II. B.Tech II – Semester (R22) Supply December 2025

Software Engineering (R22CS405PC)

1. a) what are the merits of Incremental model?

Ans:- Early delivery of working software
- Accomodates changing requirements
- Reduced development risk
- Easier testing and maintenance
- Continuous user feedback.

b) what are the advantages of unified process?

Ans:
- Iterative and incremental development
- Risk-focused development
- Architecture-centric
- Use case driven
- supports requirement changes
- High product quality.

c) write the purpose of Context model.

Ans: The purpose of a Context model is to define the system boundary, identify external entities, and show interactions between the system and its environment.

d) what is the significance of feasibility study?

Ans: The significance of a feasibility study lies in determining whether a software project is technically, economically, operationally and legally viable, thereby reducing risk and supporting informed decision-making.

e) what is the use of interface analysis?

Ans: The use of interface analysis is to identify and define how a software system interacts with users, external systems and internal components, ensuring clear communication, better usability and easier integration.

f) List the guidelines for data design.

Ans:
- Data abstraction
- Normalization
- Data integrity
- Efficient access
- Data independence
- Security and scalability
- Clear relationships.

g) Differentiate between verification and validation.

Ans:

verification	validation
> verification is the process to find whether the s/w meets the specified requirements for particular phase	> process is checked whether the software meets requirements and expectation of the customer.
> It estimates an intermediate product	> It estimates the final product.
> It describes whether the outputs are as per the inputs or not	> It explains whether they are accepted by the user or not.
> verification is done before the validation	> It is done after the verification.
> plans, requirement, specification code are evaluated during the verification	> Actual product or software is tested under validation.

h) what is meant by debugging?

Ans: Debugging is the systematic process of finding and removing defects or bugs from a software program to ensure Correct Operation.

i) what is meant by software Reliability?

Ans: Software reliability is the probability that a software system will operate without failure for a specified

period of time under specified conditions. It indicates how dependable and error-free a software product is during execution. A highly reliable system performs its intended functions consistently without crashes or incorrect results.

j) what is the importance of software reviews?

Ans: software reviews are a systematic examination of software artifacts (Requirements, design, code, test cases) to detect defects early and improve software quality.

- Early Detection of Defects
- Reduces development cost
- Improves software Quality
- Ensuring Compliance with requirements & standards.

2a) what is Legacy software? Explain briefly * its Impact in software engineering.

Ans: Legacy software:

The older programs which are developed decades ago that are still in use by performing modifications in order to meet the business requirements. the rapid increase of such systems may cause the risk to the larger organizations as they may require outdated hardware and operating system.

Many legacy systems remain supportive to core business functions and are important to business. Legacy systems sometimes have, inextensible designs, convoluted code, testcases and result that were never archived. a poorly maintained change history, lack of support and documentation make it difficult to troubleshoot the issues and leaving the system to security threats and the list can be quite long.

Types of Changes that a legacy system must take.

1) For meeting the need of new computing environment or technologies the existing legacy systems need to be changed.

2) There is a requirement to the use of modern systems, are database. This requirement the legacy system must be changed.

3) It needs to be enhanced for implementing new business requirements.

4) It needs to be re-designed in order to make its work within networking environment.

5) The legacy systems have to be reengineered.

2b) Explain the following:

i) waterfall model ii) Spiral model.

Ans: waterfall model: The waterfall model is also called as linear sequential model. or classic life cycle model. This model suggests a systematic, sequential approach. the software development starts with requirements gathering phase, then progress through analysis, design, Coding, testing and maintanance.

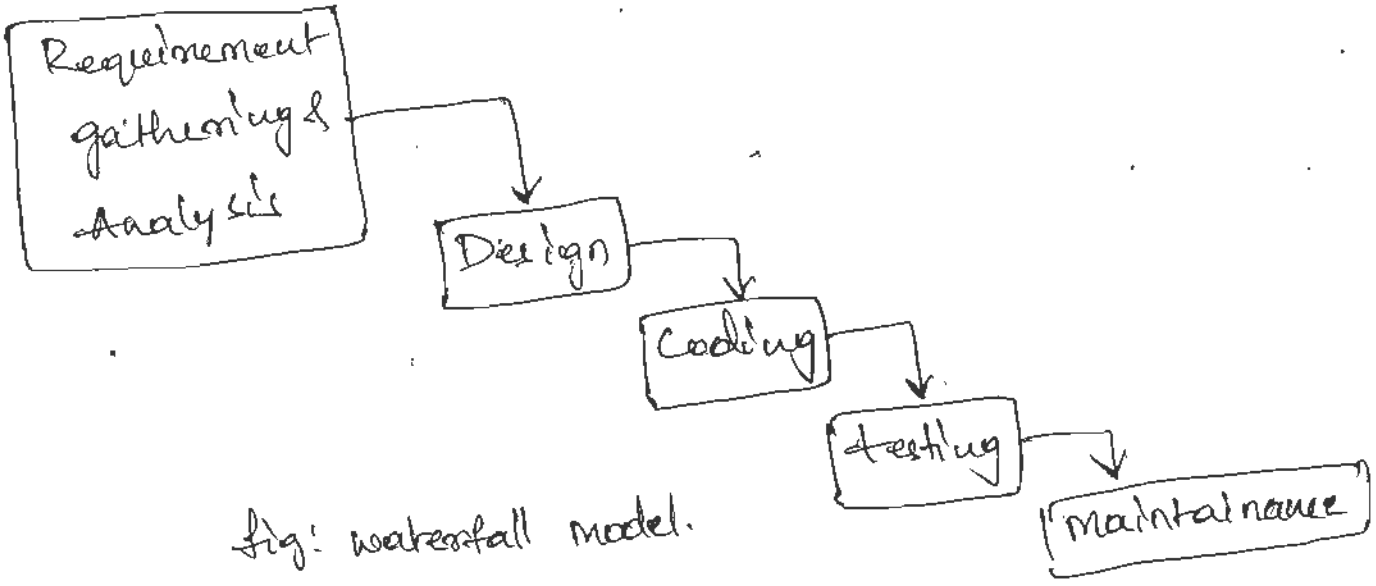


fig: waterfall model.

- In requirement gathering & Analysis phase the basic requirements of the system must be understood by software engineer, who is also called Analyst. All these requirement are then well documented & discussed further with the Customer.

The design is an intermediate step b/w requirement analysis & coding.

Coding is a step in which design is translated into machine-readable form If it is done in detail then coding can be done effectively.

Testing begins when coding is done. while performing testing the major focus is on logic internals of the software. the purpose of testing is to uncover errors fix the bugs and meet the customer requirement

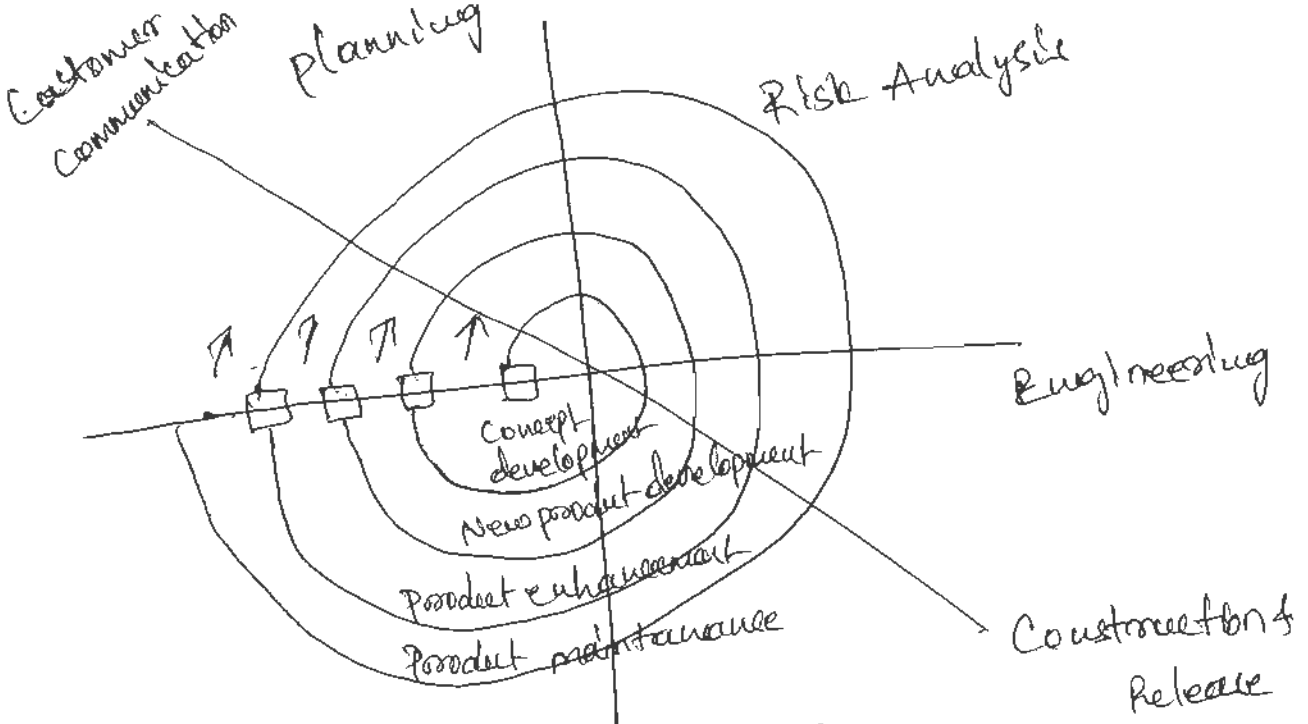
Maintainance is the longest life cycle phase the system is installed and put in practical use then errors may get introduced, correcting such errors putting it in use is the major purpose of maintainance activity.

Spiral model :- this model possess the iterative nature of prototyping model and controlled & systematic approaches of the linear sequential model.

This model gives efficient development of incremental versions of software in this model, the software is developed in series of increments.

The spiral model is divided into a no. of framework activities. usually there are six task regions.

the spiral model as initial pass, product specification is built and in subsequent passes around the spiral the prototype gets developed



Spiral model.

- During planning the cost and schedule of software can be planned and adjust based on feedback obtained. the task region can be described as,

- Customer communication
- planning
- Risk analysis
- construct and release
- customer evaluation

3. Explain in detail about Capability Maturity model Integration (CMMI) with process patterns, process assessment.

Ans: CMMI : The CMMI represents the process in two different ways.

1) Continuous model: It defines five Capability levels that are associated with specific goals & practises.

2) Staged model: It defines five maturity rather than capability levels, the same process areas, goals & practises are defined in Continuous model are also used by staged model.

Level 0 Incomplete:

The process area does not achieve the goals & objectives defined by CMMI level 1 capability.

Level 1: performed:

All the desired goals of the process are satisfied.

Level 2 Managed:

All the Criteria defined by the level 1 are satisfied
All the work associated with process area is as per the organisational policy.

All the developers have adequate resources to get the job done stakeholders.

All the work products are monitored & controlled, reviewed and evaluated as per the problem description.

Level 3 Defined:

All the level 2 criteria is achieved. the process are defined as per the organisational standards.

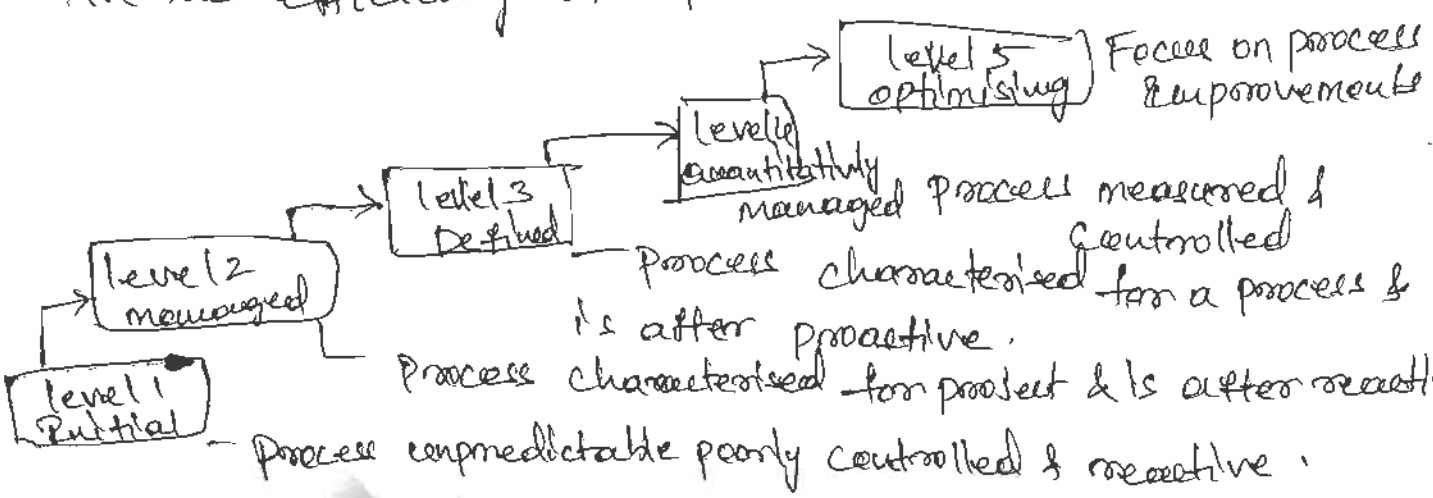
Level 4 Quantitatively managed:

All the level 3 criteria are satisfied.

process area is controlled and improved. the Quantitative objectives are used for managing the process.

Level 5 optimised:

All the level 4 criteria are satisfied. the process area are adapted. there is continuous improvement in the efficiency of process area.



Process patterns:

The process is defined as a series of activities in which one or more inputs are used to produced one or more outputs. the process pattern is a template. which appear as a general solution to a common problem.

Process pattern describes

→ Complete process

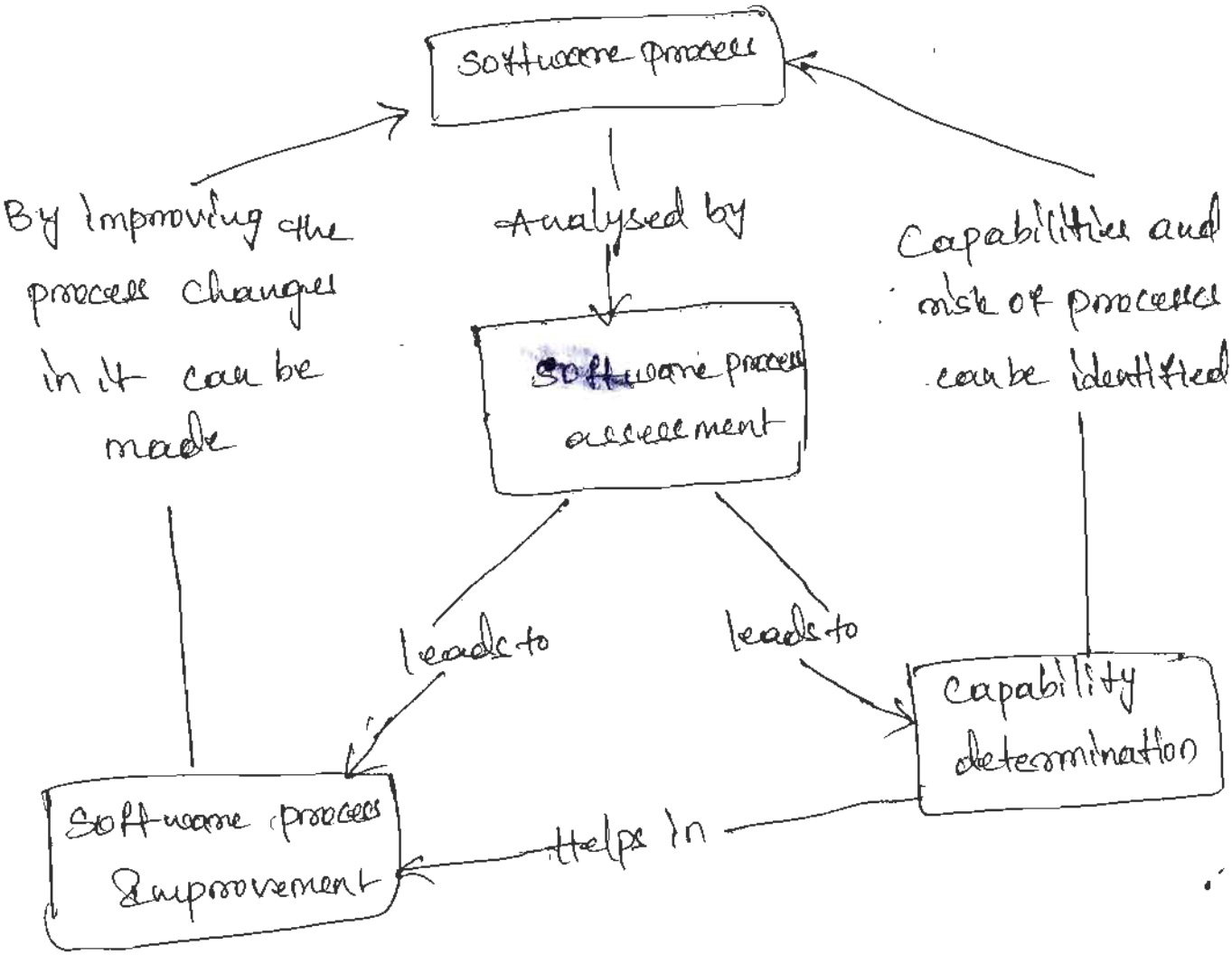
→ Important framework activity

→ Task within the framework.

- pattern name
- Intent
- type
- Initial content
- problem
- solution
- resulting content
- known uses

Process Assessment :-

- The software has to be delivered on time.
- The software should satisfy customer needs.
- The software should posses the long term quality characteristics.



- Standard CMMI assessment method for process Improvement
- CMMI based appraisal for internal process Improvement.

4 a) Compare function requirements with non-functional requirements.

Ans!

Functional Requirements	Non-functional requirements
1) Functional requirements define what the system should do i.e, specific functionality or tasks.	1) Non-functional requirements define how the system should perform i.e, system attributes or quality.
2) It is behaviour and functions of the system.	2) It is based on performance, usability and other quality Attributes
3) we define the action and operations of the system.	3) we define constraints and conditions under which the system must operate.
4) we can measure it easily	4) It is difficult to measure
5) It drives the core design and functionality of the system	5) It can affect the architecture and overall performance of the syst
6) It is directly related to user and business requirements	6) It is related to system performance and user experience.
7) we document its functional specifications, use cases etc.	7) we document technical specifications, performance etc.
8) It depends on what the system does.	8) It depends on how the system does.

ub) Discuss briefly how requirement validation is done?

Ans: Requirement validation:

It is the process of checking and confirming that the requirements defined for development accurately capture the needs and expectations of the customers.

> Gather Requirements:

- Identify all relevant stakeholders, including users, project managers and subject matter experts.
- utilize various techniques to gather requirements, such as interviews, workshops and surveys.
- Consolidate and document all gathered requirements ensuring clarity and organization.

> Analyze Requirements:

- Scrutinize each requirement for clarity, ensuring it is well defined and easy to understand.
- Assess the completeness of each requirements, ensuring it captures all necessary details and considerations.
- Check for consistency among requirements, identifying any conflicts or contradictions.

> Validate Requirements:

- Empty Inspections, where a team of reviewers examine the requirements for accuracy and consistency.
- Conduct reviews, where stakeholders provide feedback on the requirements, identifying potential issues
- Consider using automated tools to check for syntax errors, format inconsistencies and potential compliance issues.

> Document Findings:

- Document the findings from the validation process, including any identified issues, their severity and proposed solutions.
- Categorize the findings based on their type, such as ambiguity, incompleteness, inconsistency or mis-
  - alignment with objectives.
- Assign ownership and timelines for addressing the identified issues.

> Refine Requirements :

- Collaborate with stakeholders to refine the requirements based on the validation findings.
- Incorporate the proposed solutions into the requirement documentation
- Review the refined requirements to accurately reflect the customers needs and align with the object's objectives.

(10)

5a) Describe five desirable characteristics of a good software requirement specification document.

Ans: 1) Correctness: User review is used to provide the accuracy of requirements stated in the SRS. SRS is said to be perfect if it covers all the needs that are truly expected from the system.

2) Completeness: the SRS is complete if and only if, it includes the following elements.

- All essential requirements, whether relating to functionality, performance, design, constraints, attributes.
- Definition of their responses of the software to all realizable classes of input data in all available categories of situations.

3) Consistency: the SRS is consistent, if and only if no subset of individual requirements described in its conflict. there are three types of possible conflicts.

- the specified characteristics of real-world objects may conflicts.

- There may be a reasonable or temporal conflict between the two specified actions.
- two or more requirements may define the same real-world object but use different terms for that objects.

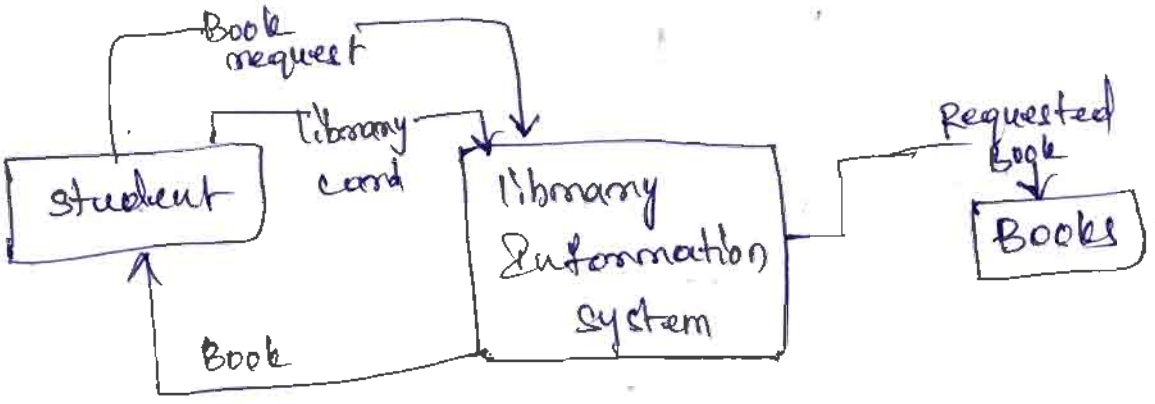
4) Unambiguousness : SRS is unambiguous when every stated requirement has only one interpretation. This suggests that each element is uniquely interpreted. In case there is a method used with multiple definitions, the requirements report should determine the implications in the SRS so that it is clear and simple to understand.

5) Modifiability : SRS should be made as modifiable as likely and should be capable of quickly obtain changes to the system to some extent. modifications should be perfectly indexed and cross-referenced.

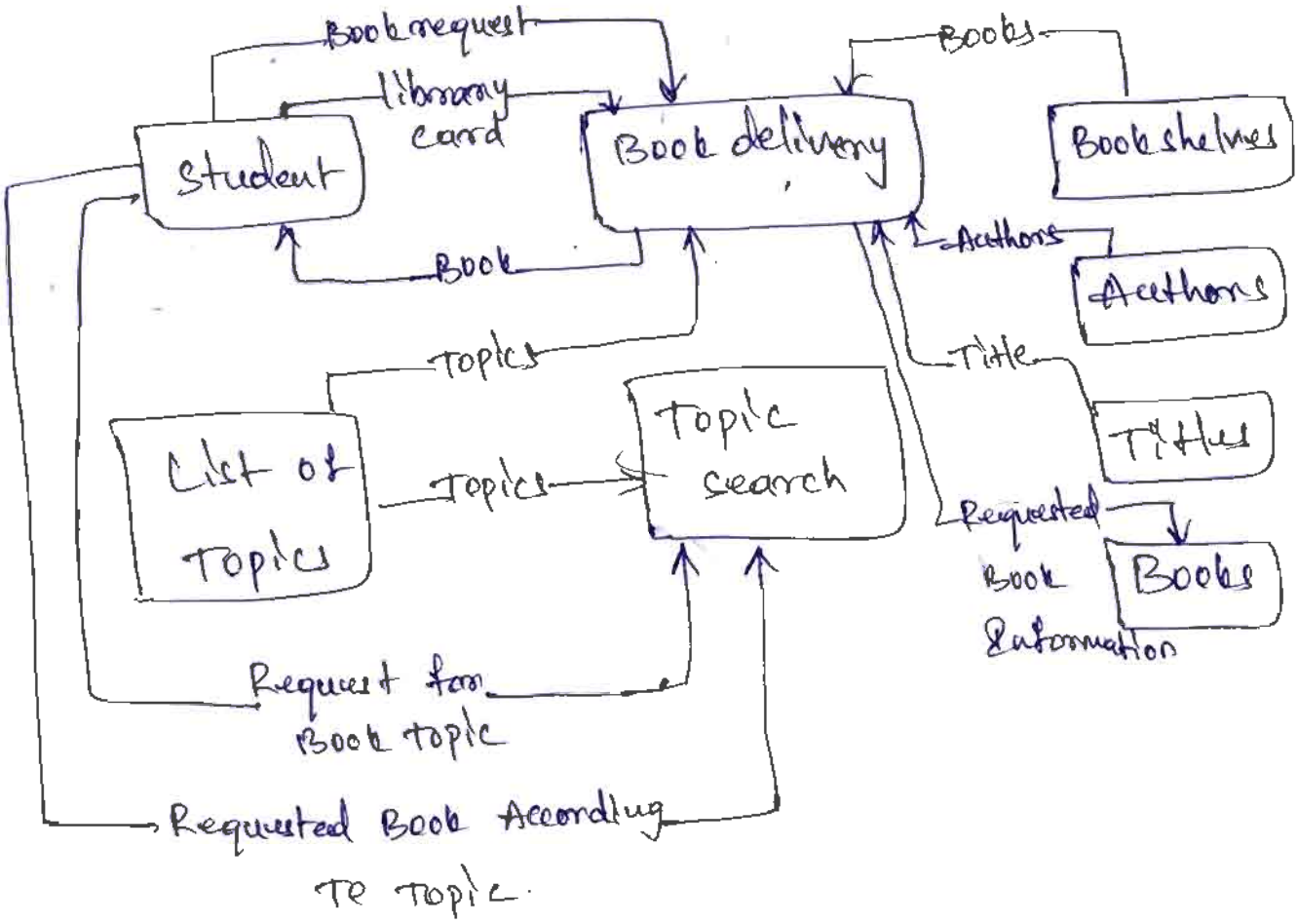
6) Testability : An SRS should be written in such a method that it is simple to generate test cases and test plans from the report.

5b) Draw the Complete DFD at least upto 2-levels for a library management system.

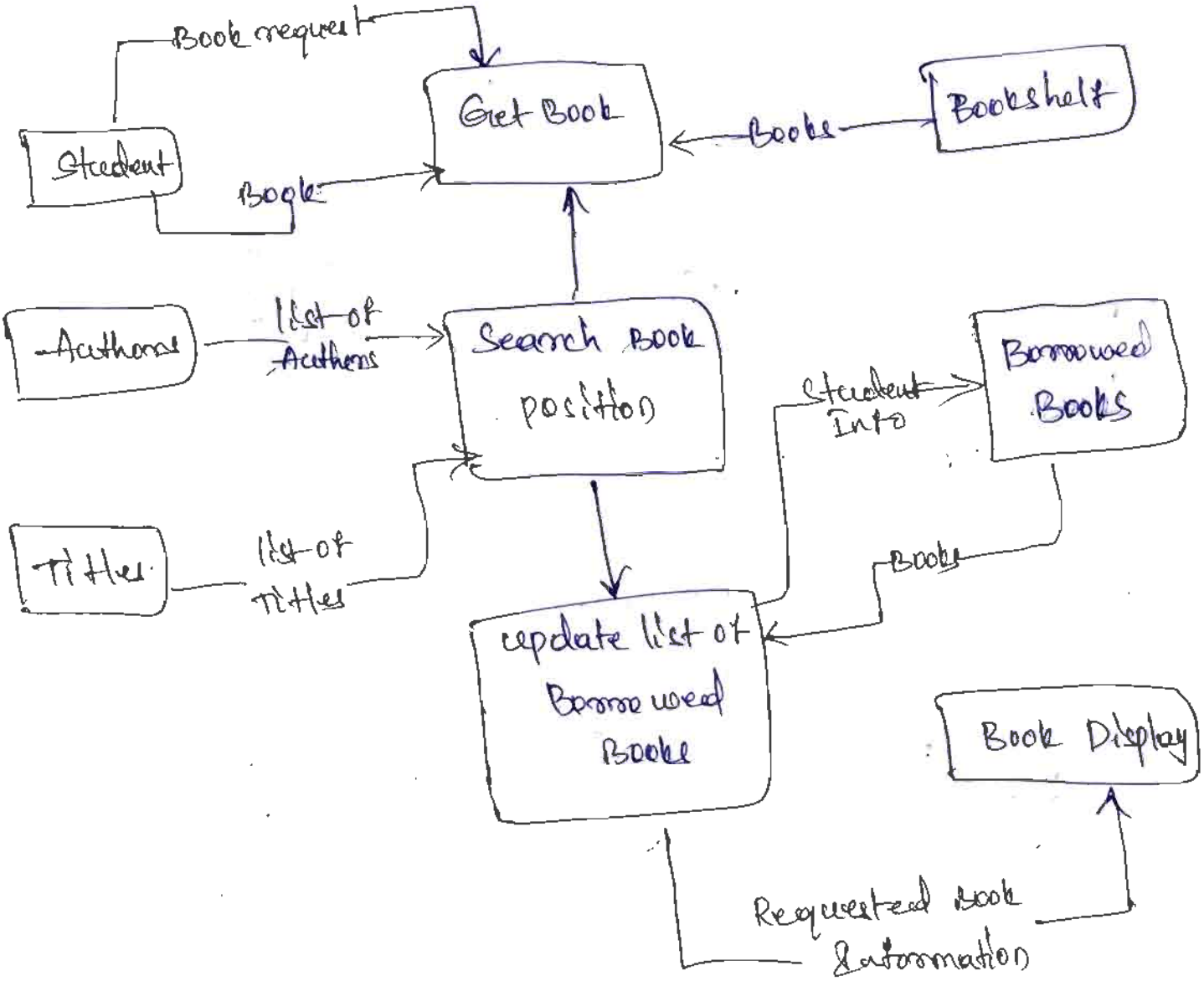
DFD for library management system Level 0:



DFD for library management system Level 1:



DFD for library management system level 2:



6) what is design model? Explain the process of design Concepts that should be considered when building models.

Ans: Design modeling in software Engineering represents the features of the software that helps engineer to develop it. the architecture, the user interface, and the component level detail. Design modeling provides Different methods like data-driven, pattern driven or object oriented methods are used for constructing the design model. It is mainly classified into four types.

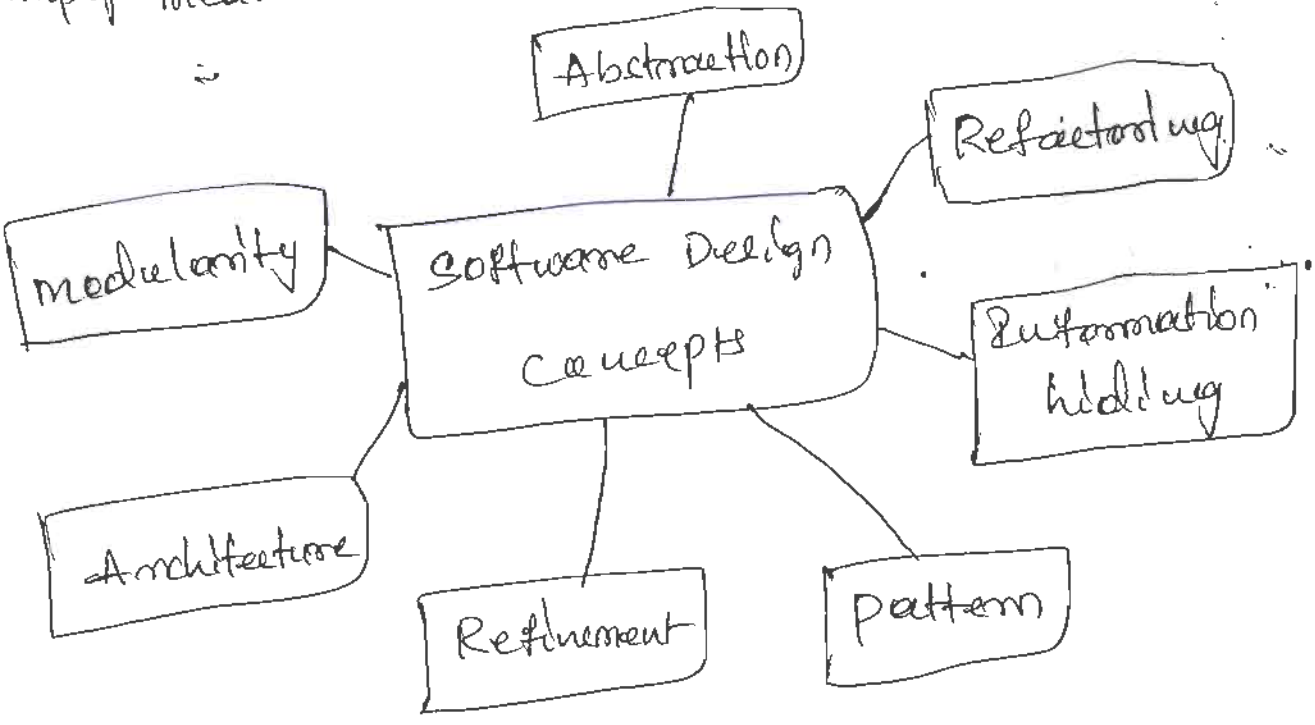
→ Data Design: It represents the data objects and their interrelationship in an entity-relationship diagram. Entity relationship consists of information required for each entity or data objects as well as it shows the relationship between these objects. It shows the structure of the data in terms of the tables. It shows three types of relationship - one to one, one to many and many to many.

Architectural design: It defines the relationship between major structural elements of the software. It is about decomposing the system into interacting components. It is expressed as a block diagram defining an overview of the system structure.

User Interfaces design: It represents how the software communicates with the user i.e. the behavior of the system. It refers to the product where user interact with controls or displays of the product.

Component level design: It transforms the structural elements of the software architecture into a procedural description of software components. It is a perfect way to share a large amount of data. Components need not be concerned with how data is managed at a centralized level i.e. components need not worry about issues like backup and security of the data.

Software Design Concepts: Concepts are defined as a principal idea or invention that comes into our mind or in thought to understand. The software design concept simply means the idea or principle behind the design.



Abstraction: hide irrelevant data.

Abstraction simple means to hide the details to reduce complexity and increases efficiency or quality.

Modularity:- subdivide the system.

Modularity simply means dividing the system or project into smaller parts to reduce the complexity of the system or project. Modularity in design means sub dividing a system into smaller parts so that these parts can be created independly in different systems to perform different functions.

Architecture:- design a structure of something.

Architecture simply means a technique to design a structure of something. Architecture in designing software is a concept that focuses on various elements and the data of the structure. these components interact with each other and use the data of the structure in architecture.

Refinement: removes impurities.

Refinement simply means to refine something to remove any impurities if present and increase the quality.

pattern: a repeated form

The pattern simply means a repeated form or design in which the same shape is repeated several times to form a pattern.

Information Hiding: hide the information

Information hiding simply means to hide the information so that it cannot be accessed by an unaccounted party.

Refactoring: reconstruct something.

Refactoring simply means reconstructing something in such a way that it does not affect the behavior of any other features.

(a)

7a) Explain the process of mapping dataflow into software architecture?

Ans: Mapping data flow into software architecture is the process of transforming data flow models into a structured software architecture that defines components, their responsibilities, and interaction. This is a key step in software design, bridging analysis and implementation.

Data flow diagram consists of processes, Datastores, Data flows, External entities. Each process in the DFD is mapped to software module, component, or a service.

Two common strategies are used in software architecture

a) transform mapping: It is used when the system has clear input, central processing and clear output.

- Identify input flow
- Identify transform center
- Identify output flow
- Map them into a layered architecture.

b) Transaction mapping :- It is used when a single input triggers multiple possible actions.

- Identify the transaction center
- Identify action paths
- Map each action path to a separate module

7 b) List the golden rules of architectural design ?

Ans: The golden rules of Architectural Design :

> Abstraction: Focus on what the system does rather than how it is implemented.

> Modularity : Divide the system into independent, manageable modules.

> Separation of Concerns: Separate different responsibilities (UI, business logic, data handling).

> Low Coupling : Minimize dependencies between modules to reduce impact of changes.

> High Cohesion : Ensure each module performs a single, well-defined task.

> Information Hiding : Hide internal details and expose only necessary interfaces.

> Simplicity : Keep the architecture simple and avoid unnecessary complexity.

> Reusability : Design components so they can be reused in other systems.

> Scalability : Architecture should support future growth in users and data.

> Maintainability and Evolvability : The system should be easy to modify and extend over time.

8a) Describe the framework for software product metrics?

Ans: A software product metrics provides a structured way to measure the quality and characteristics of a software product. these metrics help assess whether the software meets user requirements, quality standards and business goals.

Quality Attribute Based framework :

> size-oriented metrics : measure the physical size of the software.

Ex: Lines of code, number of modules, classes and functions.

> Function-oriented metrics: Measure functionality delivered to the user

Ex: Function points and Feature points.

> Quality Metrics: Derived from quality model.

- Functionality – Requirement coverage.
- Reliability – Failure rate
- Usability – User error rate
- Efficiency – Response time
- Maintainability – Mean time to change
- Portability – platform independence.

8b) Differentiate between Black box and white box testing. (6)

Ans:

White - Box Testing	Black - box Testing.
> white - Box testing known as glass box testing	> Black box Testing also called as behavioral testing.
> It starts early in the testing process.	> It is applied in the final stages of testing
> In this testing knowledge of Implementation is needed	> In this testing knowledge of Implementation is not needed
> white box testing is mainly done by the developer.	> This testing is done by the testers.
> In this testing, the tester must be technically sound	> In black box testing, testers may or may not be technically sound.

9) Discuss about metrics for design model and source code.

Ans Metrics for Design :-

Architectural design metrics focus on characteristics of the program architectural structure and the effectiveness of modules. These metrics are black box in the sense that they do not require any knowledge of the inner workings of a particular software component.

three software design complexity measures:

- structural complexity
- Data complexity
- system complexity.

> structural complexity of a module i is defined in the following manner:

$$S(i) = f_{2\,out}(i)$$

where $f_{out}(i)$ is the fan-out of module i.

> Data Complexity provides an indication of the complexity in the internal interface for a module i and is defined as

$$D(i) = v(i) / [fout(i) + 1].$$

where $v(i)$ is the number of input and output variables that are passed to and from module i.

> System Complexity is defined as the sum of structural and data complexity. specified as

$$C(i) = s(i) + D(i).$$

Metrics for source code:

Halstead proposed the first analytic laws for Computer science by using a set of primitive measures, which can be derived once the design phase is complete and code is generated. these measures are listed below

n_1 = number of distinct operators in a program.

n_2 = number of distinct operands in a program.

N_1 = total number of operators

N_2 = total number of operands.

By using these measures, Halstead developed an expression for overall program length, program volume, program difficulty, development effort. The program length (N) can be calculated by using the following equation.

$$N = n_1 \log_2 n_1 + n_2 \log_2 n_2$$

The program volume (V) can be calculated by using the following equation.

$$V = N \log_2 (n_1 + n_2)$$

Program ~~language~~ volume depends on the programming language used and represents the volume of information required to specify a program. Volume ratio (L) can be calculated by using the equation.

$$L = \frac{\text{volume of the most compact form of a program}}{\text{volume of the actual program}}$$

Value of L must be less than 1. Volume ratio can be calculated by the equation $L = (2/n_1) * (n_2/N_2)$.

10 a) Explain about Software Reviews and Formal technical reviews.

Ans: Software Review :- It is a systematic inspection of a software by one or more individuals who work together to find and resolve errors and defects in the software during the early stages of SDLC. Review helps software engineers in validating the quality, functionality and other vital features and components of the software.

Software review is used, to verify various documents like requirements, system designs, codes, test plans and test cases.

process : Entry Evaluation

↓

Management preparation

↓

Review planning

↓

preparation

↓

Examination and Exit Evaluation

There are mainly 3 types of software reviews.

1) Software peer Review : Peer review is the process of assessing the technical content and quality of the product and it is usually conducted by the author of the work product along with some other developers. Peer review is performed in order to examine or resolve the defects in the software, whose quality is also checked by other members of the team.

2) Software Management Review : It evaluates the work status. In this section decisions regarding downstream activities are taken.

3) Software Audit Review : It is a type of external review in which one or more critics, who are not a part the development team, organize an independent inspection of the software product and its processes to assess their compliance with stated specifications and standards. This is done by managerial level people.

Formal Technical Reviews: A formal technical review is a software quality assurance activity performed by software engineers. the objectives of the FTR are.

1) To uncover errors in function, logic, or implementation for any representation of the software.
2) To verify that the software under review meets its requirements.
3) To ensure that the software has been represented according to predefined standards.
4) To achieve software that is developed in a uniform
5) To make projects more manageable.

The FTR is actually a class of reviews that includes walkthroughs, inspections, round-robin reviews and other small group technical assessments of software. Each FTR is conducted as a meeting and will be successful only if it is

properly planned, controlled and attended. In the sections that follow, guidelines similar to those for a walk through are presented as a representative formal technical reviews.

> The Review Meeting

> Review Reporting and Record Keeping

> Review Guidelines

- Review the product, not the producer
- Set an agenda and maintain it
- Limit debate and rebuttal
- Enunciate problem areas, but don't attempt to solve every problem noted.
- Take written notes
- Limit the no. of participants and insist upon advance preparation
- Develop a checklist for each product that is likely to be reviewed.
- Allocate resources and schedule time for FTRs.
- Conduct meaningful training for all reviewers.

10b) Discuss software risk?

b) software risk:

(20)

Ans:

Risk is uncertain events associated with future events which have a probability of occurrence but it may or maynot occur and it occurs it brings loss to the product - Risk identification and management are very important task during software project development because success and failure of any software project depends on it.

Various kinds of Risks in software Development:

> Schedule Risk: Refers to time related risks or project delivery related planning risks. the wrong schedule effects the project development and delivery. these risks are mainly indicates to running behind time as a result project development doesn't progress timely and it directly impacts to delivery of project.

Reasons for schedule risks -

- time is not estimated perfectly
- Improper resource allocation

- Tracking of resources like system, skill, staff etc.
- Frequent project scope expansion
- Failure in function identification.

> Budget Risk :- Refers to the monetary risks mainly it occurs due to budget overruns. Always the Financial aspect for the project should be managed as per decided but if financial aspect of project mismanaged then there budget concerns will arise by giving rise to budget risk.

Reasons for Budget risks -
- wrong & improper budget estimation
- unexpected project scope expansion
- mismanagement in budget handling
- cost overrun

> Operational Risk : Refers to the procedural risks means there are the risks which happen in day-to-day operational activities during project development due to improper process implementation or some external operational risks.

Reasons for operational risks -

- Insufficient resources.
- Conflict between tasks and employees
- Improper management of tasks
- No proper planning about project
- Less no. of skilled people

> Technical Risks: Refers to the functional risk or performance risk which means this technical risk mainly associated with functionality of product or performance part of the software product.

Reasons for Technical Risks —

- Frequent changes in requirement
- Less use of feature technologies
- Less number of skilled employee
- High complexity in implementation
- Improper integration of modules.

> Programmatic Risks: Refers to the external risk or other unavoidable risks. These are the external risks which are unavoidable in nature. These risks come from outside and it is out of control of programs.

Reasons for programmatic risks -

- Rapid development of market
- Running out of fund / Limited fund for project development
- changes in Government rules / policy
- Loss of Contracts due to any reason.

11. Explain the concept of Software Quality Assurance.

Ans: SQA is simply a way to assure quality in the software. It is the set of activities which ensure processes, procedures as well as standards are suitable for the project and implemented correctly. Software Quality Assurance is a process which works parallel to development of software. It focuses on improving the process of development of software so that problems can be prevented before they become a major issue. Software Quality Assurance is a kind of umbrella activity that is applied throughout the software process.

Generally the quality of software is verified by the third party organization.

Software quality assurance focuses on:

- Software's portability
- Software's reliability
- Software's reusability
- Software's correctness
- Software's maintainability
- Software's error control.

Software Quality Assurance has:

1) A quality management approach
2) Formal technical reviews
3) Multi testing strategy
4) Effective software engineering technology
5) Measurement and reporting mechanism

Major Software Quality Assurance Activities:

1) SQA Management plan
2) Set the Check points
3) Multi testing Strategy

4) Measure change Impact

5) Manage Good Relations

Benefits of Software Quality Assurance :-

1) SQA produces High quality software.

2) High quality application saves time and cost.

3) SQA is beneficial for better reliability.

4) SQA is beneficial in the condition of no maintenance for a long time.

5) High quality commercial software increase market share of Company.

6) Improving the process of creating software.

7) Improves the quality of the software.

8) It cuts maintenance costs. Get the release right the first time, and your company can forget about it and move on to the next big thing. Release a product with chronic issues, and your business bogs down in a costly, time-consuming, never-ending cycle of repairs.