram for the given example:  
IF A = 10 THEN  
IF B > C THEN  
A = B  
ELSE  
A = C  
ENDIF



15 >

ENDIF  
Print A  
Print B  
Print C  
Ans:



The Cyclomatic complexity is calculated using the above control flow diagram that shows seven nodes(shapes) and eight edges (lines), hence the cyclomatic complexity is  $8 - 7 + 2 = 3$

5. Attempt any three of the following:

15

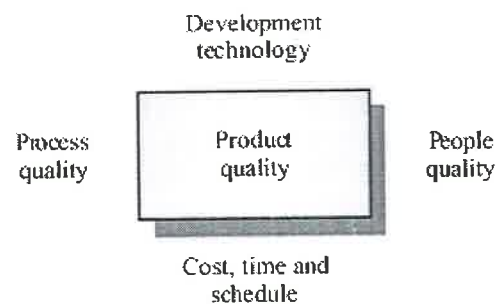
a. Explain process and product quality.

Ans:

Understanding, Modelling and Improving the Software Process is process improvement.

- Process quality and product quality are closely related
- A good process is usually required to produce a good product
- For manufactured goods, process is the principal quality determinant
- For design-based activity, other factors are also involved especially the capabilities of the designers

Principal product quality factors are as shown in fig



- For large projects with average capabilities, the development process determines product quality
- For small projects, the capabilities of the developers is the main determinant
- The development technology is particularly significant for small projects
- In all cases, if an unrealistic schedule is imposed then product quality will suffer

b. Explain the different levels of CMMI (Capability Maturity Model introduced) Framework?

	<p>Ans</p> <p>The CMMI framework is the current stage of work on process assessment and improvement that started at the Software Engineering Institute(SEI) in the 1980s.</p> <ul style="list-style-type: none"> <li>✧ Initial             <ul style="list-style-type: none"> <li>▪ Essentially uncontrolled</li> </ul> </li> <li>✧ Repeatable             <ul style="list-style-type: none"> <li>▪ Product management procedures defined and used</li> </ul> </li> <li>✧ Defined             <ul style="list-style-type: none"> <li>▪ Process management procedures and strategies defined and used</li> </ul> </li> <li>✧ Managed             <ul style="list-style-type: none"> <li>▪ Quality management strategies defined and used</li> </ul> </li> <li>✧ Optimising             <ul style="list-style-type: none"> <li>▪ Process improvement strategies defined and used</li> </ul> </li> <li>✧ Intended as a means to assess the extent to which an organisation's processes follow best practice.</li> <li>✧ By providing a means for assessment, it is possible to identify areas of weakness for process improvement.</li> <li>✧ There have been various process assessment and improvement models but the SEI work has been most influential.</li> </ul>	
c.	<p>Explain briefly WSDL (Web Service Description Language).</p> <p>Ans:</p> <ul style="list-style-type: none"> <li>✧ The service interface is defined in a service description expressed in WSDL (Web Service Description Language).</li> <li>✧ The WSDL specification defines             <ul style="list-style-type: none"> <li>▪ What operations the service supports and the format of the messages that are sent and received by the service</li> <li>▪ How the service is accessed - that is, the binding maps the abstract interface onto a concrete set of protocols</li> <li>▪ Where the service is located. This is usually expressed as a URI (Universal Resource Identifier)</li> </ul> </li> </ul> <div data-bbox="665 1193 1218 1433" data-label="Diagram"> <p>WSDL service definition</p> <ul style="list-style-type: none"> <li>Intro             <ul style="list-style-type: none"> <li>▪ XML namespace declarations</li> </ul> </li> <li>Abstract interface             <ul style="list-style-type: none"> <li>▪ Type declarations</li> <li>▪ Interface declarations</li> <li>▪ Message declarations</li> </ul> </li> <li>Concrete implementation             <ul style="list-style-type: none"> <li>▪ Binding declarations</li> <li>▪ Endpoint declarations</li> </ul> </li> </ul> <ul style="list-style-type: none"> <li>✧ The 'what' part of a WSDL document, called an interface, specifies what operations the service supports, and defines the format of the messages that are sent and received by the service.</li> <li>✧ The 'how' part of a WSDL document, called a binding, maps the abstract interface to a concrete set of protocols. The binding specifies the technical details of how to communicate with a Web service.</li> <li>✧ The 'where' part of a WSDL document describes the location of a specific Web service implementation (its endpoint).</li> </ul> </div>	
d.	<p>What are the benefit and problem of reusing Software?</p> <ul style="list-style-type: none"> <li>✧ In most engineering disciplines, systems are designed by composing existing components that have been used in other systems.</li> <li>✧ Software engineering has been more focused on original development but it is now recognised that to achieve better software, more quickly and at lower</li> </ul>	

17

cost, we need a design process that is based on systematic software reuse.

- ✧ There has been a major switch to reuse-based development over the past 10 years.

#### Benefits of software reuse

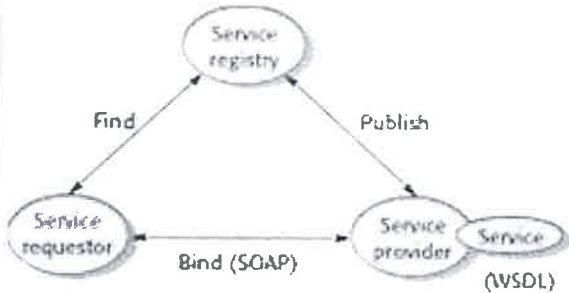
Benefit	Explanation
Increased dependability	Reused software, which has been tried and tested in working systems, should be more dependable than new software. Its design and implementation faults should have been found and fixed.
Reduced process risk	The cost of existing software is already known, whereas the costs of development are always a matter of judgment. This is an important factor for project management because it reduces the margin of error in project cost estimation. This is particularly true when relatively large software components such as subsystems are reused.
Effective use of specialists	Instead of doing the same work over and over again, application specialists can develop reusable software that encapsulates their knowledge.
Standards compliance	Some standards, such as user interface standards, can be implemented as a set of reusable components. For example, if menus in a user interface are implemented using reusable components, all applications present the same menu formats to users. The use of standard user interfaces improves dependability because users make fewer mistakes when presented with a familiar interface.
Accelerated development	Bringing a system to market as early as possible is often more important than overall development costs. Reusing software can speed up system production because both development and validation time may be reduced.

#### Problems with reuse

Problem	Explanation
Increased maintenance costs	If the source code of a reused software system or component is not available then maintenance costs may be higher because the reused elements of the system may become increasingly incompatible with system changes.
Lack of tool support	Some software tools do not support development with reuse. It may be difficult or impossible to integrate these tools with a component library system. The software process assumed by these tools may not take reuse into account. This is particularly true for tools that support



18

	<p>embedded systems engineering, less so for object-oriented development tools.</p> <p><b>Not-invented-here syndrome</b></p> <p>Some software engineers prefer to rewrite components because they believe they can improve on them. This is partly to do with trust and partly to do with the fact that writing original software is seen as more challenging than reusing other people's software.</p> <p><b>Creating, maintaining, and using a component library</b></p> <p>Populating a reusable component library and ensuring the software developers can use this library can be expensive. Development processes have to be adapted to ensure that the library is used.</p> <p><b>Finding, understanding, and adapting reusable components</b></p> <p>Software components have to be discovered in a library, understood and, sometimes, adapted to work in a new environment. Engineers must be reasonably confident of finding a component in the library before they include a component search as part of their normal development process.</p>	
e.	<p><b>Briefly describe the concept of SOA (Service Oriented Architecture) and the benefits of SOA?</b></p> <ul style="list-style-type: none"> <li>✧ A means of developing distributed systems where the components are stand-alone services</li> <li>✧ Services may execute on different computers from different service providers</li> <li>✧ Standard protocols have been developed to support service communication and information exchange</li> </ul>  <pre> graph TD     SR((Service registry))     SReq((Service requestor))     SProv((Service provider))     S((Service WSDL))     SReq -- Find --&gt; SR     SProv -- Publish --&gt; SR     SReq -- "Bind (SOAP)" --&gt; SProv     SProv --- S   </pre> <p><b>Benefits of SOA</b></p> <ul style="list-style-type: none"> <li>✧ Services can be provided locally or outsourced to external providers</li> <li>✧ Services are language-independent</li> <li>✧ Investment in legacy systems can be preserved</li> <li>✧ Inter-organisational computing is facilitated through simplified information exchange</li> </ul>	
f.	<p><b>Write a short note on SaaS(Software as a service)?</b></p> <ul style="list-style-type: none"> <li>✧ Software as a service (SaaS) involves hosting the software remotely and providing access to it over the Internet. <ul style="list-style-type: none"> <li>▪ Software is deployed on a server (or more commonly a number of servers) and is accessed through a web browser. It is not deployed on a local PC.</li> <li>▪ The software is owned and managed by a software provider, rather than the organizations using the software.</li> <li>▪ Users may pay for the software according to the amount of use they make of it or through an annual or monthly subscription.</li> </ul> </li> </ul>	

19

	<b>Benefits of SaaS</b> Save time Save money Cost effective scalable	
--	--	--