Total No. of Pages 02

Roll No.....

5TH SEMESTER

B.Tech (COE)

MID SEMESTER EXAMINATION (September – 2018)

CO 301 Software Engineering

Time: 1:30 Hours Max. Marks: 30

Note: Answer all questions. Assume suitable missing data, if any.

Total No. of Pages 2

Roll No. .....

FIFTH SEMESTER

B.Tech. (CSE)

MID SEMESTER EXAMINATION *(September-2019)*

CO301 SOFTWARE ENGINEERING

Time: 1 Hour 30 Minutes *Max. Marks: 30*

Note: Answer all questions.
Assume suitable missing data, if any.

Total No. of Pages: 01

Roll No. 438

B. Tech. (CO)

Mid-Semester Examination Fifth Semester

(September 2022)

CO301 SOFTWARE ENGINEERING

Time: 1.5hrs *Max. Marks: 20*

Note: *Answer ALL questions. Assume suitable missing data if any.*

Total No. of Pages 3

Roll No. .....

FIFTH SEMESTER

B.Tech. (CO)

SUPPLEMENTARY EXAMINATION *(Feb-2018)*

CO-301 SOFTWARE ENGINEERING

Time: 3 Hours *Max. Marks: 40*

Note: Question 1 is compulsory. Answer any FOUR questions from remaining. Assume suitable missing data, if any.

Total Number of Pages 3

Roll No. .....

EIGHTH SEMESTER

B.E. (COE)

END SEMESTER EXAMINATION MAY-JUNE 2010

COE- 412 SOFTWARE ENGINEERING

Time: 3 Hour *Max. Marks : 70*

Note : Answer any *FIVE* questions .
Assume suitable missing data, if any.

Total Number of Pages 1

Roll No. .....

EIGHTH SEMESTER

B.E. (COE)

MID SEMESTER EXAMINATION *MARCH* 2010

COE-412 SOFTWARE ENGINEERING

Time: 1 Hour 30 Minutes *Max. Marks : 20*

Note : Answer *ALL* questions.
Assume suitable missing data, if any.

Total No. of Pages 2

Roll No. .....

EIGHTH SEMESTER

B.E. (COE)

MID SEMESTER EXAMINATION *MARCH* 2005

COE-412 SOFTWARE ENGINEERING

Time: 1 Hour 30 Minutes *Max. Marks : 20*

Note : Attempt *ALL* questions.
Assume suitable missing data, if any.

Roll No. .....

B.Tech. (CO)

(Nov/Dec-2018)

Total No. of Pages 2

FIFTH SEMESTER

END SEMESTER EXAMINATION

CO301 SOFTWARE ENGINEERING

Time: 3 Hours *Max. Marks: 40*

Note: Answer any FIVE questions.
Assume suitable missing data, if any.

Roll No. .....

B.Tech. (CO)

(February -2020)

Total No. of Pages 2

FIFTH SEMESTER

END SEMESTER EXAMINATION

CO-301 SOFTWARE ENGINEERING

Time: 3 Hours *Max. Marks: 40*

Note: Answer any FIVE questions. Question No. 1 is compulsory.
Assume suitable missing data, if any.

Total no. of Pages: 02

Roll no. 138

5th SEMESTER

B.Tech.

END TERM EXAMINATION *Nov/Dec-2022*

CO 301 SOFTWARE ENGINEERING

Time: 03:00 Hours *Max. Marks: 40/50*

Note: All questions carry equal marks.
Attempt all five questions.
Assume suitable missing data, if any.

Total No. of Pages=1

Roll No. .....

IVthSEMESTER OLD

B.Tech.(CO/SE)

END SEMESTER EXAMINATION *May-2018*

CO/SE-215 SOFTWARE ENGINEERING

Time:3:00 Hours *Max. Marks:70*

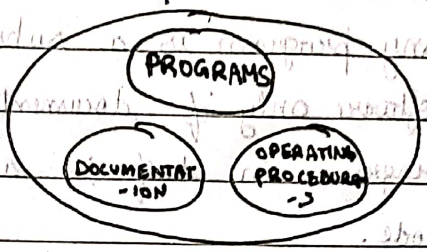
Note:Answer any 5 questions. Assume suitable missing data, if any.

Date : / /
Page No.

UNIT-1 : Introduction

1.a) What is a Software? Explain its components? 2020 E

Software is instructions (computer programs). that when executed provide desired function and performance, data structures that enable the programs to adequately manipulate information and documents that describe the operation and use of the programs.



COMPONENTS:

→ PROGRAM: It is a collection of source code and object code. A program is a subset of software and it becomes software only if documentation and operating manuals are prepared.

→ SOFTWARE DOCUMENTS: It consists of all the descriptions, programs, graphics and instructions pertaining to design, coding, testing and preparation of software.
Good software contains the following type of documentation:
→ Analysis/specification | → Design | → Coding | → Testing.

→ OPERATING PROCEDURE: It provides help to operating staff for producing desired output with that s/w including how to work with it, how do install it etc.
It include user manual and operational manual.

Q.1 Differentiate between:
a) Process and product 2018 S

	PRODUCT	PROCESS
BASIC	End product of the project	Use to create the project
FOCUS IS ON	End result	Evolution Steps
TEND TO BE	Short term	Long term
FOLLOWS	Firm Deadlines	Consistent routines
GOALS	successful completion of the job	Successful growth in the particular skill.

Date : / /
Page No.

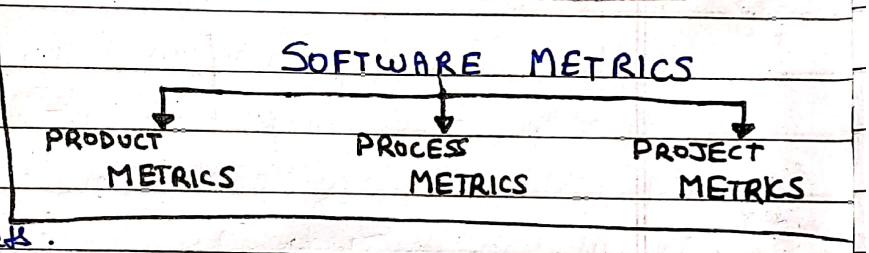
b) Program and software 2018E

Software is more than programs which consist of programs, documentation of any facet of the program and the procedures used to set up and operate the software system.

Any program is a subset of software and it becomes a software only if documentation and procedure manuals are prepared. Program in itself is a combination of: source code and object code.

(c) Define software metrics and classify them. 2018M [c] Product metrics and quality 2010E

A metric is a quantitative measure, so Software metrics is a standard to quantitatively characterize different aspects of software process or products.



Product metrics: They quantify the software product characteristics such as size, reliability, complexity, functionality etc.

Process metrics: They are measure for the attributes of SD process and environment including productivity, quality, failure rate, efficiency etc.

Project metrics: They describe the project characteristics and execution including # SDEs, staffing pattern over life cycle of software, cost etc.

Date : / /
Page No.

1 [a] What are the major advantages of first constructing a working prototype before starting to develop actual software? What are the disadvantages of this approach? 2018 E (7)

PROTOTYPING MODEL

By constructing a prototype and submitting it for evaluation, many customer requirements get properly defined and technical issues get resolved with the prototype. This minimizes later changes request from the customer and associated redesign costs.

This model is the most appropriate for projects that suffer from risks arising from technical uncertainties and unclear requirements. A constructed prototype helps overcome these risks.

DISADVANTAGES

1. Increased Cost for Routine Projects : It can lead to high costs for projects that do not face significant risks. This is because it may include added unnecessary complexity for straightforward tasks.

2. It is mostly effective for projects where risks are identified upfront before development.

3. It is ineffective for addressing risks that emerge later in the dev. cycle and were not anticipated at beginning. E.g. someone left the project.

1[a] What do you mean by process models. Explain prototype and Incremental process model. Discuss their strengths and weaknesses. 2010 E

A process model / software life cycle model is a particular abstraction that represents a software life cycle often called software development life cycle (SDLC). Its defined as:

"The period of time that starts when a software product is conceived and ends when the product is no longer available for use. The SDLC typically includes a requirement phase, design phase, implementation phase, test phase, installation and check out phase, operation and maintenance phase and sometimes retirement phase".

Q2. What are the characteristics to be considered for the selection of life cycle model? Discuss in detail. 2022M [5][CO1,CO4]

3. Why do we feel that characteristics of requirements play a very significant role in the selection of a life cycle model? Also, discuss the selection process parameters for a life cycle model. 2018M [2.5+2.5=5]

Date : / /
Page No.

They play a significant role in selecting a life cycle model because they determine which model aligns best with the project's specific needs and constraints. Diff life cycle methods are suited to handle various types of requirements.

The selection of a suitable model is based on the following characteristics

1.) CHARACTERISTICS OF REQUIREMENT

Requirements are very important for the selection of an appropriate model. There are number of situations and problems during requirements capturing and analysis.

→ Are requirements easily understandable and defined?
→ Do we change requirements quite often?
→ Can we define requirements early in the cycle?
→ Requirements are indicating a complex system to be built?

2.) STATUS OF DEV. TEAM

It in terms of availability, effectiveness, knowledge, intelligence, team work etc. is very important for the success of the project. If we know above mentioned parameters and characteristics, then we can choose appropriate model by answering the following Qs.

→ Less exp. on similar projects → less domain knowledge (new to team)
→ Less exp. on tools to use → Availability of training (if reqd.)

3.) INVOLVEMENT OF USERS

It increases the understandability of the project. Hence, user participation, if available plays a very significant role, some issues are:

→ User invol. in all phases? → Limited user participation
→ User has no. exp of participation in similar projects → users are expert of prob. domain

4.) TYPE OF PROJECT and ASSOCIATED RISK

→ Project is enhancement of existing system
→ Funding is stable for the project
→ High reliability req.
→ tight project schedule
→ use of reusable components
→ are resources scarce?

Date : / /
Page No.

Q3. Explain the following:
a) Spiral life cycle model and its limitations 2022M

Q3. Explain Spiral model in software development life cycle model. Discuss the conditions based on all the parameters where spiral model is the one best suited. 2018M [5]

Spiral model is a SDLC model which provides support for risk handling. The radial direction of the model represents cumulative costs. Each path around the spiral is indicative of increased costs. The angular dimension represents the progress made in completing one cycle. Each phase is split into four sections of major activities:

→ PLANNING: determination of objectives, alternatives and constraints.
→ RISK ANALYSIS: analyze alts and attempts to identify and resolve risks involved
→ DEVELOPMENT: product dev. and testing product.
→ Assessment: customer evaluation.

It is a flexible model. In the first phase, planning and risk analysis occurs, along with prototype creation and customer evaluation. In the second phase, a refined prototype is built, requirements are documented and validated and customers assessed further. In the third phase, a more traditional development approach begins, focusing on identifying and classifying problems by the level of risk.
Each phase ends with a review by stakeholders and if plan fails, spiral terminates otherwise it moves to next cycle.

This model offers flexibility, can adapt to other models and emphasizes risk analysis and early error elimination in development.

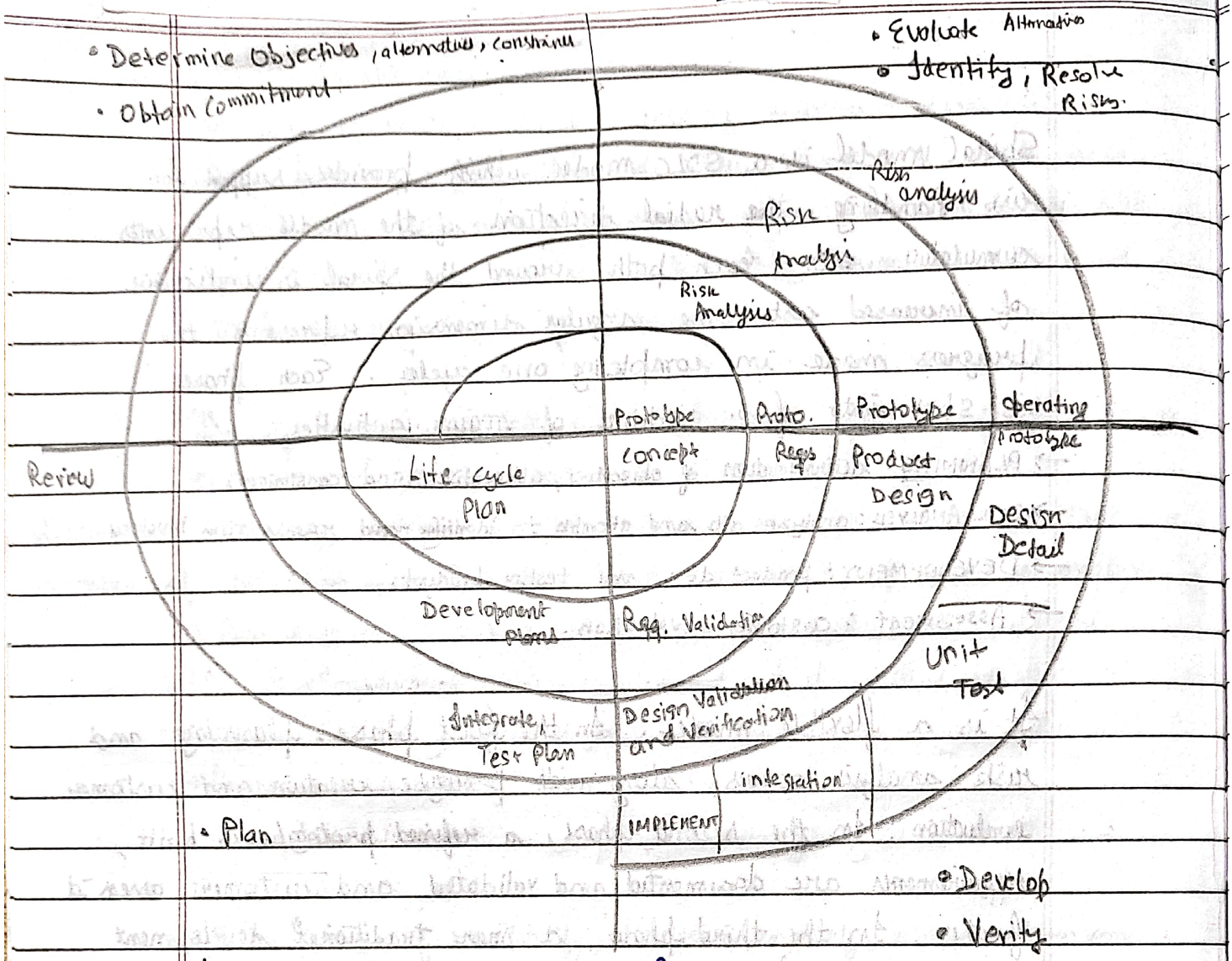
LIMITATIONS
It has some limitations that have to resolved to make it universal:
→ lack of explicit process guidance in determining objectives, constraints, alternatives.
→ relying on risk assessment expertise
→ providing more flexibility than req. for many applications.

2. What are the framework of activities associated with Spiral Model and V Model? Identify their strength and weakness. 5 2005M

OUT OF SYLLABUS

Date : / /

Page No.



It is used for complex and large SD projects, as it allows for a more flexible and adaptable approach to SD.

Its also well suited to projects with significant uncertainty or high lvl of risk.

c) As you move outward along the process flow path of the spiral model, what can you say about the software that is being developed or maintained? 2019M

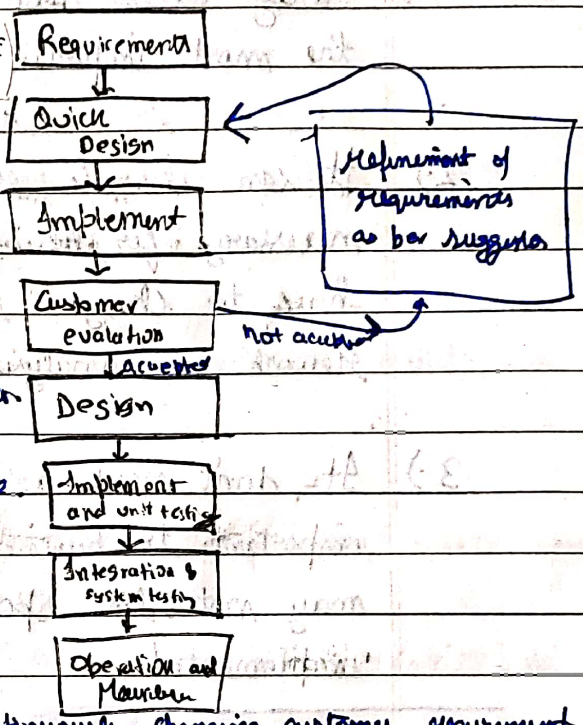
As product/work moves outwards on the spiral, the product moves towards a more complete state and the level of abstraction at which work is performed is reduced. (i.e. implementation specific work accelerates as we move further from the origin).

Date : / /
Page No.

5. Discuss the advantage and disadvantage of the prototyping model over the evolutionary model for software development. 2019M [5]

Prototyping model involves creating a working prototype of the S/w before the system which is refined using customes feedback. The prototype code is discarded but exp. gained from it aids in creating actual system.

Q2.a) Explain prototyping model. What are the advantages and disadvantages of this model? 2018S [4]



ADVANTAGES OVER Evolutionary

→ flexible in design and easy to detect errors
→ there is scope of refinement : new req, missing functionality can be added
→ It can be reused in future.
→ It helps developers and user both to understand system better.
→ Integration req are very well understood and deployment channels are decided at an early stage.

DISADVANTAGES

→ It is costly.
→ It has poor documentation because of continuously changing customer requirement.
→ There may be too much variation in requirement
→ There may be suboptimal soln coz of developes in a hurry to build prototypes.
→ There may be incomplete or inadequate problem analysis.

[b] Explain why the spiral model is considered to be a meta model? 2018E (7)

Spiral model is called a meta-model it subsumes all the other SDLC models.

→ It incorporates stepwise approach of classical waterfall model.

→ It uses the approach of prototyping model by building a proto. before at each phase as a risk handling technique.

→ It can be considered as supporting the evolutionary model as iterations along the spiral can be considered as evolutionary lvls through which the complete system is built.

d) Explain why incremental development is the most effective approach to developing business software systems. Why is this model less appropriate for real-time systems engineering? 2019M

[2+2+3+3=10]

Date : / /
Page No.

It is most effective for developing business soft. systems coz:

1.) It allows early access to the most valuable functionality. Early access not only gains value early, but also gives the most important component of the system most testing.

2.) It can handle requirement changes as well, which is necessary for business software systems whose requirements have to change when business changes. It also reduces rework on analysis and documentation when change happens.

3.) It don't need a complete specification in advance. This is important to business software because the detailed requirements may not be spotted until part of the functionalities are implemented.

It is less appropriate for real-time systems engineering because the computing time may suffer ~~when~~ without a complete plan for whole system.
Real time systems are time critical, therefor can not start w/o a complete plan.

b) How does "project risk" factor affect the spiral model of software of software development? 2020E [4+4]

Barry Boehm incorporated "project risk" and gave results to life cycle model. Each phase involves risk analysis which include analyzing alternatives, and attempts to identify and resolve the risks involved.
So, during initial & first phase risks are analyzed and by the third phase risks are known and a traditional approach is followed.

The focus is on identification of problems and the classification of these into different levels of risk, the aim being to eliminate high-risk problems before they threaten the software operation and cost.

Date : / /
Page No.

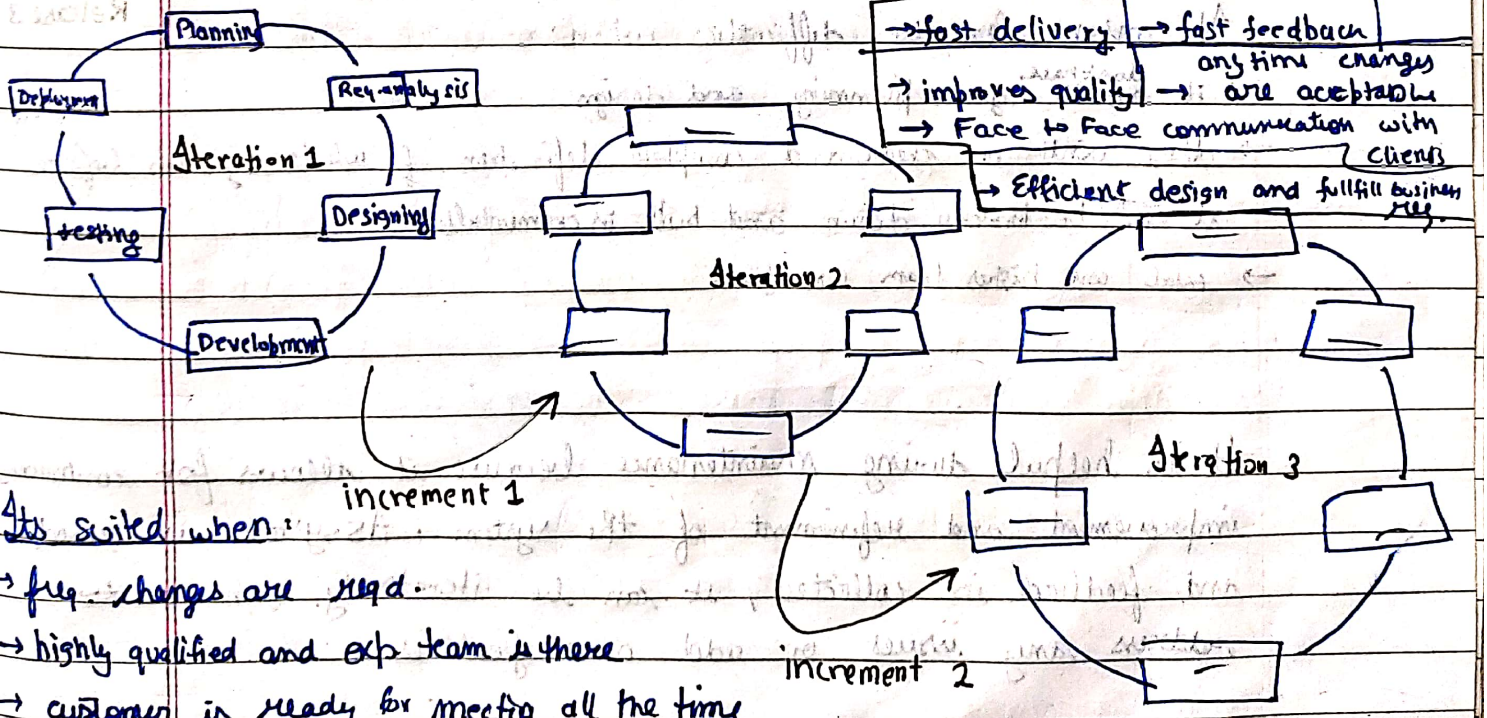
(iii) Agile life cycle model 2022M [5][CO1,CO4]

Agile ⇒ "ability to respond to change", change refers to change in terms of requirement, technology and people.

Agile model is a combination of iterative and incremental models. It is a methodology which promotes continuous development and testing throughout the SDLC of project.

It focusses on process flexibility and customer satisfaction by fast delivery of working software product.

→ Here, the product is broken into small builds and these builds are provided in iterations.
→ Iteration typically lasts from about 2-3 weeks.
→ Each iteration comprises of cross-functional teams working simultaneously on various areas.
→ Each build is incremental in terms of features and the final build holds all the features required by the customer.
→ A working product is provided to the customer and imp. Stakeholders at the end of each iteration.



ADVANTAGES
→ fast delivery
→ fast feedback
→ improves quality
anytime changes are acceptable
→ Face to Face communication with clients
→ Efficient design and fulfill business req.

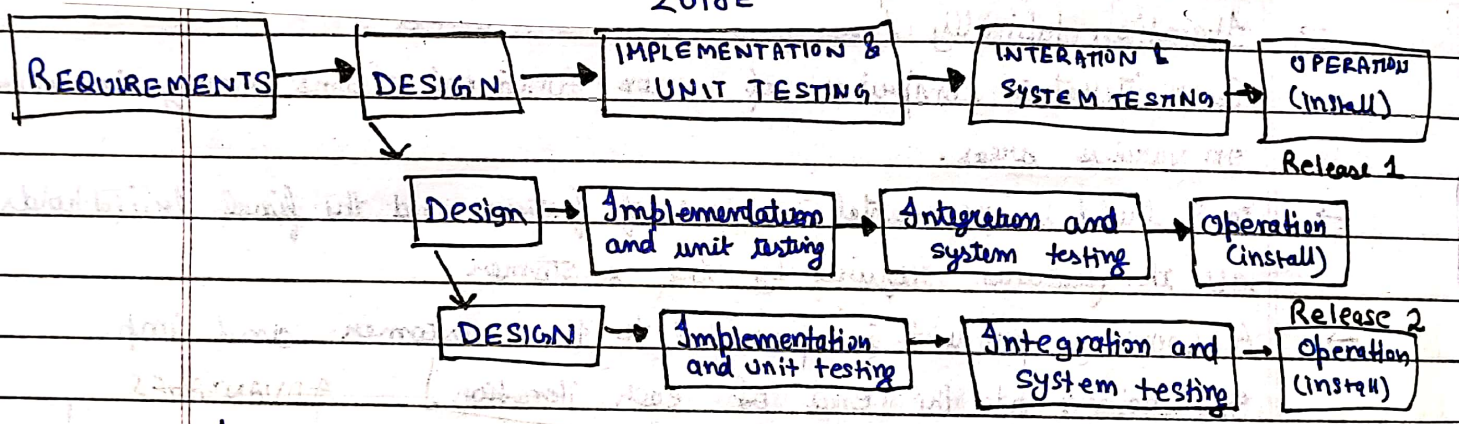
Its suited when:
→ freq. changes are reqd.
→ highly qualified and exp team is there
→ customer is ready for meeting all the time
→ Project size is small / large.

Date : / /
Page No.

(c) Differentiate between increment and evolutionary process models. 2018M [5*2=10]

In evolutionary model, the complete cycle of activities is repeated (i.e it undergoes each phase of selected model) for each version whereas in incremental model, the user req. defination, ~~sss~~ SRS and system design activities are factored out of the sequence of incremental deliveries and occur only once, at the outset of project (i.e. only the required features in the form of increments are implemented skipping other phases), therefore sum of all increments represents the totality of system.

b) Draw the diagram of iterative enhancement model and also describe the types of situations where it might lead to difficulties. [4+4] P.T.O. 2018E



It might lead to difficulties when:

→ it ~~needs~~ dont have good planning and design

→ there isn't a clear and complete defination of whole system before it can be broken down and built in cremently.

→ ~~total~~ cost hishes than waterfall.

(c) How iterative enhancement model is helpful during maintenance? Explain the various stage cycles of this model. 2022E [4][CO6]

It is helpful during maintenance because it allows for continuous improvement and refinement of the system. As system is used and feedback is collected, it can be iteratively enhanced to address any issues or add new features.

Date : / /
Page No.

Its phases :

1.) Requirement Gathering : Requirements are collected and documented

2.) Design : Sys. design is based on the gathered requirements like architecture, user interface, and other design element.

3.) Implementation and Testing : The system is developed, implemented acc to design specifications and tested to make sure it meets required specifications and functions correctly.

4.) Evaluation : The system is evaluated to identify any areas for improvement and enhancement.

5.) Feedback and Enhancement : Based on the evaluation, feedback is collected from the customer / user. This feedback is used to make enhancements / improvements in system.

1 What are the two characteristics of spiral model, which distinguishes it from other process model? Give an example of software process where: 2010M
[a] spiral model is suitable
[b] spiral model is not suitable
Justify your answer for the above cases. 4

Distinguishing characteristic :

1. RISK DRIVEN : It places strong emphasis on risk management
2. ITERATIVE : the s/w is developed and released in increments building on prev iteration

a.) SPIRAL SUITABLE : Developing a new operating system
Spiral model is good choice for this case as its a complex and risky project. Risk mgmt and iterative nature will help to reduce the risk of failure and to ensure that the OS meets the need of the customer

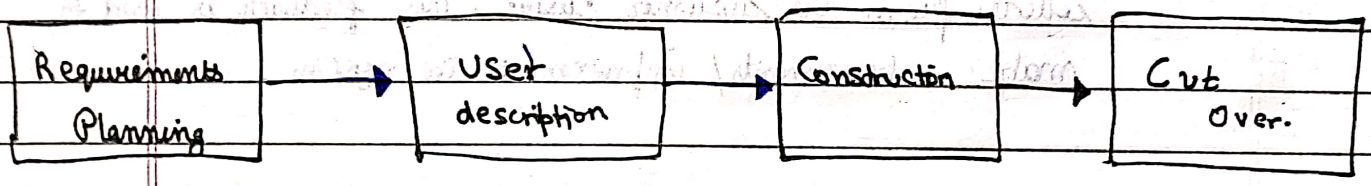
b.) SPIRAL NOT SUITABLE : Developing a simple website
Spiral isn't necessary for developing simple website as its a low risk project. It would be an overkill and wastage of resources

Date : / /
Page No.

(b) Describe the rapid application development (RAD) model. Discuss each phase in detail. 2022E [4][CO1]

It is developed by IBM in 1980 where user involvement is essential from requirement phase to delivery of product which ensures the involvement of user's expectations and perspective in requirement elicitation, analysis and design of the system.

The process is started by building a rapid prototype and is given to user for evaluation. The user feedback is obtained and prototype is refined. This process continues till the requirements are finalised.
The four phases in this model:



i) Requirements planning phase: Requirements are captured using any group elicitation technique. only issue is the active involvement of users for understanding the project.

ii) User Description: Joint team of developers and users are constituted to prepare, understand and review the requirements. The team may use automated tools to capture information from the other users.

iii) Construction Phase: This phase combines the detailed design, coding and testing phase of waterfall model. Here, we release the product to customer. It is expected to use generators, screen generators and other types of productivity tools

iv) Cut-over phase: The phase incorporates acceptance testing by the users, installation of the system, and user training.

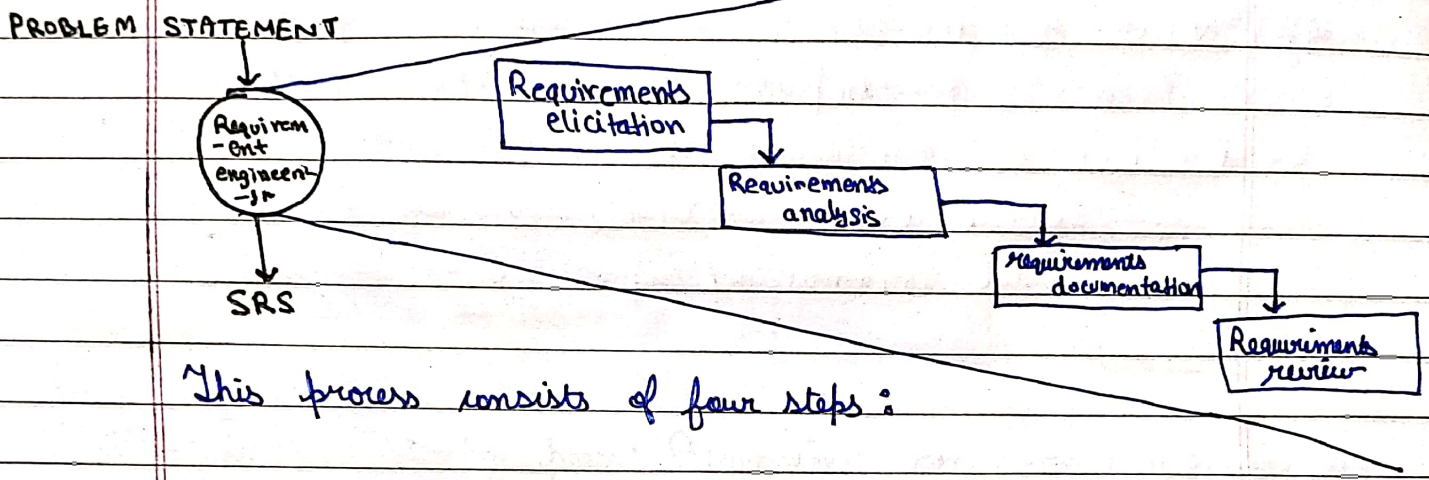
Date : / /
Page No.

UNIT-2 : SRS

7 Write short note (ANY TWO) on following:
(a) Requirement Engineering 2018E

Requirement engineering refers to the process of defining, documenting, and maintaining requirements in the engineering design process.

Requirements engineering produces one large document, written in natural language, contains a description of what the system will do without describing how they do it.



This process consists of four steps :

i) Requirements Elicitation : Involves identifying requirements through customer input and existing system process.

ii) Requirements Analysis : Follows elicitation, identifying inconsistencies, defects and resolving conflicts.

iii) Requirements Documentation : Its the end production of info doc for system design called software requirements specification (SRS)

iv) Requirements Review : req. verification is carried out to improve the quality of SRS. Its a continious process incorporated in elicitation, documentation, analysis.

Date : / /
Page No.

b) How Functional Requirement is different from Non-Functional Requirements? 2020E

Software requirements are broadly classified into:

i) Functional Requirements: These are related to the expectations from the intended software. They describe what the software has to do. They are also called product features.
Sometimes, functional requirements may also specify what the software has to do.

ii) Non-Functional Requirements: These are mostly quality requirements that stipulate how well the software does what it has to do. They include specs of desired performance, availability, reliability, usability and flexibility. For developers these are portability, testability and maintainability.

The functional requirements are directly related to customer's expectations and are essential for the acceptance of product. However, non-functional requirements may make the customer happy and satisfied.

Eg:) If we want to develop a secure system, security becomes essential to us. It is a non-functional req. apparently, but when design is carried out, it may have serious implications. These implications find place in functional req. category of SRS doc.

[b] Who are the stakeholders of SRS? 2010M

The term stakeholder is used to refer to any one who may have some direct or indirect influence on system requirements. It includes end users who will interact with the system and everyone else in an organization who will be affected by it.

INTERNAL STAKEHOLDER	EXTERNAL STAKEHOLDER
→ Project Manager	→ Customer
→ Project dev. team	→ Direct user
→ Company	→ Indirect user
→ Partners	→ Supplier
→ investors	→ Legal authorities

Date : / /
Page No.

1. Answer the following questions briefly:

a) Discuss any two requirements elicitation technique with suitable example. 2019M

Requirement Elicitation is the process to find out the requirements for an intended software system by communicating with client, end users, system users, and others who have a stake in Sof sys De. The techniques are:-

[b] Discuss any two techniques for requirements elicitation. 2010E 6

→ INTERVIEWS

A conversation where both parties would like to understand each other. They can be of two types:

i.) open ended: there is no present agenda, context free questions can be asked to understand the problem, to have an overview of situation

ii.) Structured: agenda is pre-set and Qs asked are short and simple

→ BRAINSTORMING

It is a kind of group discussion which lead to ideas very quickly and help to promote creative thinking. (its being used in most ors nowerdays) All participants are encouraged to say whatever idea come to their mind and no one will be criticized for any idea no matter how goofy it seems.

2 [a] Discuss facilitated application specification technique for requirement elicitation. 2018E (7)

→ FAST (facilitated application specification technique)

This approach encourages the creation of joint team of customer and developer who works together to understand correct set of req. Everyone is asked to prepare a list of:

→ What surrounds the system.
→ Produced by the system
→ used by the system
→ List of services, constraints, performance criterion

Activities during fast session:

→ Participants present their lists of objectives, services, constraints and performance

Date : / /

b) Differentiate between Verification and Validation. 2019M

3 [a] Distinguish between software verification and software validation. When are verification and validation performed during the software life cycle? 2018E (7)

(ii) Verification and validation. 2010E

c) Differentiate between Verification and Validation? 2020E

	VERIFICATION (White Box)	VALIDATION (Black Box)
→	Verification means "Are we building the software right?"	Validation means "Are we building the right software?"
→	Verification is the static testing and code execution is not included	Validation is the dynamic testing and code execution is included
→	Methods : reviews, walkthroughs, inspections, desk-checking	Methods : Black Box testing, white Box testing, non functional testing.
→	It checks whether the s/w conforms to requirements and specifications	It checks if s/w meets the req. and expectation of customer.

verification : Every stage of development
Validation : at the end of module, or after s/w is built

FAST cont...

→ Combine lists by removing redundancies and adding new ideas

→ Discuss and finalize consensus lists with the facilitator

→ Divide the team into subgroups to develop mini-specifications.

→ Present mini-specifications for discussion and make adjustments.

→ Maintain an issues list for unresolved matter, attendees create validation criteria lists and form a consensus list of same.

→ Subteams may draft complete specifications using input.

FAST aids in early requirement elicitation but not a complete sol^n.

Date : / /
Page No.

[b] What is meant by SRS document? What are the characteristics of a good SRS document? 2018PE (7)

A software requirements specification (SRS) is a document that describes what the software will do, how it will be expected to perform. It also describes the functionality that product needs to fulfill the needs of all stakeholders.

Thus, it is a formal report, which acts a representation of the software having a detailed desc. of software system to be developed with its functional and non functional requirements. It contains all logical details like how the software will look like, which lang will be used, database design, modular design etc.

Characteristics of Good SRS:

1. CORRECT : All requirements must accurately represent what S/w should do.
2. UNAMBIGUOUS: Each requirement should have a single, clear interpretation to avoid confusion.
3. COMPLETE: It should cover all significant aspects of functionality, performance, constraints and confusion.
4. CONSISTENT: No conflicts or contradictions should exist among the specified requirements.
5. Ranked for importance and/or stability
6. VERIFIABLE : Every requirement should be tested to ensure it can be confirmed as met.
7. MODIFIABLE: The SRS should be allow easy and consistent updates without causing errors.
8. TRACEABLE: Requirements should be clearly sourced and referenced for future dev. and maintenance.

b) What is the purpose of SRS? List down the characteristics of a good requirement. 2018S [4]

(c) List five desirable characteristics of a good SRS document. Discuss the relative advantage of formal requirement specification. List the important issues, which an SRS must address. 2022E [4][CO2]

Date : / /
Page No.

ADVANTAGE

Formal specifications have several advantages over informal one especially in that they are mathematically precise. They tend to be more complete than informal specifications because the formality tends to highlight any incompleteness, which might go unnoticed.

Important issues which must be addressed by the SRS:-

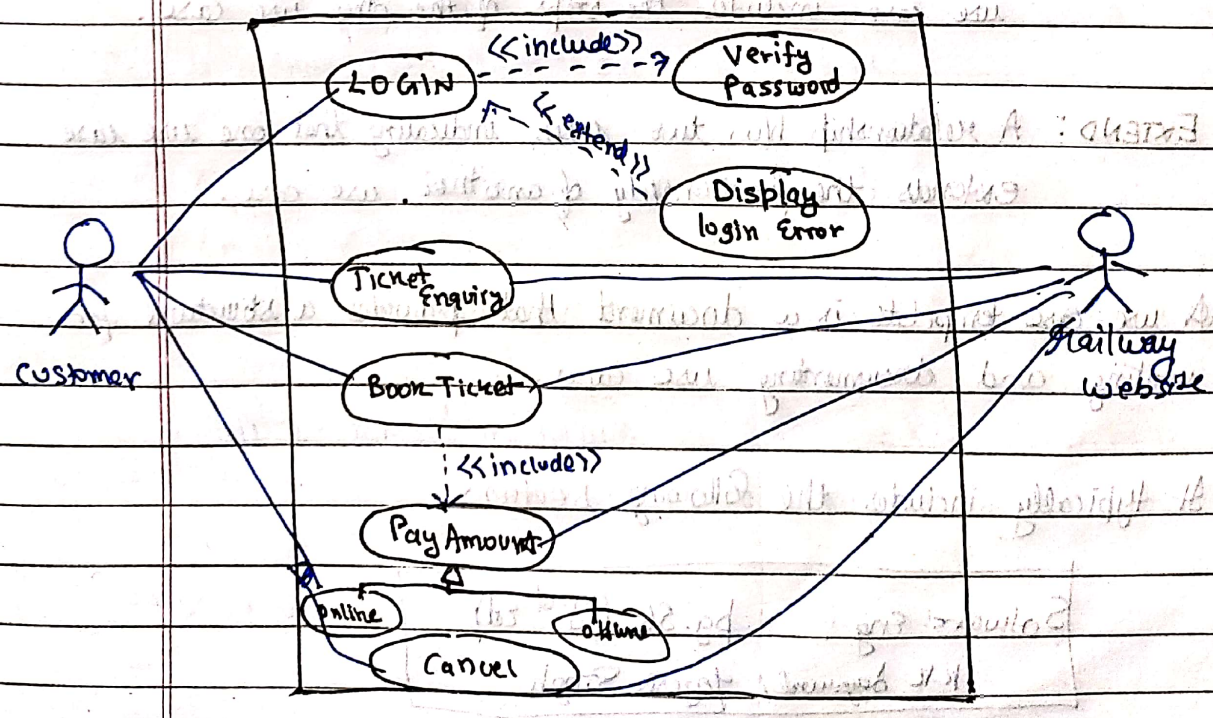
1.) **Functionality**: What exactly will software do?

2.) **Performance**: Details of speed, response time, recovery time etc.

3) **Attributes**: Portability, correctness, maintainability, security etc.

4.) **External Interfaces**: How does s/w interact with external entities.

Q3.a) Consider railway reservation system. Identify at least two actors and five use cases of the system outlined above. Also write the use case description of reservation of train. 2018S [4]



Date : / /
Page No.

5[a] Explain various concepts used in Use Case Diagram. Discuss the importance of use case diagram in requirement engineering. 2016C-7

4 a) Explain the use case approach of requirements elicitation and also discuss the use-case template. 2018B.

Use case approach is a technique for gathering and documenting the requirements of software system from the perspective of its users. It is based on the idea that a s/w system can be described as a set of use cases, each of which represents a specific task that a user can perform with the system.

Various concepts used in use case diagram are :-

→ ACTOR : An external entity that interacts with the system, such as a user, other software or hardware devices.

→ USE-CASE : A sequence of steps that an actor takes to achieve a specific goal with the system.

→ SYSTEM BOUNDARY : A box that represents the system being modelled.

→ ASSOCIATION : A line b/w an actor and a use case, indicating that the actor can initiate and perform the use case.

→ INCLUDE : A relationship b/w two use cases, indicating that the one use case includes the steps of the other use case.

→ EXTEND : A relationship b/w two cases, indicating that one use case extends the functionality of another use case.

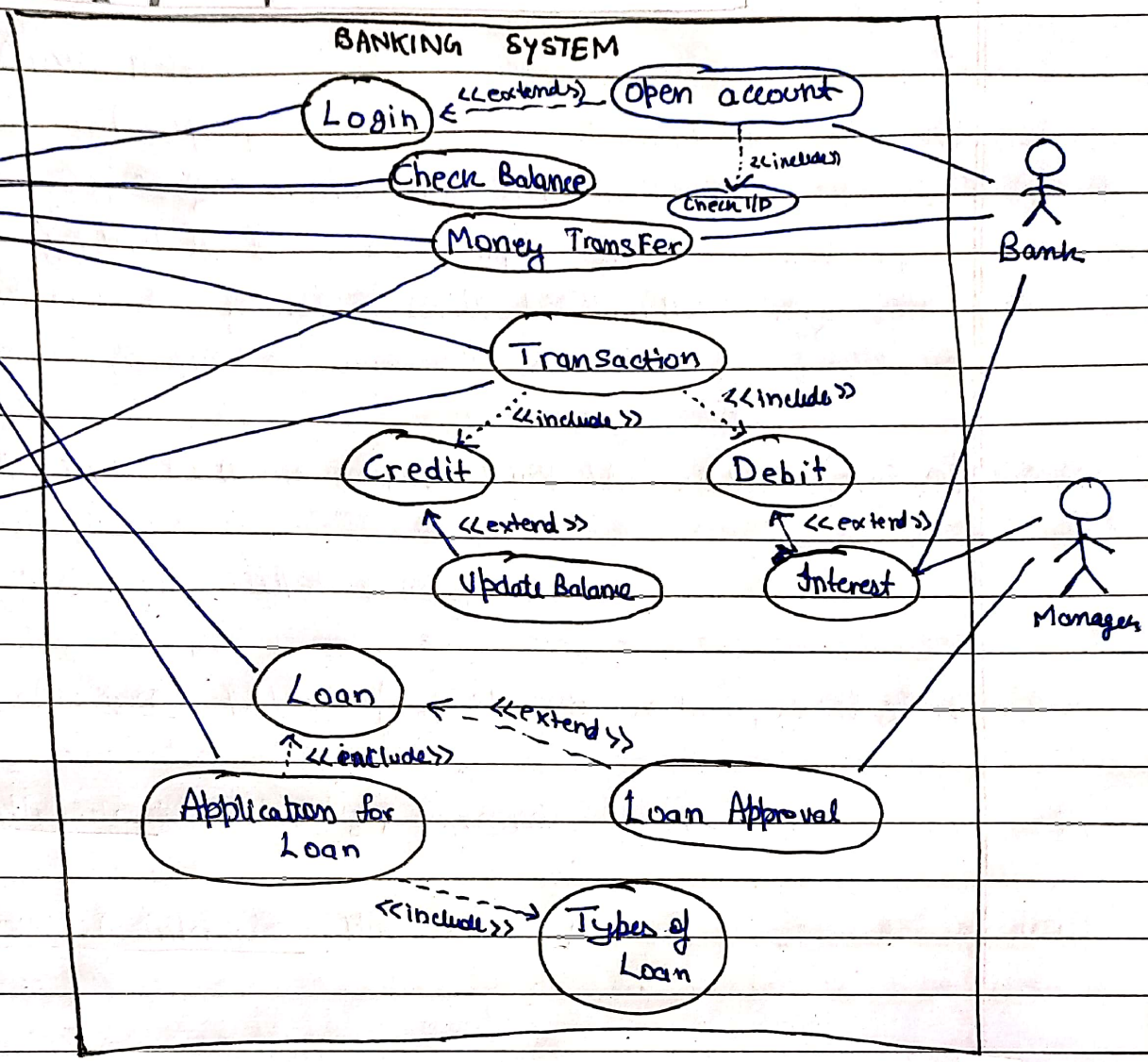
A use-case template is a document that provides a structure for writing and documenting use cases.

It typically includes the following sections

Software Eng. pg. 56 (3rd Ed)
KK Aggarwal, Yogesh Singh

Date : / /
Page No.

3 Draw the use case diagram for Banking system. What are the different dependencies in Banking system? 2010M 5



Dependencies in Banking system.

→ Withdraw cash depends on check balance
→ Money transfer " "
→ Open a/c depends on Verify Identity
→ Loan depends on Loan Approval
etc.

Date : / /
Page No.

2 Answer the following questions to the point. (ANY THREE)

[a] Why requirements are hard to elicitate? 2010M

6[a] Explain the problems in the formulation of requirement for any software project. 2010E 4

Requirement elicitation is challenging due to its communication intensive nature. The difficulty arises from the fact that real requirements are in user's minds, requiring effective collaboration between customers and developers to put into words.

The disparity in mindset, expertise and vocabulary between these groups leads to communication gaps, potential conflicts and inconsistencies. The process involves asking questions, understanding user expectations and navigating background of participants in the domain.

Various elicitation techniques exist but selecting appropriate requires insight into stakeholder needs, which is often lacking, leading to reliance on familiarity or personal preferences.

UNIT-3.2 (DFD)

Date : / /
Page No.

3. What are the guidelines to draw DFD? What constrains must be observed while creating leveled DFD? 2009M 5

4[a] What rules should be observed while drawing DFD? 2010M 2

These observations are important for DFD:

1. All names should be unique. This makes it easier to refer to items in the DFD.
2. Remember that DFD is not a flowchard. Arrows in a flow chart represent order of events; arrow in DFD represent flowing data. A DFD does not imply any order of events.
3. Supress logical decisions. There arent multiple paths of which one is taken in DFD as it implies ordering of events which dont make sense.
4. Do not become bogged down with details. Defer error conditions and error handling until the end of analysis

Constraints:

- Each level of the DFD should represent a higher level of detail than the previous one. i.e. top-lvl should give a general overview while lower lvl provide specific info about its components and processes
- Each lvl should be complete and self contained
- Each data flow should be named and labelled with function.
- All the data-flows and processes should be connected to each other
- The DFD should be balanced
  i.e. no. of data inputs = no. of data outputs.

Q4.Consider the following case study: 2018M

There is always a need of an efficient system that can generate and maintain the transaction receipts, inventory reports and sales records in the big retail businesses like Restaurants, Gas Stations, Convenience Stores, Retail Stores (Walmart, Loblaws) etc. The purpose of this system is to automate the checkout process, providing faster services and better purchasing experience for customers. Moreover, the system can keep the record of sales in the database operated on cloud which can be updated, modified, deleted as whenever is the case or required.

Point of Sales(POS) is a computerized system managing sales and performs simple day to day operations in the retail stores. POS terminal building involves interfacing printer, bar code scanner, computer, Debit/Credit Card Reader, keyboard, touch screen input, weighing machine and software on a CPU to run operations in a store. When the customer arrives with all the items to be purchased, the cashier scans item, pre-entered price shows up and the items are added with all information to the current running transaction. The barcode reader after scanning sends signal to retrieve name and price of item from the backend catalog and further interacts with inventory system for any discrepancy. The display interfaced will show the individual price while scanning of each item and total price to the customer including all sales tax. The customer will decide the type of payment viz. cash, credit or debit card, coupons, Points/Promotion Card or cheque. On completion of successful transaction (payment done), the POS software updates the inventory and generates a receipt for customer and transaction record for Merchant. The cashier handovers the receipt record to the customer. In addition, the system requires user log in to use POS software. Cashiers, Managers and administrators are the users that can access POS. Further, the protected and confidential system management functions can only be accessed by administrators for security purposes. These systems should be fault tolerant and have the capability to run offline sales and cash management when remote services are temporarily unavailable. It is assumed that self-checkout functionality is missing in the system described and always involves cashier for the transactions. Design following diagrams for the above case study:

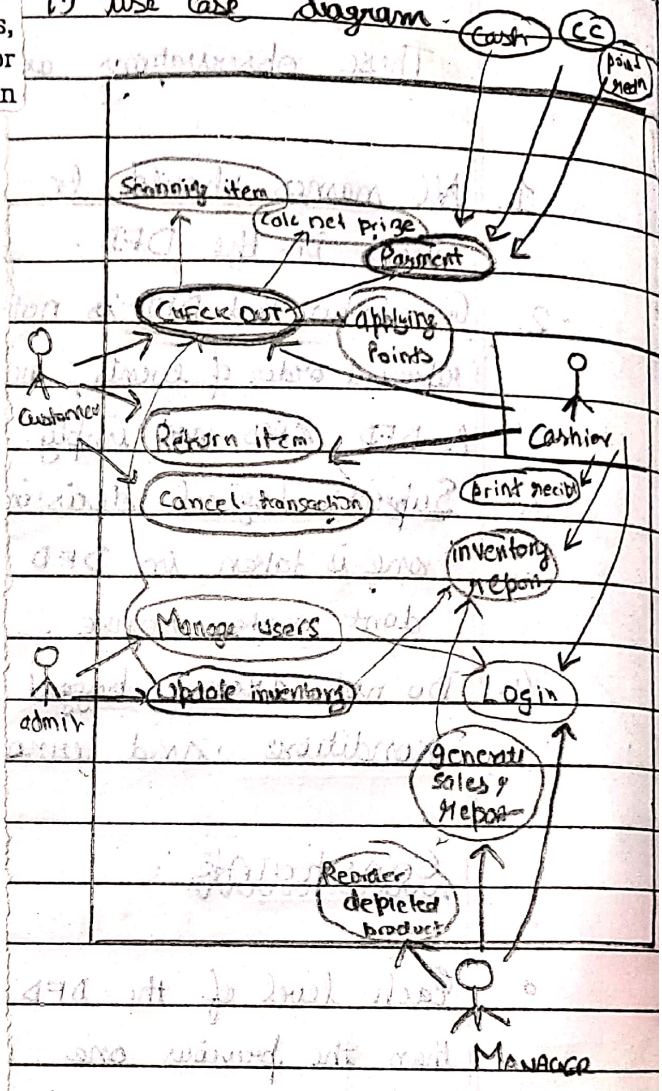
i) Use Case Diagram
ii) 1-level DFD

[5*2=10]

END

Date : / /
Page No.

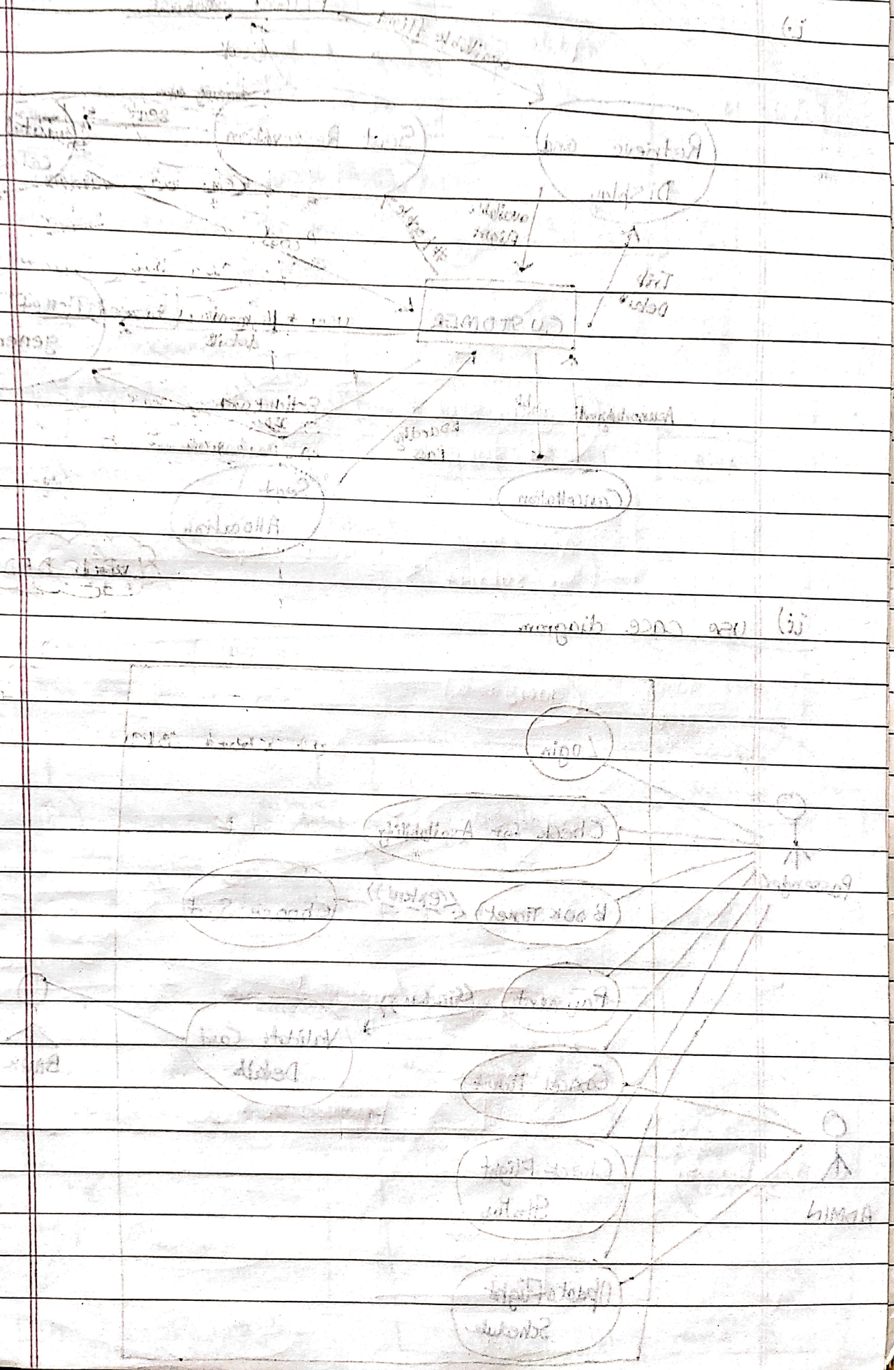
i) use case diagram



ii) 1-level DFD

Date : / /
Page No.

[d] What are the logical constraints while drawing ERD? 2010M



Date : / /
Page No.

A data center, involved in the design of software products, has two type of employees; Technical and Administrative. Technical staffs are again of three types; Data-entry-operator, Programmer and System Analyst. A data-entry-operator is judged by his speed-of-entry in key depressions per second. A programmer has an assigned language in which he normally writes his programs (like C, COBOL, etc.). A system analyst has a field of specialization (like, API-designer, System software specialist, Database consultant etc.). In general, an employee is identified by his employee-no. Besides this, for each employee, the company maintains the following information,

(name, address, date-of-birth, date-of-joining and montly-salary)

The company also maintains information on the number of projects where each technical staff is involved. If technical staffs is not involved in any project, this field should contain zero. Each project is identified by a unique Project-no. Each project also has a unique name. The other information maintained for each project are. 2005M

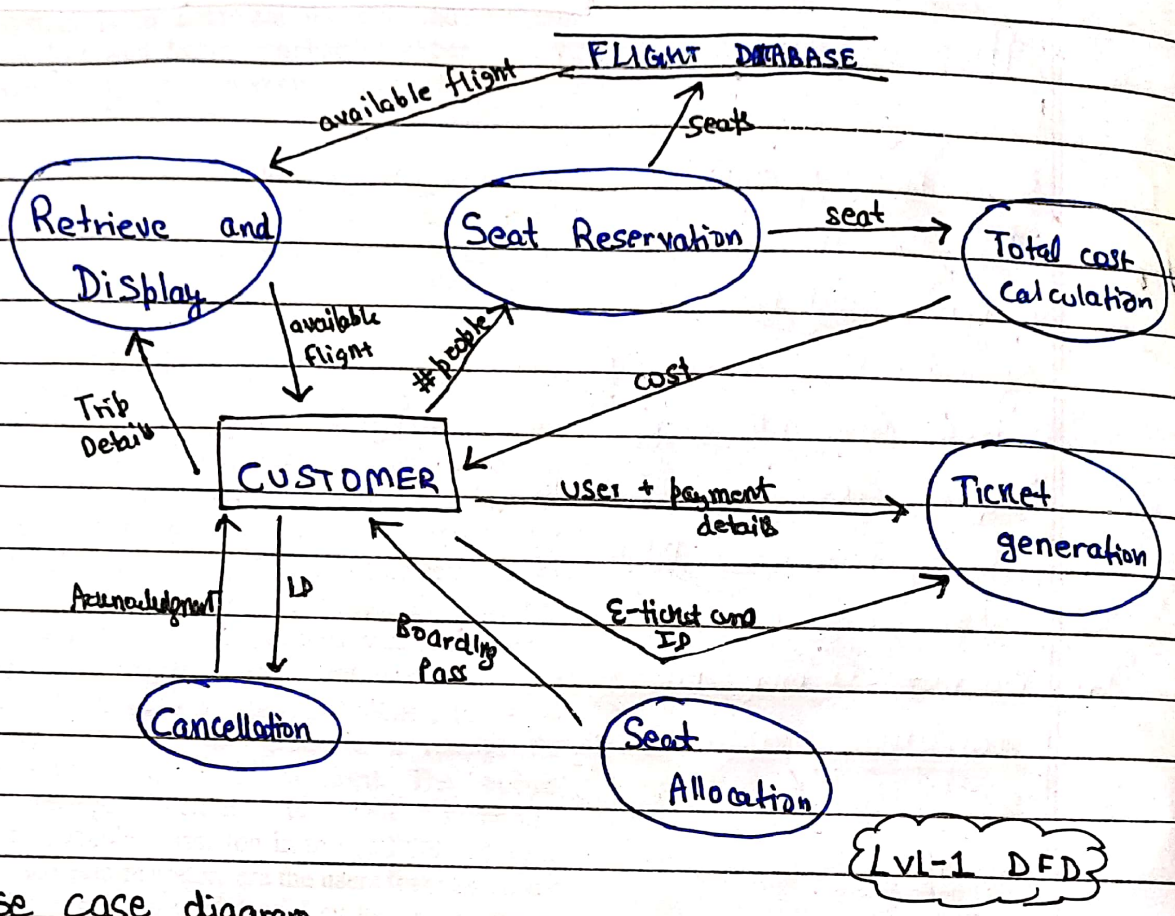
(project-name, budget, starting-date, expected-date-of-complétion, organization-name, organization-address).

The organization-name and address identify the organization that has given the project to the company. Each technical staff may be associated with one or more projects or with no project at all. One of the technical staff may not be the project-leader of any project. Draw an ER-diagram. 2005M 6

Date : / /
Page No.

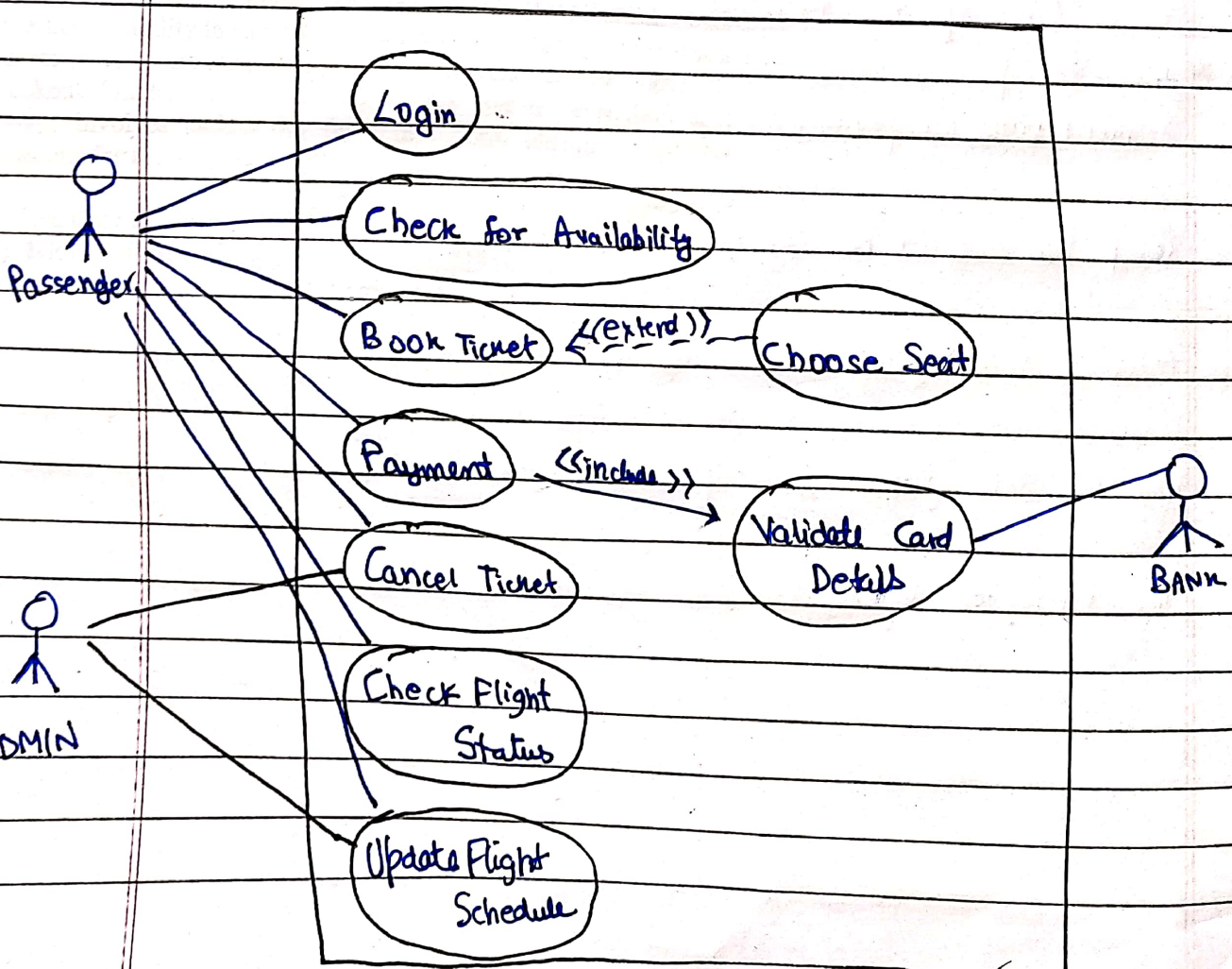
Q1. Consider the problem of airline reservation system and design the following: 2022M
(i) DFD Diagram (Upto Level 1)
(ii) Use Case Template and Diagram [5][CO2]

i)



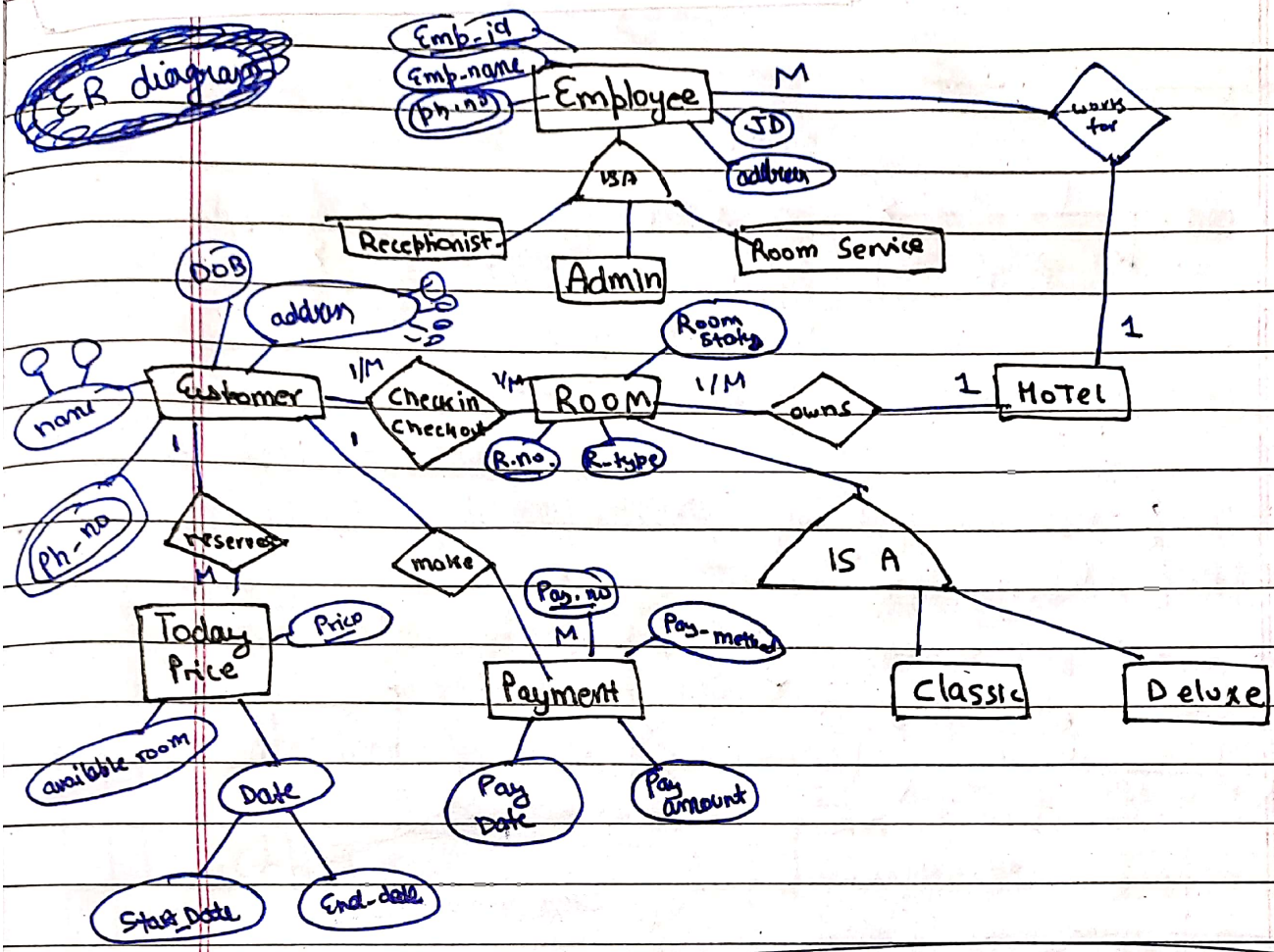
LVL-1 DFD

ii) Use case diagram

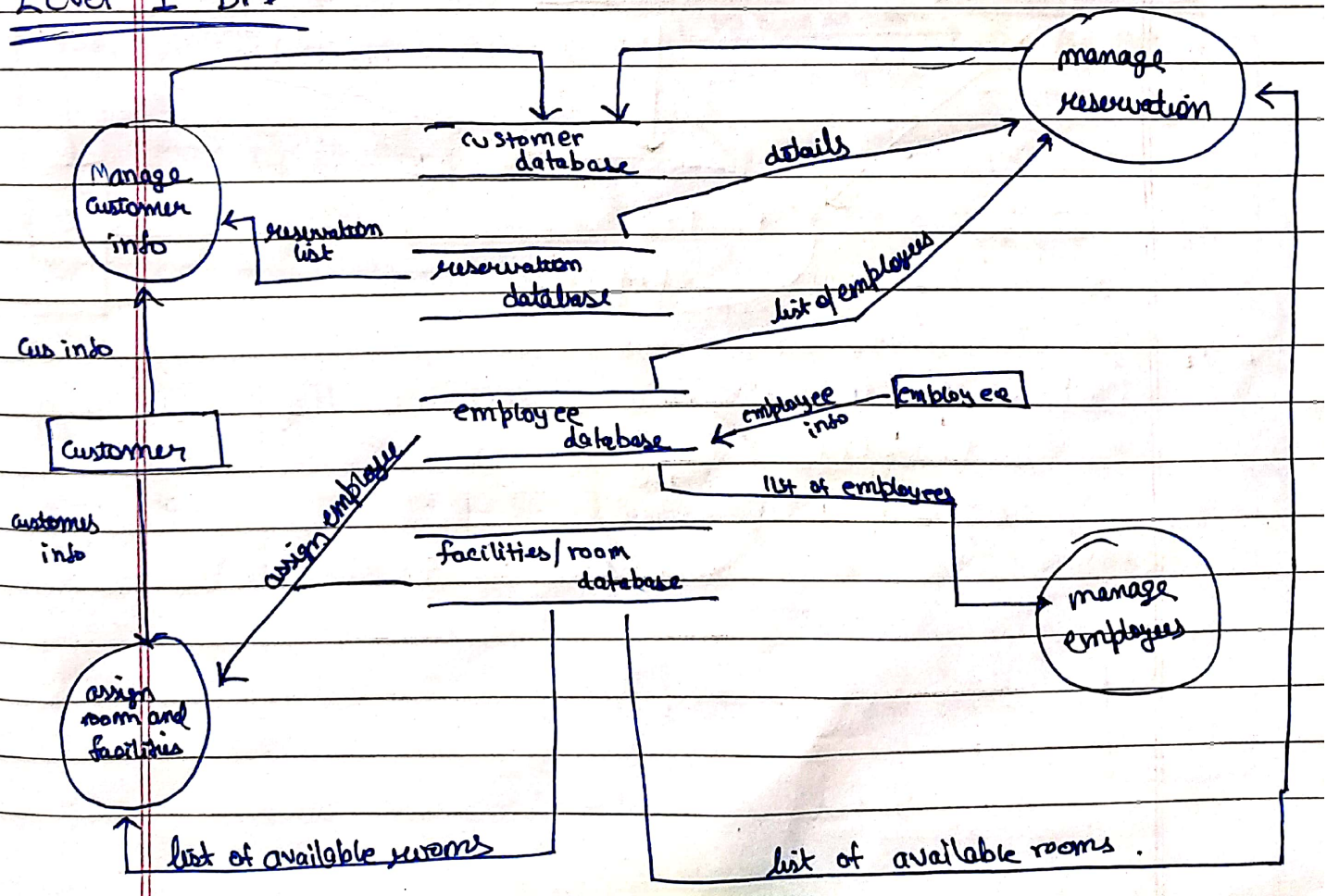


2. Draw ER diagram and Level 1 DFD for automation of hotel management system also mention the requirements, which you have considered for a typical hotel management system. 2019M [3+2=5]

Date : / /
Page No.



Level -1 DFD



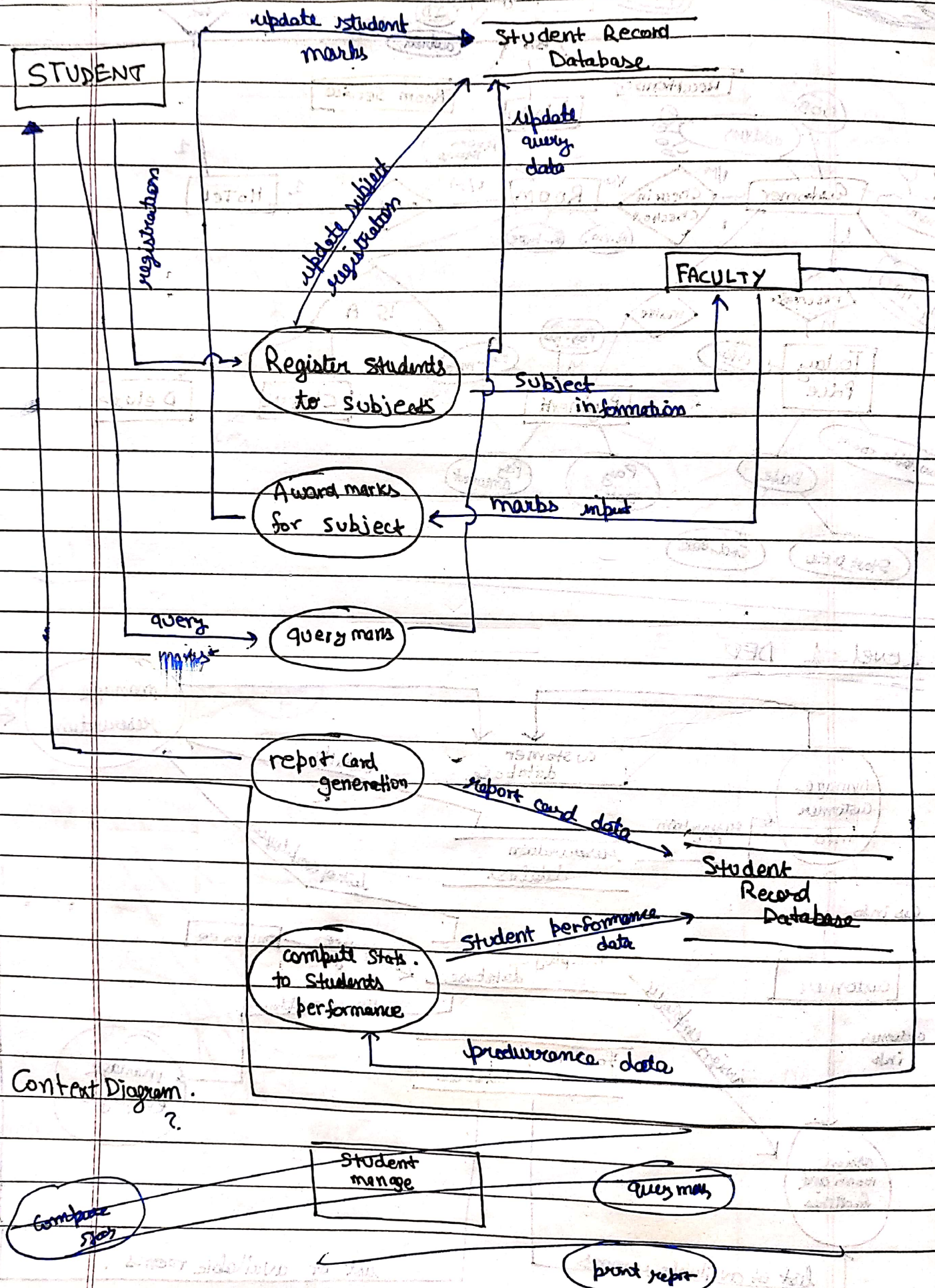
Date : / /

Page No.

[b] Draw a labeled DFD model (context diagram and level 1 DFD for a student academic management software that should support the following features : 2018E (old)

- Register students to subjects
- Award marks for subjects
- Query marks
- Print report cards
- Compute statistics of student performance. (7)

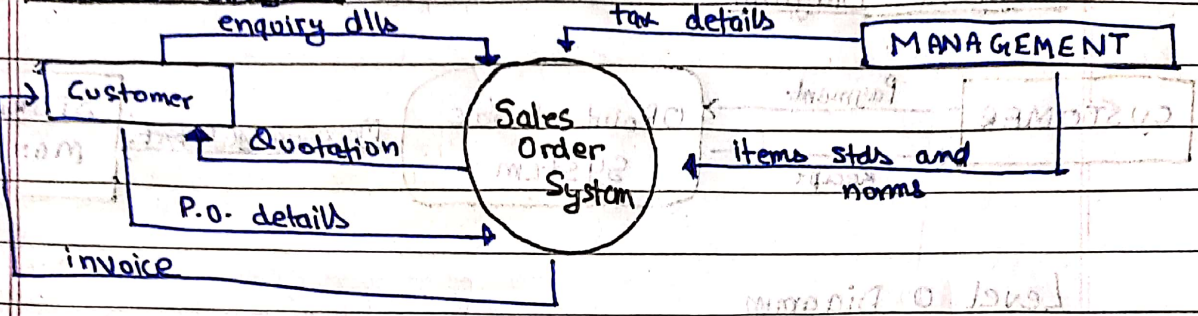
DFD Lvl 1



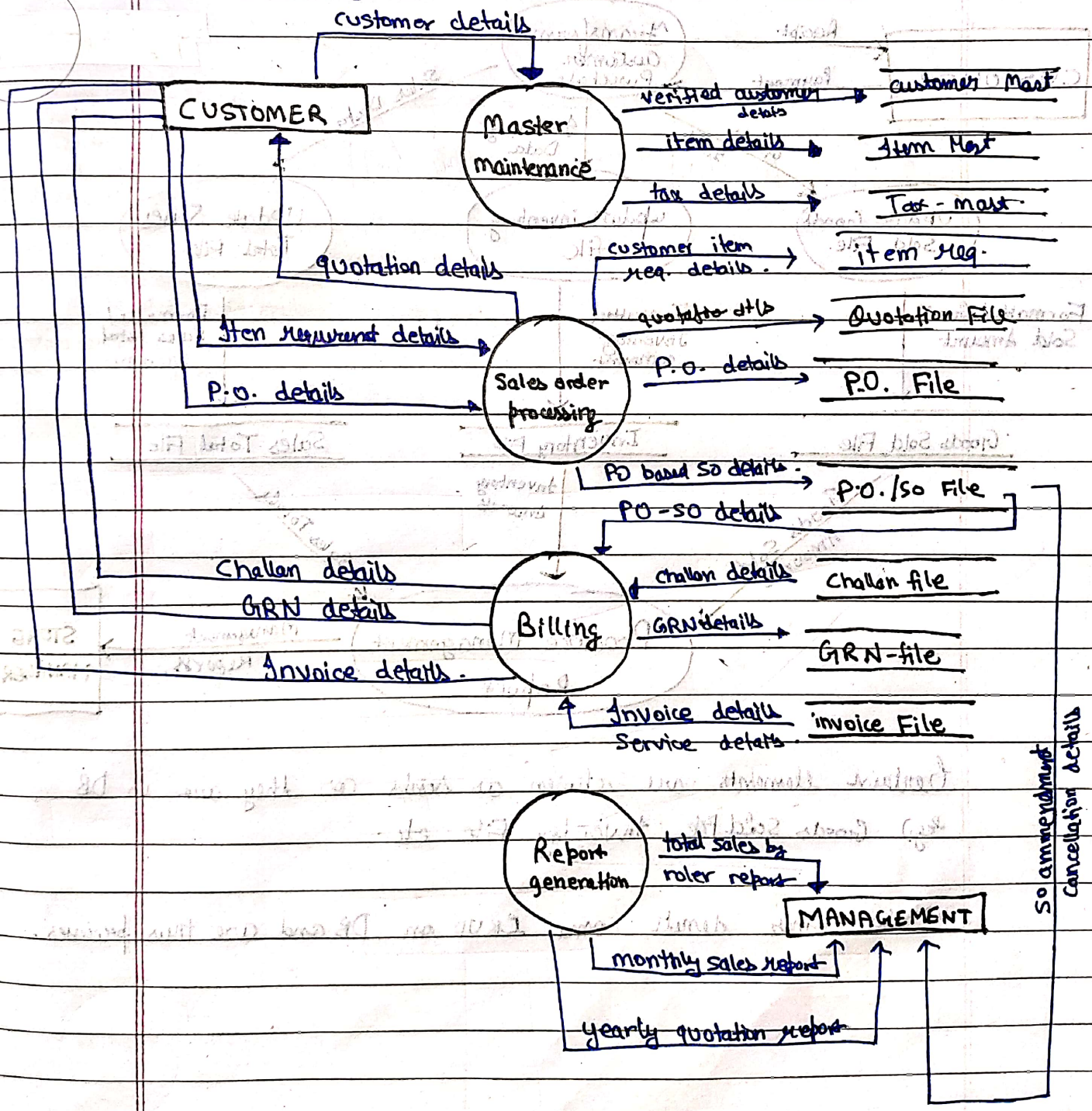
Date : / /
Page No.

3 a) Consider an order processing system that receives the orders through its sales man and sends goods by a transport and maintain the track order for every order including order receiving dates, delivery dates, invoice details and so on. Use your assumption to add more functionality to the system and develop zero level DFD and one level DFD to model this system. M 2018 E

Level 0 diagram



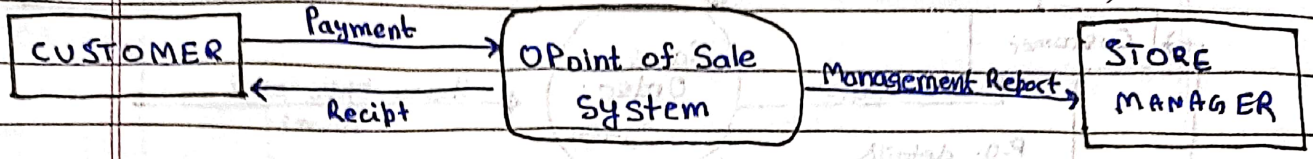
Level 1 Diagram



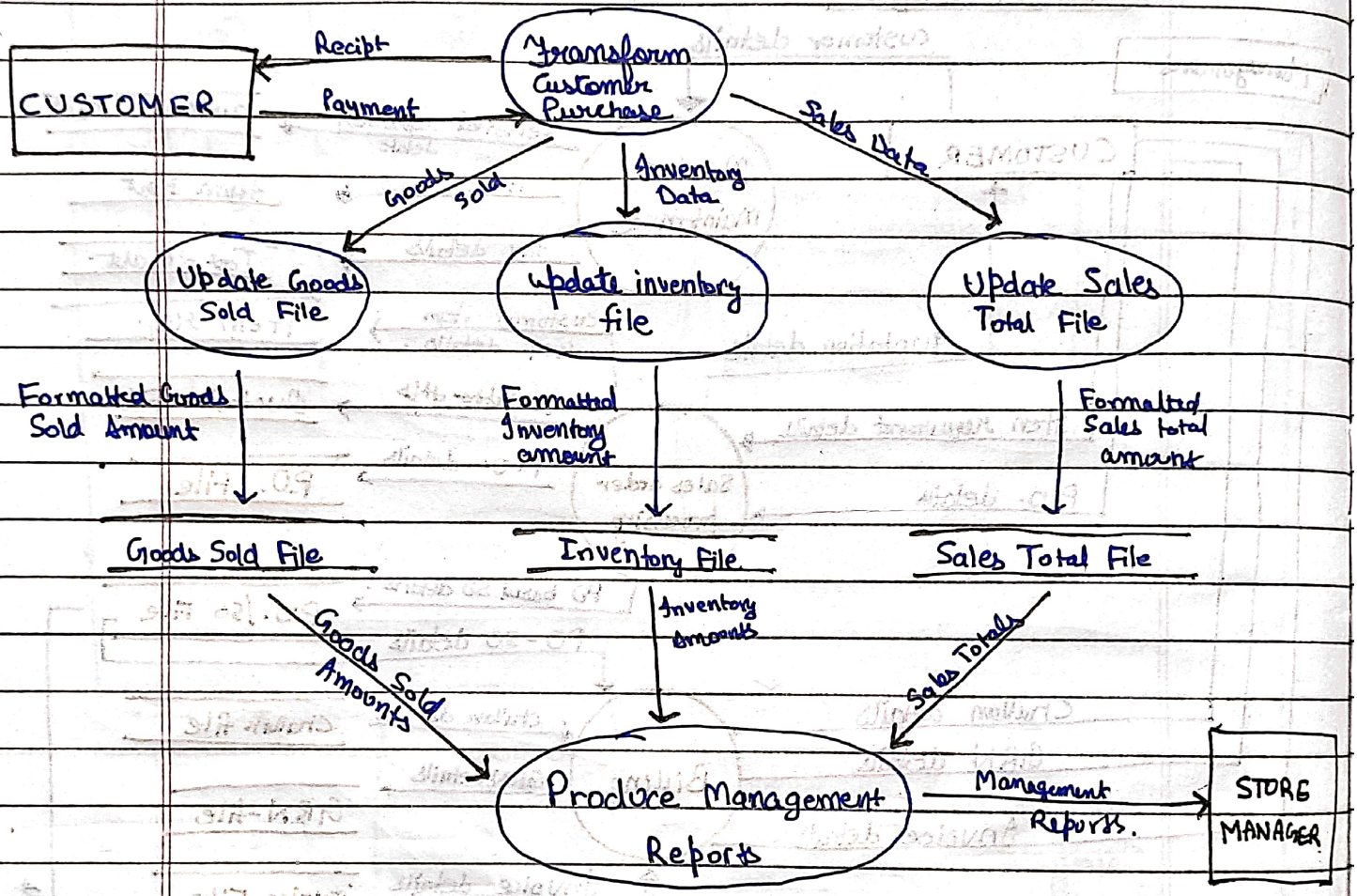
Date : / /
Page No.

3 a) Using the example of a retail clothing store in a mall, list relevant data flows, data stores, processes, and sources/sinks. Observe several sales transactions. Draw a context diagram and a level-0 diagram that represent the selling system at the store. Explain why you chose certain elements as processes versus sources/sinks. M 2020E

Context Diagram



Level 0 Diagram



Certain elements are chosen as sinks cos they are in DB
eg:) Goods Sold File, Inventory File etc.

While other denote some CRUD on DB and are thus processes.

4 Write short notes on any *TWO*:

[a] Gantt chart and Pert chart 2005M OUT OF SYLLABUS

Q4. Draw the context diagram, 1-level DFD and Structured Chart for the following library management system:

2018S

1) Issue of Books:
   a) A student of any course should be able to get books issued.
   b) Books from general section are issued to all but book bank books are issued only for their respective courses.
   c) A limitation is imposed on the number of books a student can issue.
   d) A maximum of 4 books from book bank and 3 books from general section per student are allowed.
   e) The books from book bank are issued for the entire semester while books from general section are issued for 15 days only.
   f) The software takes the current system date as the date of issue and calculates the corresponding date of return.
   g) A bar code detector is used to save the student as well as the book information.
   h) The due date for return of the book is stamped on the book.
2) Return of Books:
   a) Any person can return the issued books.
   b) The student information is displayed using the bar code reader.
   c) The system displays the student details on whose name the books were issued as well as the date of issue and return of the book.
   d) The system operator verifies the duration for the issue and if the book is being returned after the specified date, a fine of rupee 1 is charged for each day.
   e) The information is saved and the corresponding updates take place in the database.
3) Query Processing:
   a) The system should be able to provide information like:
      - Availability of a particular book.
      - Availability of any particular book.
      - Number of copies available of the desired book.

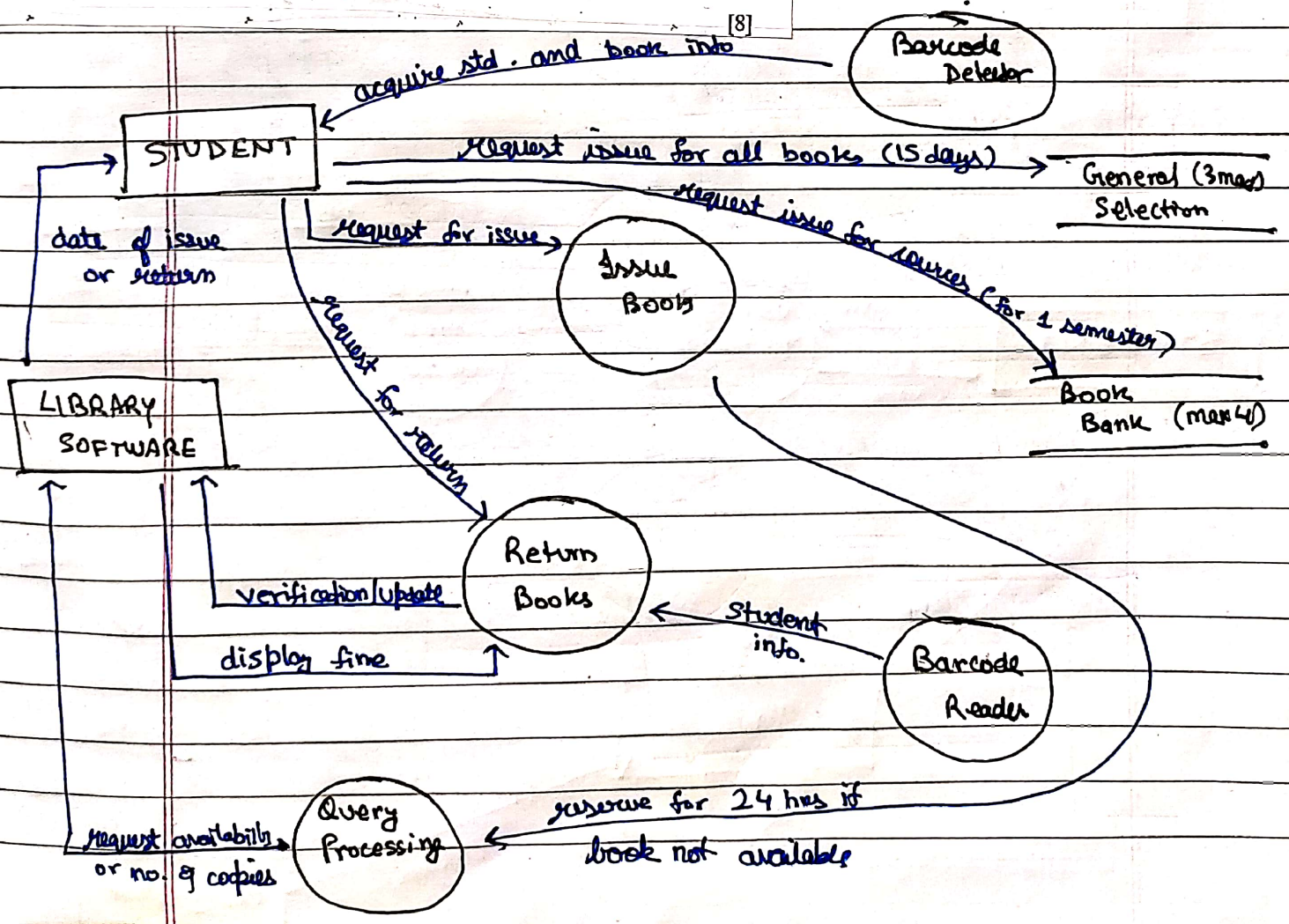
The system should be able to reserve a book for a particular student for 24 hrs if that book is not currently available.

[8]

Date : / /
Page No.

Structured Chart

Level 1 DFD



[c] Data Dictionary 2005M

[c] What is the role of Data Dictionary? 2010M

(d) Describe the contents and model of a data dictionary. 2018M

Date : / /

Page No.

A data dictionary is a file or a set of files that includes a database's metadata. It holds records about other objects in the database, such as ownership data, data relationship to other objects, and other data. The data dictionary is an essential component of any RDB.

So, these are repositories to store info about the data in DFDs. At requirement stage, they should atleast define customer data items to ensure that the customer and developer use the same definitions and terminologies. This info includes :-

- → Name of data item
- → Alias (other names for item)
- → Description / Purpose
- → Related data items
- → Range of values
- → Data structure def^n / form.

DATA DICTIONARY NOTATION (HOPER)

NOTATION	MEANING
$x = a+b$	x consists of data elements a and b
$x = [a/b]$	x consists of either data element a or b
$x = a$	x consists of an optional data element a
$x = y\{a\}$	x consists of y or more occurances of data element a
$x = \{a\}z$	x consists of z or fewer occurances of data element a
$x = y\{a\}z$	x consists of some occurances of de. a which are b/w y and z

Role / uses

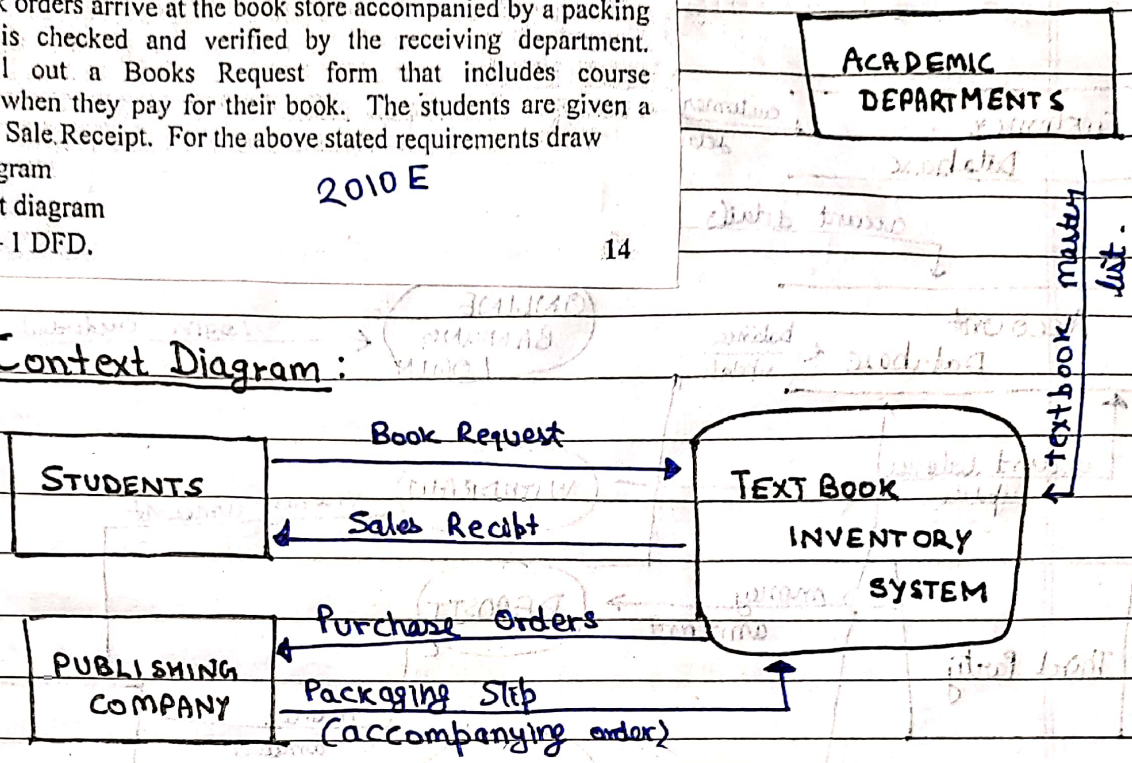
- → Create an ordered listing of all data items.
- → Create an ordered listing of a subset of data items.
- → Finding a data item name from a description.
- → Design the software and test cases.

2 The purpose of Text Book Inventory system at a campus broke store is to supply text books to students for classes at a local university. The university's academic department supply initial data about courses, instructors, text books and projected enrollment to the book store on a Test Book Master List. The book store generates a form No. 11 purchase order, which is sent to publishing company supplying text books. Book orders arrive at the book store accompanied by a packing slip which is checked and verified by the receiving department. Students fill out a Books Request form that includes course information when they pay for their book. The students are given a cash register Sale Receipt. For the above stated requirements draw

(i) ER diagram
(ii) Context diagram
(iii) Level – 1 DFD. 14

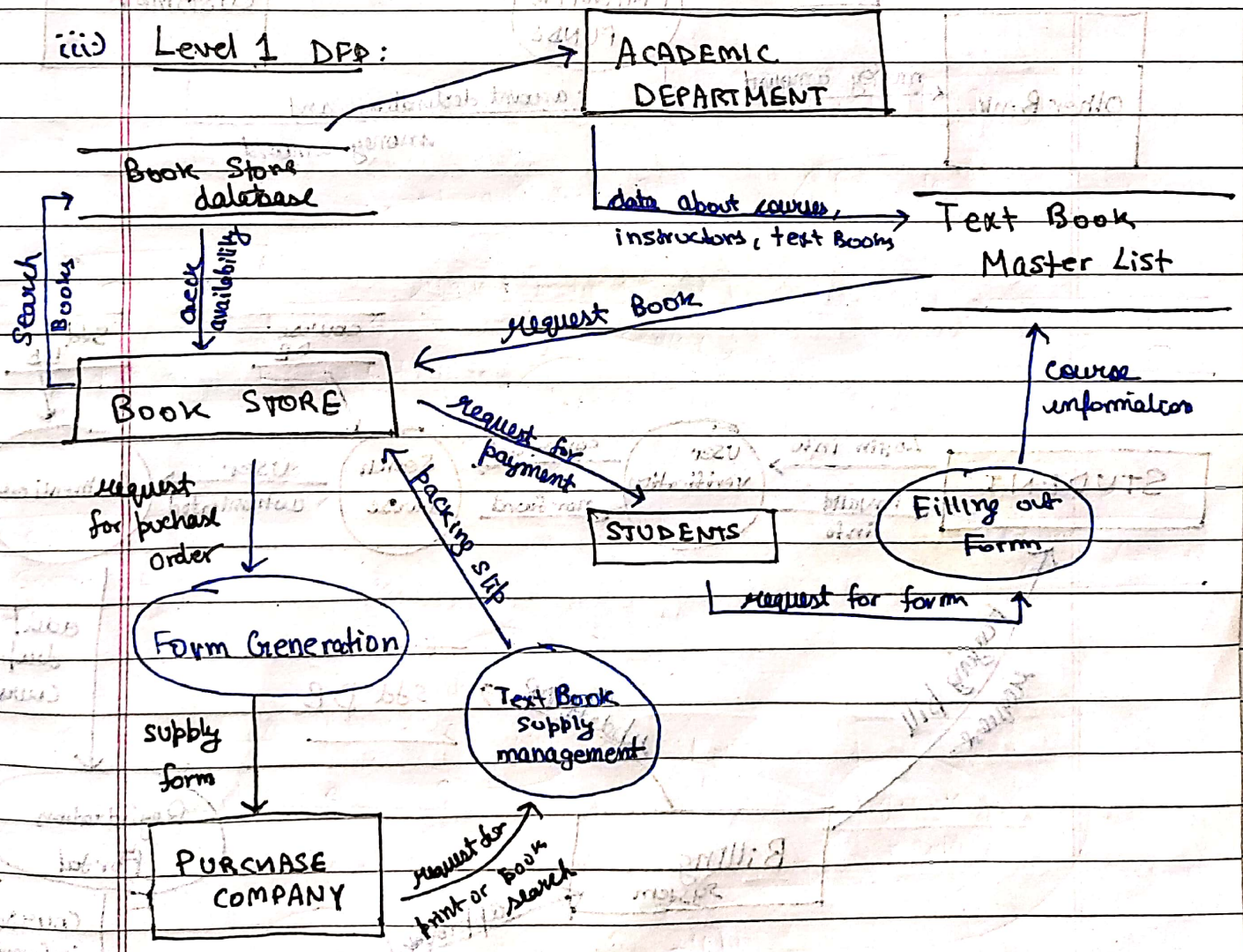
2010 E

Date : / /
Page No.



ii) Context Diagram :

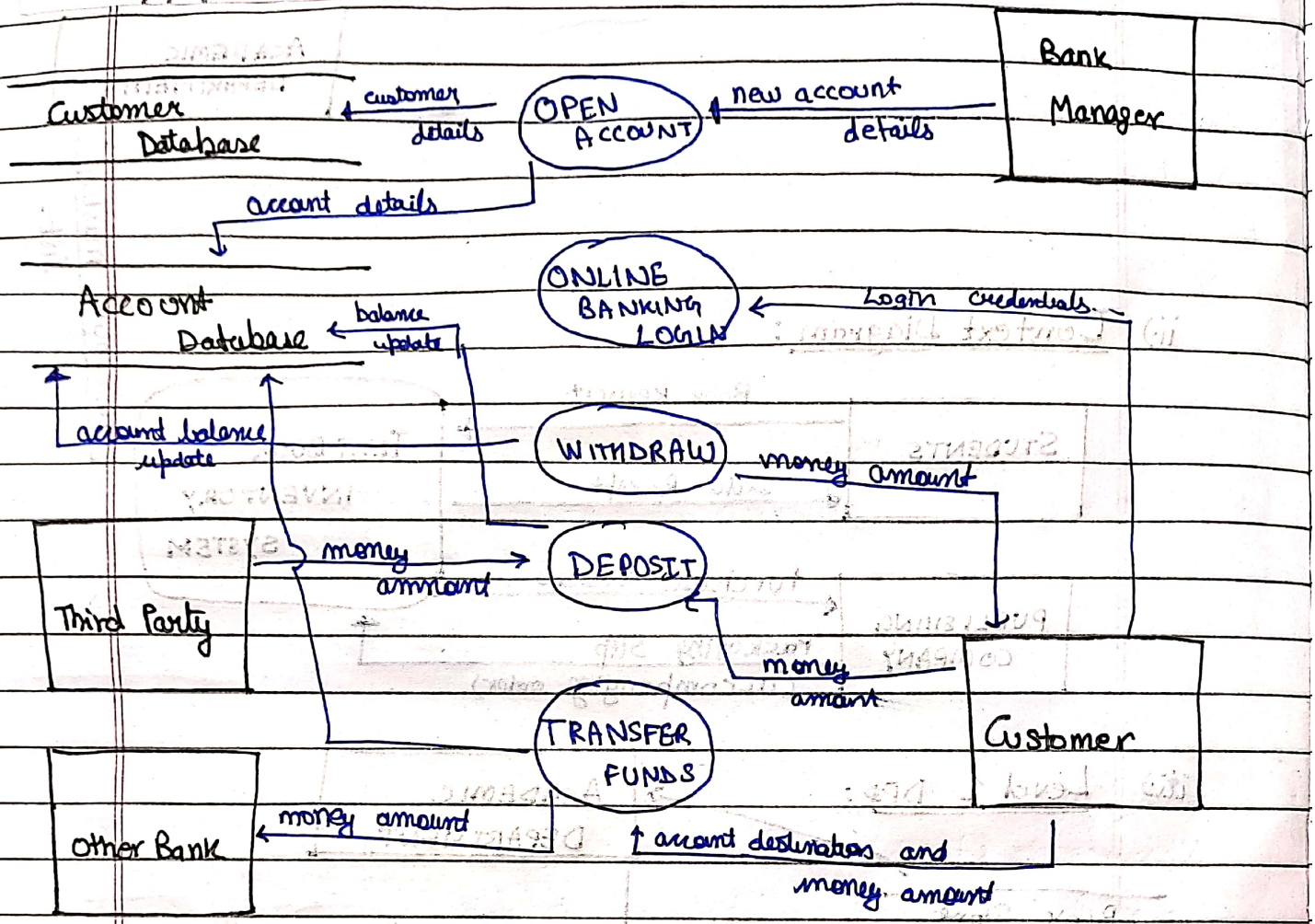
iii) Level 1 DFD :



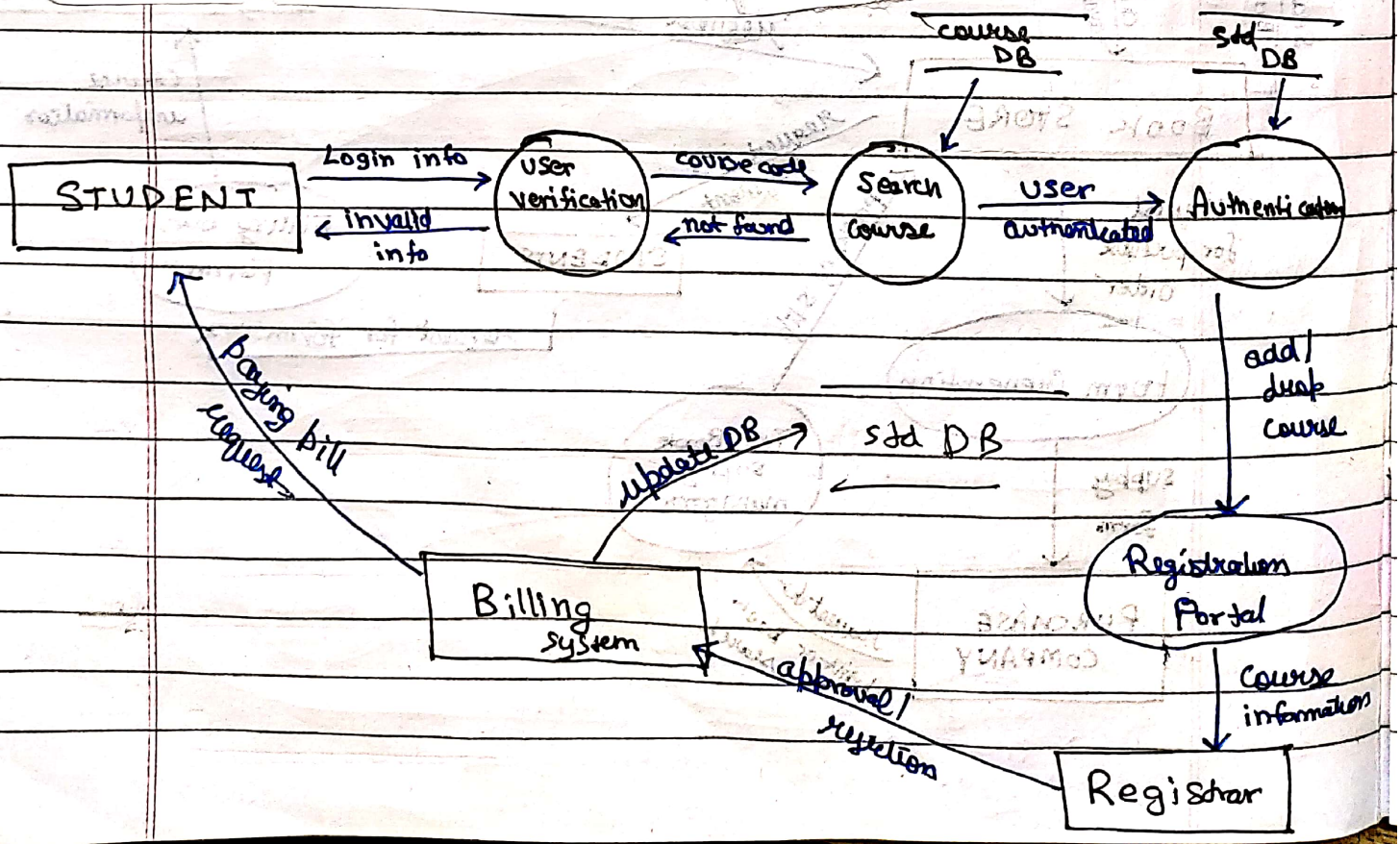
Date : / /
Page No.

[b] Draw a DFD for banking system to deposit an amount.

OR



Draw DFD for University Course Registration system to register for a course, paying the fees. 2010M 3



UNIT 4.1 - Software Project Management.

Date : / /
Page No.

b) At which point in the software development life cycle, does the project planning start? Why does a project manager need to estimate the size of the project? How is the size estimated? [4]

2018 S

Project planning typically begins in the SDLC after the project is found to be feasible. Once its found feasible, the project manager undertakes project planning to determine the necessary routines, time estimates, and other parameters for successful project execution.

Its done to correctly determine the effort duration, cost required to develop the product, and is helpful in making informed decisions regarding scheduling, staffing and resource allocation.

It helps in setting realistic goals, negotiating prices with customers and ensuring the successful completion of the project within the allocated resources.

Ways to estimate size :

→ TOP-DOWN ESTIMATION:

→ BOTTOM-UP ESTIMATION:

→ EXPERT JUDGEMENT:

→ COMPARATIVE ESTIMATION:

→ PARAMETRIC MODEL ESTIMATION:

[d] Project effort estimation techniques.

Date : / /
Page No.

[b] List important shortcomings of LOC for use as a software size metric. Does the function point metric overcome these? Explain your answer. (7) 2018E

Short comings :-

→ There is no industry standard for measuring LOC as its language dependant and a line of assembler ≠ a line of COBOL. There is ambiguity if comment should be counted and which comm. are.

→ Soure instructions may vary with coding language and design method

Function Point tells about the functions in product when dealing with customers, the manufacturer talks in terms of functions available and not in terms of components

b) Explain the concept of Function point. Why function points are becoming acceptable in the industry? 2018E E [4+4]

Function points are function oriented metrics, derived using an empirical relationship based on countable (direct) measure of software's information domain and assessment of software's complexity.

Function points are calculated by counting the number of functional user requirements of the software and assigning a complexity weighting to each requirement. It's types are:

→ External inputs → Logical internal files → External interface Files
→ External outputs → External Enquiries

They are becoming applicable in industry because:

→ Improved Estimation Accuracy : Cost and effort can be estimated with greater accuracy than line count.

→ Reduced Risk : Its estimates can be used to identify and mitigate risks early.

→ Improved communication : It can be used to communicate the size and complexity to the stakeholders in a clear and concise way about project scope, budget and schedule.

→ Better Decision-Making :

[b] Explain the concept of function points. Compute the function point value for a project with the following information domain characteristics.
Number user inputs = 30
No. of user outputs = 25
No of user enquiries = 10
No. of files = 06
No. of external interfaces =4
Assume that all complexity adjustment factors are average. 7

2010E

Date : / /
Page No.

for avg, weighting →
EI : x4
EO : x5
EQ : x4
ILF : x10
EIF : x7

Unadjusted Function Point (UFP) = $\sum_{i=1}^{5} \sum_{j=1}^{3} Z_{ij} w_{ij}$

⇒ UFP = 30x4 + 25x5 + 10x4 + 6x10 + 4x7
= 120 + 125 + 40 + 60 + 28
= 273

Complexity adjustment factor (CAF) : all CAF are 3 (average)
= (0.65 + 0.01 x ΣFi) = 0.65 + 0.01(3x14) = 1.07

∴ FP = UFP x CAF
= 273 x 1.07
FP = 292.11

(b) Compute the function point value for a project with the following information domain characteristics.
Number of user inputs = 30
Number of user outputs = 42
Number of user enquiries = 08
Number of files = 07
Number of external interfaces = 6
Assume that all complexity adjustment values are moderate. [4][CO4]

2022E
5

for avg weighting →
EI : x4
EO : x5
EQ : x4
ILF : x10
EIF : x7

Unadjusted Function Point (UFP) : $\sum_{i=1}^{5} \sum_{i=1}^{3} Z_{ij} w_{ij}$

⇒ UFP = 30x4 + 42x5 + 8x4 + 7x10 + 6x7
= 474

Complexity adjustment factor (CAF) : moderate → 2
= (0.65 + 0.01 x ΣFi) = 0.65 + 0.01 (2x14) = 0.93

∴ FP = UFP x 1.07
FP = 440.82

Date : / /
Page No.

[b] The COCOMO Model 2005M

COCOMO model (construction cost model) is the most widely used estimating technique. It is a regression-based model developed by Barry Boehm. It considers three classes of software:-

- Organic (application) | • Semidetached (utility) | • Embedded (system).

MODE	PROJECT SIZE	NATURE OF PROJECT	INNOVATION	DEADLINE	DEV. ENV.
ORGANIC	2-50 KLOC	Small size project, experienced developers in familiar environment Eg) Payroll, Inventory Project	Little	Not tight	Familiar and in house
Semi-Detached	50-300 KLOC	Medium size project and team average few-ext onsimilar projects Eg) Utility system like compiler, DBMS, editors.	Medium	Medium	Medium
Embedded	over 300 KLOC	Large project, RTS, Complex-interface, very-little prior exp S/w is strongly coupled with h/w Eg) ATMS, air traffic control	Significant	Tight	Complex H/w / customer interface reqd.

types

BASIC MODEL: A simple and static cost estimation model that calculates project efforts and duration based on size of S/w, measured in KLOC. It provides a rough order-of magnitude estimation

$DE = E = a_b (KLOC)^{b_b}$

$DD = D = c_b (E)^{d_b}$

Team size = $DE/DD = E/D$

E → efforts applied in person-months

D → dev. time in months

a_a, b_b, c_c, d_b → coefficients.

INTERMEDIATE COCOMO: An extension of basic cocomo model that refines effort and duration estimates by considering additional project factors such as product attributes, hardware constraints, and personnel/team attributes

$$DE = a\,(KLOC)^{b} \times EAF \text{ (Error adjustment factor)}$$

Date : / /
Page No.

DETAILED COCOMO : The most comprehensive and accurate COCOMO model that divides a software project into multiple components or modules, and accounts for the interactions between cost drivers and project phases.

It estimates effort and duration for each component using the intermediate COCOMO model, then sums them up.
It considers software reuse, hardware constraints, and personnel/team attributes in estimation.

[b] Explain with the help of an example, how an intermediate COCOMO provide more accurate estimate as compare to basic COCOMO. 2018E (7)

Consider a software project to develop a new word processing app whose size is estimated to be 100KLOC.

Using basic cocomo,

$$E = a \times (KLOC)^b \text{ PM}$$
$$= 2.4 \times 100^{1.05} = 354 \text{ PM}$$

Now to refine the estimate, intermediate cocomo is used, where cost drivers be 1.38, 1.01, 1.20, 0.9, 1.0, 1.01, 1.04, 1.04, 1.00.

$$\therefore \text{Effort Multiplier} = \frac{\quad}{\quad} = 1.162$$

$$\therefore \text{Now Effort, } E = 2.4 \times (100)^{1.05} \times 1.162 = 504 \text{ PM}$$

$$504 > 354 \text{ PM}$$

As we can see, intermediate estimates a higher effort than basic COCOMO, it is because it considers several cost drivers which affect software development.
Here, the intermediate COCOMO model requires more info about the project to be estimated and has software reliability, product complexity etc. are considered

Q.1 (a) Explain all levels of COCOMO model. Assume that the size of an organic software product has been estimated to be 32,000 lines of code. Determine the effort required to develop the software product and the nominal development time. 2.5 2022E [4][CO4]

Date : / /
Page No.

for organic Software project:

$a = 3.2$ KLOC = 32

$b = 1.05$

$c = 2.5$

$d = 0.38$

Effort

$\therefore E = a(KLOC)^b$

$= 3.2\ (32)^{1.05}$

$= 121$ Person Months

Nominal Development Time

$\therefore D = c(E)^d$

$= 2.5(121)^{0.38}$

$= 15$ months.

4 a) Suppose a project was estimated to be 400 KLOC. Calculate the effort and development time for each of the three model i.e., organic, semi-detached & embedded. 2020E

Project	a	b	c	d
Organic	3.2	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	2.8	1.20	2.5	0.32

a) KLOC = 400

Effort

organic $3.2(400)^{1.05} = 1727$ PM

Semiattached $3.0(400)^{1.12} = 2462$ PM

Embedded $2.8(400)^{1.20} = 3712$ PM

Dev Time

ORGANIC $2.5 \times (1727)^{0.38} = 42$ Months

SEMI-ATTACHED $2.5 \times (2462)^{0.35} = 38$ Months

EMBEDDED $2.5 \times (3712)^{0.32} = 34$ Months

b) Consider a project to develop a full screen editor. The major components identified are: (i) Screen edit (ii) Command Language Interpreter (iii) File Input & Output (iv) Cursor Movement (v) Screen Movement. The size of these are estimated to be 4k, 2k, 1k, 2k and 3k delivered source code lines. Use COCOMO to determine the overall cost and schedule estimates (values for four cost drivers are 1.15, 1.15, 0.86 and 1.07 with rest of them are being nominal). 2018E

Project	a	b	c	d
Organic	3.2	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	2.8	1.20	2.5	0.32

[4+4]

b) using cost drivers,

$EAF = 1.15 \times 1.15 \times 0.86 \times 1.07 = 1.2168$

total lines of code,

KLOC = 4 + 2 + 1 + 2 + 3 = 12

$\therefore$ initial effort estimate → (organic) → small size (2 KLOC)

$E = a(KLOC)^b = 3.2(12)^{1.05} = 52.91$ PM

dev time $D = c(E)^d = 2.5(52.91)^{0.38} = 11.29$ M

Date : / /
Page No.

using the following equations, phase wise cost and schedule estimates can be calculated

$E_P = JUP\ E$

$D_P = J_P D$

effort Distribution

∴ System Design = 0.16 × 52.91 = 8.465 PM

Detailed Design = 0.26 × 52.91 = 13.756 PM

Module code and test = 0.42 × 52.91 = 22.222 PM

Integration and test = 0.16 × 52.91 = 8.465 PM

Schedule estimates

Phase wise

System Design = 0.19 × 11.29 = 2.145M

Detailed Design = 0.24 × 11.29 = 2.709M

Module code and Test = 0.39 × 11.29 = 4.403M

Integration and Test = 0.18 × 11.29 = 2.032M

1 a) Discuss the program length, potential volume, effort & time, and language level as per Token Count metrics. M

b) Define Halstead software science metrics. 2018S

Token count metrics are a set of software metrics that are calculated on the basis of tokens in a program.

(Token refers to building blocks of program like operators, operands, keywords, ; etc.)

PROGRAM LENGTH: Its the no. of tokens

$N = N_1$ (# operators) + N_2 (# operands)

for machine language, each line consists of one operand and one operator

$N = 2 * LOC$

POTENTIAL VOLUME: It is the #bits required to encode the program

$V = N * \log_2 \eta$ → Vocabulary size.

Effort and Time:

$E = V/L = DV$ (D → difficulty)

$T = \frac{E}{f * s} = E/\beta$

f → programming prod. factor

S → Stound number

β ≈ 18

Date : / /
Page No.

LANGUAGE LEVEL: It measures the expressiveness of a programming language. A higher level means that it is more expressive and easier to use.

$$\lambda = L \times V^* = L^2 V$$

Q.3 Attempt any two:
(a) For a program with number of unique operators =20 and number of unique operands =40, compute the following: 2022E
(i) program length (ii) program volume (iii) program level
(iv) effort (v) language level [4][CO4]

$N_1 = 20 = \eta_1$
$N_2 = 40 = \eta_2$

i) program length, $N = N_1 + N_2$
$= 20 + 40$
$\boxed{N = 60}$

tokens, $\eta = \eta_1 + \eta_2$
$\Rightarrow \boxed{\eta = 60}$

ii) program volume,
$V = N \log_2(\eta)$
$= 60 \log_2(60)$
$= 354.41$ bits

iii) program level,
$L = V^*/V = \frac{2\eta_2}{(\eta_1 N_2)} = \frac{2}{\eta_1} = \frac{1}{10} = 0.1$

iv) Effort, $E = V/L$
$= 354.41 \times 10$
$= 3544.1$ PM

v) Language level, $\lambda = L \times V^* = L^2 V = 0.01 \times 354.41$
$= 3.54$

Date : / /
Page No.

b) Explain the Halstead theory of software science. Is it Significant in today's scenario of component based software development? 2020E [4+4]

Halstead theory of software science is a predictive model for estimating the size, effort and cost of SD. It is based on the following metrics:

→ PROGRAM LENGTH (N): total # operators and operand occurances in program.
→ VOCABULARY SIZE (n): total # distinct operators
→ PROGRAM VOLUME (V): $V = N \log_2(n)$. Theoritical min volume
→ Program Level (L): $L = n_1/n_2$ $n_1 \rightarrow$ operator | $n_2 \rightarrow$ operand
→ Program Difficulty (D): $D = (n_1) * (N_2 / n_2)$
→ Program Effort (E): $E = V * D$
→ Development Time (T): Time required to develop a program, calculated using the effort and the avg programmer's productivity.

It is significant in today's component based S/w development because it can be used to estimate size, cost, effort of developing and integrating components. This info can be used to make informed decisions about the design and dev of component based software systems

POTENTIAL VOLUME: minimum size program for an algo has volume V*

$$V^* = (2 + n_2^*) \log (2 + n_2^*)$$

$n_2^* \rightarrow$ # conceptually unique input/output params
$2 \rightarrow$ two unique operators for procedure call

Estimated Program Lvl	Difficulty
$\hat{L} = \frac{2 n_2}{(n_1 N_2)}$	$\hat{D} = \frac{1}{\hat{L}} = \frac{n_1 N_2}{2 n_2}$

Effort and Time

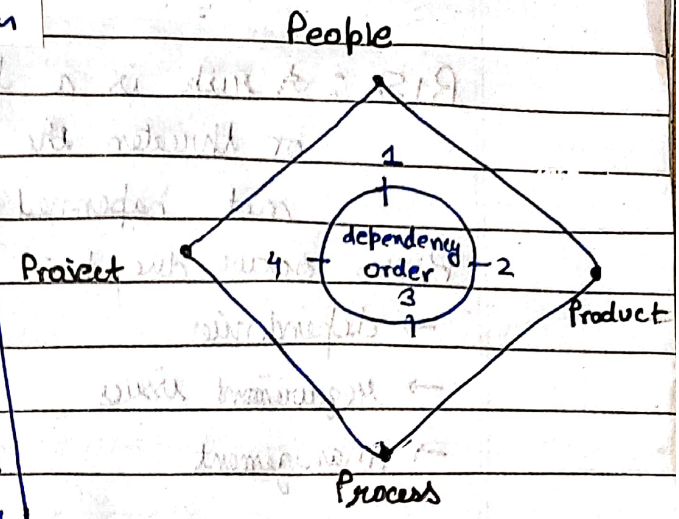
$E = V/L^{\alpha} = V * \hat{D}$
$= (n_1 N_2 N \log_2 n_1) / 2 n_2$

Language Lvl

$\lambda = L * V^* = L^2 V$

Date : / /
Page No.

Q1. Answer all questions.
(a) Explain the four P's of software management. 2018PM



The management of software development is heavily dependent on four factors:

1.) The PEOPLE

In s/w mgmt, effective leadership and understanding of team dynamics are crucial. Managerial selection is pivotal, as a skilled manager can significantly impact project success by managing, motivating and guiding the team.

2.) The PRODUCT

The focus is on delivering a clear sol^n to the customer's problems. Well-defined objectives and scope, along with consideration of alt sol^n are essential. Clear requirements are vital for estimating project costs, development time and schedule.

3.) The PROCESS

The software development process is the framework that influences the quality of the end product. Choosing an appropriate SDLC M such as CMM is critical. The process ensures a comprehensive plan and addresses the unique characteristics of each project.

4.) The PROJECT

Proper planning, monitoring and control are required for the success of project mgmt. Understanding potential challenges and freezing requirements early in the process helps mitigate risks and avoid costly surprises. The goal is to deliver s/w on time and within budget and meeting all requirements.

Date : / /
Page No.

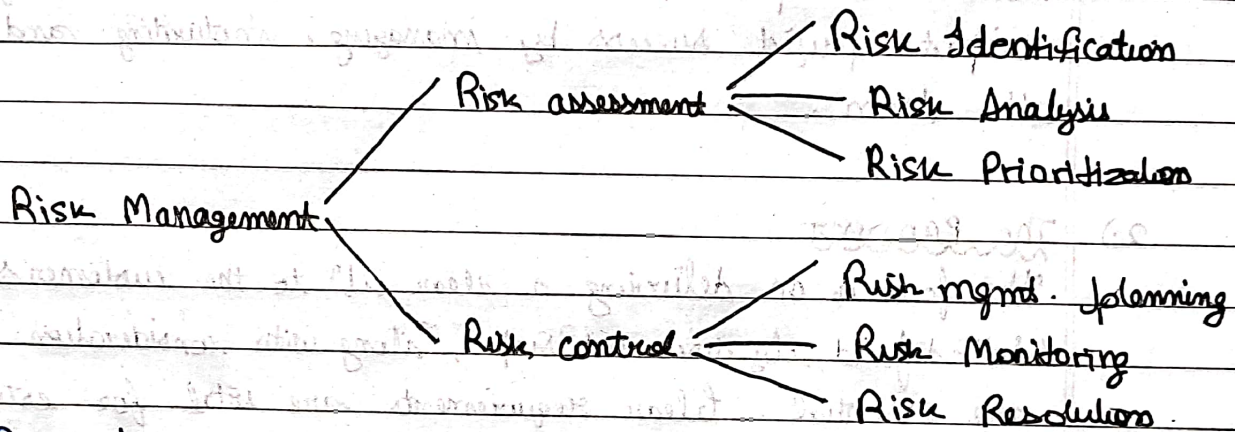
7 Write short notes on any TWO of the following
[a] Stages in risk management 2010E

6 [a] What do you understand by risk? Explain risk management activities in detail 2018E (7)

RISK : A risk is a problem that could cause some loss or threaten the success of the project, but which has not not happened yet.

Risks occur due to :
- → dependencies
- → requirement issues
- → management
- → lack of knowledge
- → other areas

RISK MANAGEMENT



→ **Risk assessment** : It involves identification of risks.
- **RISK ANALYSIS** : involves examining how project outcome might change with the modification of risk input variables.
- **RISK PRIORITIZATION** : focus on severe risks.
- **RISK EXPOSURE** : It is the product of the probability of incurring a loss due to the risk and the magnitude of that loss.

We can also practise risk avoidance like not taking certain projects or by relying on proven rather than cutting edge tech. . not doing risky things

→ **RISK CONTROL**

It is the process of managing risks to achieve desired outcomes. It starts with risk management planning, where a plan is created for each significant risk. Monitoring the project's progress involves periodic reevaluation of risks, their probabilities and likely impact. Risk resolution entails executing plans to address identified risks.

Date : / /
Page No.

UNIT 3.1 (System Design)

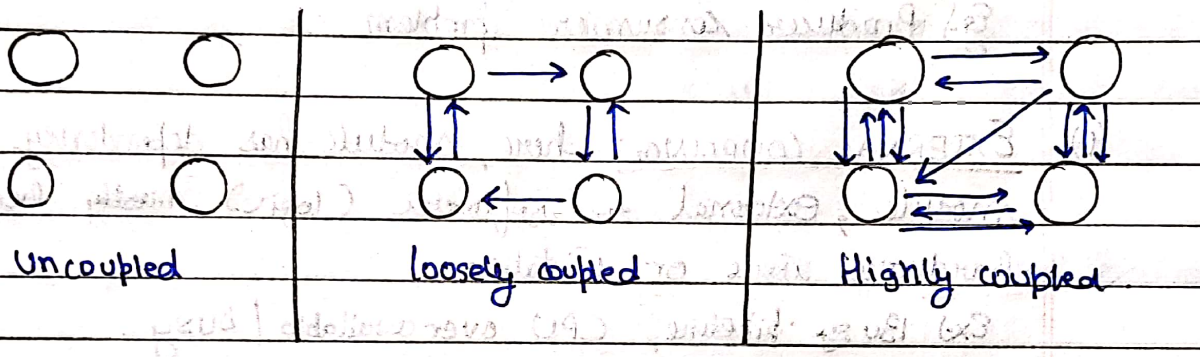
(b) What is coupling? Explain types of coupling. 2018S [4]

2 a) Discuss module coupling and its types with suitable example. 2018E

Coupling is the measure of degree of dependence between modules.

∴ Two modules with high coupling are strongly interconnected and thus dependant on each other.

Two modules with low coupling are not dependent and are made up of relatively independent modules → losely coupled.



A good design will have low coupling, thus interfaces should be carefully specified in order to keep low value of coupling.

∴ coupling is measured by # inter-connections between modules.

DATA coupling (BEST)
STAMP coupling
CONTROL coupling
EXTERNAL coupling
COMMON coupling
CONTENT coupling (WORST)

TYPES:

1) DATA COUPLING: here, modules communicate with each other only by passing data, other than this data comm, they are independent. (No tramp data)

Eg: addition by using "call by value".

Date : / /
Page No.

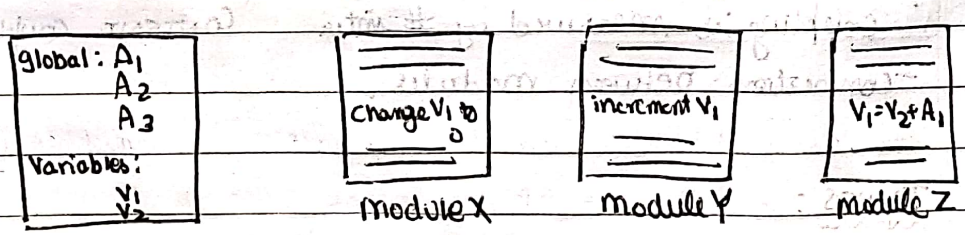
2.) STAMP COUPLING: here, two modules communicate with each other by passing complete data structures. Tramp data is involved as not all data in DS is reqd.
Eg.) addition by "call by reference"

3.) CONTROL COUPLING: here, modules are coupled if they communicate by passing of control information. This is usually accomplished by means of flags that are set by one module and reacted upon by the dependent module.
Eg.) Producer consumer problem

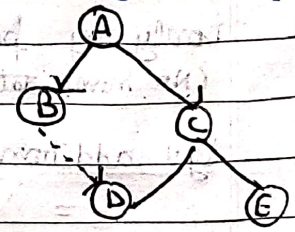
4.) EXTERNAL COUPLING: here, module has dependency to another module, external to software (logic) mostly because of hardware issue or limitation.
Ex.) Busy pipeline, CPU not available / busy.

5.) COMMON COUPLING: here, two modules have shared data. usually in global areas. Making a change to the common data means tracing back to all the modules with the access to that data to evaluate effect of change

Ex.)



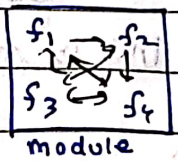
6.) CONTENT COUPLING: It occurs when module A changes data of module B or when control is passed from one module to the middle of another. i.e when one module is a part or content of another module.



Date : / /
Page No.

(Q) Define module cohesion and explain different types of cohesion. 2022E [4][CO3]

Cohesion is a measure of the degree to which the elements of a module are functionally related.
A strongly cohesive module implements functionality that is related to one features of the solution and requires little or no interaction with other modules.



FUNCTIONAL cohesion BEST (hight)
SEQUENTIAL cohesion
COMMUNICATIONAL cohesion
PROCEDURAL cohesion
TEMPORAL cohesion
LOGICAL cohesion
COINCIDENTAL cohesion WORST (low)

It is the measure of the mutual affinity of the components of a module.
Thus, we wants to maximise the interactions within a module. Hence, an important design objective is to maximize the module cohesion and minimize module coupling.

TYPES

1.) Coincidental cohesion: here, modules have no rel" other than shared code and relationship blw functiontion is random
Ex.) Check validity AND print is a single component with 2 parts.
It has to be avoided.

2.) Logical Cohesion: All the elements of a module perform similar or slightly similar functions.
Considerable duplication can exist in the logical strength level.

Ex.) mouse, printer, scanner input functions are written in same module

3.) Temporal Cohesion: here, functions are related by the fact that all tasks must be executed in the same time span. So,

Date : / /
Page No.

there is a flow control

Ex) The set of functions responsible for initialization, startup, activities such as setting program counters or control flags associated with programs exhibit temporal cohesion. This is not a good reason to put them in same procedure.

4.) Procedural Cohesion : here, functions of a module are related to each other through a flow control i.e. they are part of an algorithm or procedure.
It occurs in modules whose instructions although accomplish different tasks yet have been combined because there is a specific order in which the tasks are to be completed
Ex.) Calling one function to another function. Loop statement, reading records etc.

5.) Communicational cohesion: Both functions operate on same input data or contribute towards the same output data. This is okay, but we might consider making them separate procedures.
Ex.) update record in DB and send it to the printer.

6.) Sequential cohesion : here output data by function X is forming input for function Y. This is the reason for them to be contained in a single procedure.
Ex.) addition of marks of individual subjects into a specific format is used to calculate the GPA as input for preparing the result of the students.

7.) Functional Cohesion : here, functions X, Y etc are a part of a single functional task and are thus in same procedure. Such a module often transformed a single input datum to a single output datum.
Ex.) Railway reservation system.

Q2. What do you understand by coupling and cohesion? Explain their types. What roles they play in software design? 2018M E [5]

Date : / /
Page No.

[c] What do you mean by cohesion and coupling? How are they related to quality of software design? 2010E 4

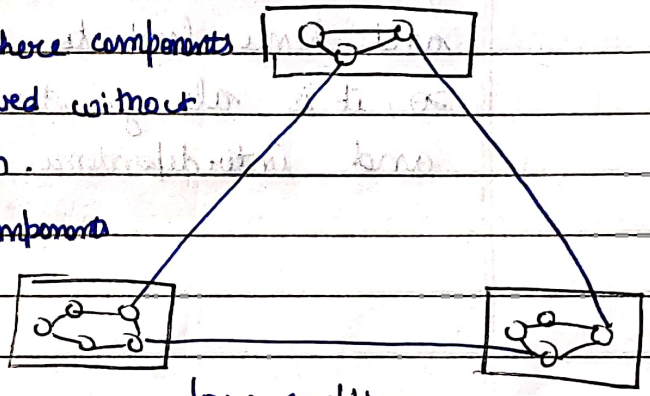
Design process involves breaking a system into parts for better understanding. Projects don't fail often due to significant requirement changes, but rather when non-modular software can't handle minor changes.

Good software design divides problem into modules and organises them neatly.

Software engineers should aim for high cohesion and low coupling in module design.

Ex.) "plug and play" feature where components can be added or removed without affecting the entire system.

This is because the add on components provide services in highly cohesive manner.



∴ High cohesion and low coupling makes modules function as black boxes, allowing separate handling of each module when describing their functionality.

4 [a] What do you mean by cohesion and coupling in a design? Is it true that a good design should have high coupling and low cohesion? Explain your answer. 2018E (7)

No, a good design should have high cohesion and low coupling instead.

undesirable
- High coupling makes a system inflexible and hard to maintain
- Low cohesion results in poorly organized and less clear code.

Date : / /
Page No.

(b) What problems are likely to arise if two modules have high coupling? [4][CO3] 2022E

A system with high coupling means there are strong interconnections between its modules. If two modules are involved in high coupling, it means their interdependence will be very high. Any changes applied to one module will affect the functionality of the other module. Greater the degree of change, greater will be its effect on the other. As the dependence is higher, such change will affect modules in a negative manner and in turn, the maintainability of the project is reduced. This will further reduce the reusability factor of individual modules and hence lead to unsophisticated software.
So, it is always desirable to have inter-connection and interdependence among modules.

4. Can a system ever be "completely decoupled"? Can the degree of coupling be reduced so much that there is no coupling between modules? Also, explain the term cohesion in software. 2019M E [3+2=5]

b) Can a system ever be completely "decoupled"? That is, can the degree of coupling be reduced so much that there is no coupling between modules? Explain your answer. E 2022M [5] [CO1,CO4]

A system can never be decoupled completely. This is because a system is made of various modules that work together to achieve the final objective of the system. Coupling represents the dependency of modules on each other which is necessary for the efficient functioning of the system. Modules need to use services from one another to produce results.

Coupling can be reduced to some extent which is a good thing to achieve. But complete decoupling isn't possible as if the modules are decoupled, the system will not be able to do anything.

Date : / /
Page No.

b) Compare top-down and bottom-up software design approach with advantages and disadvantages of both approaches. 2020E

[4+4]

TOP-DOWN APPROACH

In top-down design, we start with a high-level view of the system and then decompose it into smaller modules and sub-modules.

ADVANTAGES

→ It provides a clear and abstract view of the system, making it easier to understand and manage the design.
→ We can start coding and testing higher level modules even before lower-level modules are fully designed which speeds up development.
→ Suitable for new systems being developed from scratch and with well defined specification.

DISADVANTAGES

→ It heavily relies on having clear and accurate specifications. If the initial specifications are incorrect or incomplete, the entire design may need to be revised.
→ Testing cannot occur until all subordinate modules are implemented.

BOTTOM-UP APPROACH

In bottom-up design, we start by identifying and designing smaller modules that are collected into libraries, and then combine these modules to create larger ones until the final program is assembled.

ADVANTAGES

→ individual modules can be coded and tested as soon as they are designed, allowing for early validation of the design
→ effective when building upon existing modules or libraries, as it leverages pre-existing components.

Date : / /
Page No.

DISADVANTAGES

→ deciding the exact functionality of modules may rely heavily on intuition which can lead to incorrect designs.

→ if module functionalities are not well defined initially, there's a risk of having to recode them, which can be time consuming.

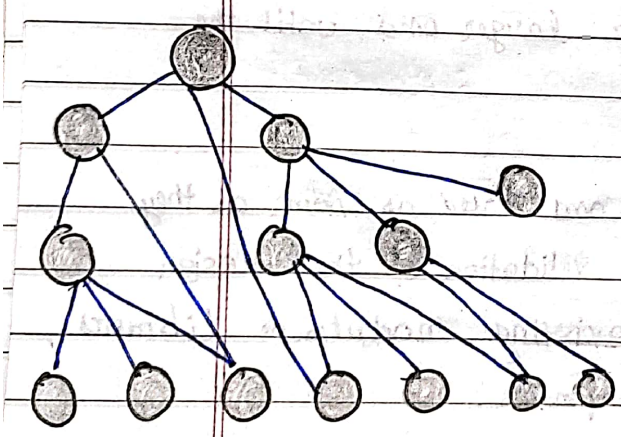
Q4. a) Discuss the difference between object oriented and function oriented design. 2022M E

HYBRID APPROACH

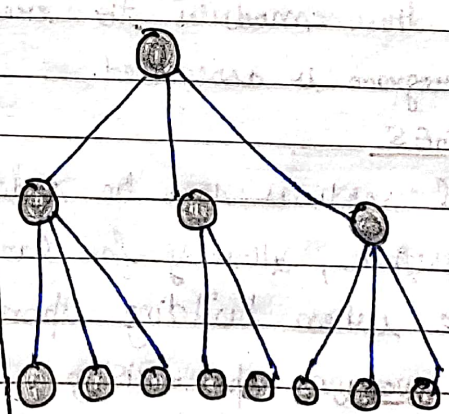
The hybrid strategy of design combines elements of both top-down and bottom up approaches. A bottom up approach requires a clear understanding of the top layers for success, while a top down approach benefits from some bottom-up input, especially at lower design levels.

The hybrid approach allows for common sub-modules, simplifies intuition at lower levels, and supports the reuse of pre-written library modules.

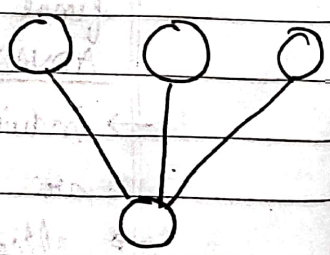
It has become really popular after the acceptance of reusability of modules. Std libraries, microsoft foundation classes (MFCs), object oriented concepts are the steps towards this.



Bottom-up Tree Structure



Top-Down Tree Structure



Design Reusable Structure

Date : / /
Page No.

FUNCTION ORIENTED DESIGN	OBJECT ORIENTED DESIGN
The basic abstractions, which are given to the users, are real world functions	The basic abstractions are not real world functions but are data abstractions where real entities are represented.
Functions are grouped together by which a higher level function is obtained	Functions are grouped together on the basis of the data they operate since the classes are associated with their methods
Its carried out using structured analysis and structured design i.e. DFD.	Its carried out using UML
In this approach the state information is often represented in centralized shared memory.	In this approach the state information is implemented or distributed among the objects of the system.
It is a top down approach which we can decompose into func/procedures	It is a bottom up approach which we can decompose in class level.
It begins by considering the use case diagrams and the scenarios.	It begins by identifying objects and classes.
This approach is mainly used for computation sensitive application	This approach is mainly used for evolving systems which mimics a business or business case.

Date : / /
Page No.

UNIT 4.2 – Reliability and Quality Assurance

Q5.a) What do you mean by software quality? Explain its attributes. 2018S [4]

Software quality can be understood as the degree to which a software product meets its specified requirements, functions without defects (lack of bugs), and satisfies the needs and expectation of its users.
The lack of bugs can be measured in terms of:
i.) defect rate
ii.) reliability.

(b) Software quality attributes 2018E

Software Quality attributes help to measure the quality from a multi-faceted angle:

a.) **Reliability:**
The extent to which a software performs its intended functions without failure.
i.) Correctness: The extent to which a s/w meets its specifications.
ii.) Consistency and precision: The extent to which a s/w is consistent and gives result with precision.
iii.) Robustness: The extent to which a s/w tolerates unexpected problems.
iv.) Tracability: The extent to which an error is tracable in order to fix it.
v.) Simplicity: The extent to which a s/w is simple in its operation.

b.) **Usability:**
The extent of effort required to learn, operate and understand the functions of a software.
i.) Accuracy: Meeting specifications with precision.
ii.) Clarity and accuracy of documentation: The extent to which documents are clearly and accurately written.
iii.) Conformity of operational environment: The extent to which a s/w is in conformity of operational environment.

Date : / /
Page No.

iv.) Completeness : The extent to which a s/w has specified functions.

v.) Efficiency : The amount of computing resources and code required by software to perform a function.

vi.) Testability : The efforts required to test a software to ensure that it performs its intended functions.

c.) Maintainability :
The ability to locate and fix an error during maintenance phase.

i.) Modularity : It is the extent of ease to implement, test, debug and maintain the software.

ii.) Readibility : The extent to which a software is readible in order to understand.

iii.) Accuracy and Clarity of documentation.

iv.) Simplicity

d.) Adaptability :
The extent to which a software is adaptable to new technologies and platforms.

i.) Modifiability : The effort required to modify a software during maintenance phase.

ii.) Expandability : The extent to which a software is expandable without undesired side effects

iii.) Portability : The effort required to transfer a program from one platform to another platform.

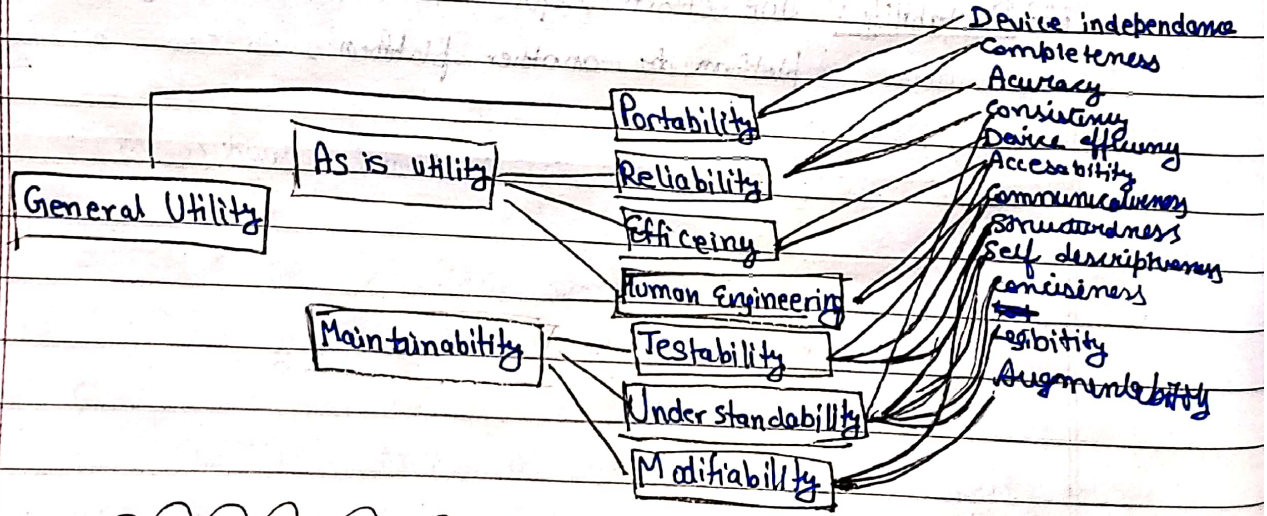
Date : / /
Page No.

c) Boehm Software Quality Model (1978) 2018E END

In this model → Boehm addresses the contemporary shortcomings of models that automatically and qualitatively evaluate the quality of software. He attempts to qualitatively software quality as a measure of given set of attributes and metrics.

Boehm's model asserts that a quality software satisfies the needs of the users. It reflects an understanding of quality where the S/W:
→ does what the user wants it to do.
→ uses resources correctly and efficiently.
→ is easy for the user to learn and use
→ is well defined, well coded, easily tested and maintained.

It is a heirarchal model with three lvl of characteristics.
→ highest lvl : It represents basic high lvl req. of actual use to which evaluation of soft. quality could be put - the general utility of S/W. It addresses-
→ As-is utility : How well can i use it as-is.
→ Maintainability
→ Portability
→ Intermediate lvl : represents 7 quality factors.
→ Lowest lvl : primitive characteristics metrics hierarchy.



The Boehm SQM

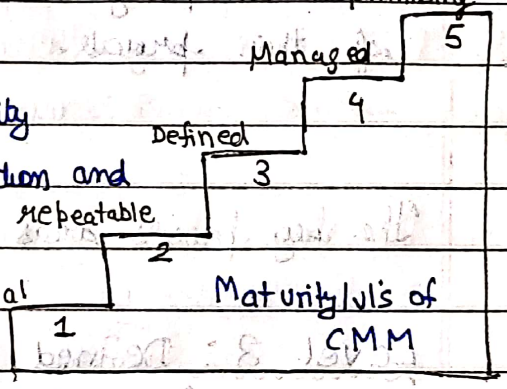
Date : / /
Page No.

Q7. (a) Explain the CMM model and its levels. 2018 S (c) SEI Capability Maturity Model 2018 E
-END-

CMM is a strategy for improving the software process irrespective of the actual life cycle model used, to generate quality software. Its developed by SEI of Carnegie-Mellon Uni (1986).

The term "maturity" relates to the degree of formality and optimization of processes, from ad hoc practices, to formally defined steps, to manage resuet metrics, to active optimization of the process.

CMM is used to judge the maturity of the software process of an organization and to identify the key practices that are required to increase the maturity of these processes.



There are five levels of CMM:

1. INITIAL (process unpredictable and poorly controlled)
   → no engineering management, everything done on ad hoc basis
   → software process is unpredictable wrt time and cost
   → It depends on current staff, as staff changes so does the process

2. REPEATABLE (basic project management)
   → planning and managemning of new projects based on experience with similar projects.
   → realistic plans based on the performance based on the previous projects

3. DEFINED (process standardization)
   → process of developing and maintaining S/w across the organization is documented including engineering and management.
   → training programs are implemented to ensure that the staff has skills and knowledge required
   → Risk Management.

Date : / /
Page No.

4. MANAGED (Quantitative measurement)
→ organization set qualitative goals for both product and process.
→ here process is predictable both wrt time and cost.

5. OPTIMIZED (Continuous process improvement)
→ here organization analyis defects to determine their causes and goals is to preventing the occurance of defects.
→ here company continuously improve the process performance of their projects.

[b] Define CMM. Discuss KPAs at level 3 and level 4 of CMM. 20106

The key process areas (KPAs) of lvl 3 and lvl 4 of CMM are:

Level 3 : Defined

→ Organization Process Focus (OPF) : The org. establishes and maintains an infrastructure to improve its s/w processes.

→ Organization Process Definition (OPD) : The org. defines and documents its software processes.

→ Training Program (TP) : The organization provides training to its employees on the defined s/w processes.

→ Integrated Software Management (ISM) : The org integrates its software processes into a coherent whole.

→ Software Product Engineering (SPE) : The org. uses sound engineering principles to develop software products.

→ Intergroup Coordination (IC) : The organization coordinates the activities of its different groups involved in s/w development.

Date : / /
Page No.

→ Peer Reviews (PR): The organization uses peer reviews to improve the quality of its software processes and products.

LEVEL 4 : Quantitatively Managed

→ Qualitative Process Management (QPM): The org. measures and control its s/w development process using Stats and qual. techniq.

→ Software Quality Assurance (SQA): The org. uses quantitative methods to measure and improve the quality of its s/w product.

→ Process and Product Quality Assurance (PPQA): The org. integrates its SQA and process improvement activities.

b) Discuss the ISO 9000 certification for software industry and also explain what is SEI capability maturity model. 2020E [4+4]

ISO 9000 is a series of standards released in 1987 by International Standards Organization which determines the guidelines for maintaining a quality system.

The ISO 9000 series is based on the assumption that if a proper stage is followed for production, then good quality product are bound to follow automatically.

ISO 9001 : This standard applies to the organizations enagged in design, development, production, servicing of goods.

ISO 9002 : _____ Donot design products, only involved in the production.

ISO 9003 : _____ involved in the installation and testing of products.

Date : / /
Page No.

5 [a] Write salient requirements for obtaining ISO 9000 model. 2018E (7)

An organisation determines to obtain ISO 9000 certification applies to ISO registrar offic for registration which consists of the following stages :

1. APPLICATION: Once an organisation decided to go for ISO certification, it applies to the registrar for registration

2. PRE-ASSESSMENT : During this stage, the registrar makes a rough assessment of the organisation.

3. DOCUMENT REVIEW AND ADEQUACY OF AUDIT : During this stage, the registrar reviews the document submitted by the organisation and suggest an improvement.

4. COMPLIANCE AUDIT : Here, the registrar checks whether the organisation has complied the suggestions made by it during the review or not.

5. REGISTRATION : The registrar awards ISO certification after the successful completion of all phases.

Q.2 Attempt any two:

(a) Discuss the relative merits of ISO-9001 certification and the SEI CMM based evaluation. Point out some of the shortcoming of the ISO-9001 certification process as applied to the software industry.

2022E [4][CO4]

Date : / /

Page No.

The biggest difference b/w ISO-9001 and CMM is the emphasis of the CMM on continuous process improvement. ISO-9001 addresses the minimum criteria for an acceptable quality system. It may be due to the fact that CMM strictly focusses on software while ISO-9001 has a much broader scope: hardware/software, processed materials, and services.

Similarly is that bottom line for them is "Say what you do and do what you say". They focus on documentation and quality check of deliverables.

CMM emphasizes the need to record information for later use in the process and for improvement of the process. which is equivalent to the quality records of ISO-9001 that document if required quality is achieved or not.

ISO 9001 is a standard for quality system within ISO 9000 series and is most applicable for software development.

Because of the broadness of ISO-9001, ISO has published specific guidelines to assist in applying ISO9001 to software namely ISO90003

There is significant room for interpretation in using ISO 9001 in the software world. ISO 9000-3 is a guide to interpret ISO-9001, yet many-to-many relationships b/w their clauses raise the suspicion that liberties have been taken in creating this guideline.

Date : / /
Page No.

b) Discuss the difference between CMM and ISO 9001. Why is it suggested that CMM is the better choice than ISO 9001? 2018 [4+4]

Contrast:

→ CMM focusses strictly on software, while ISO 9001 covers a broader scope, including hardware, software, processed materials, and services.

→ The CMM places a strong emphasis on continious process improvement, while ISO 9001 addresses the minimum criteria for an acceptable quality system.

→ CMM emphasizes documentation and practicing processes, including recording information for later use and process improvement.

→ ISO 9001 requires documentation with instructions and quality control, similar to the CMMs focus on documented procedures and written organizational policies.

∴ ISO → "Say what you do; do what you say"
CMM → "according to a documented procedure" / "a written org. policy".

CMM better than ISO 9001?
It is better since CMM is specifically designed for software dev and provides a more comprehensive and structured approach to process improvement (ISO is generic)
Thus, CMM provides a focused and prescriptive approach to improving SD process.

Date : / /
Page No.

b) Define a) Test case b) Test suite 2018S

a) TEST CASE:

A test case (test) is a detailed description comprising preconditions (that are the circumstances existing before its execution) and actual inputs identified through testing methods. It includes expected outputs, categorized into post conditions and actual outputs

Test case ID:	
SECTION - 1 (Before Execution)	SECTION - 2 (After Execution)
Purpose:	Execution History:
Pre-condition (if):	Result:
Inputs:	Failure reason (if):
Expected outputs:	any other observation:
Post conditions:	any suggestion:
Written by:	Run by:
Date:	Date:

Test cases are vital for testing. They set conditions, compare output, and find errors.

They are as important as source code, needing development and preservation.

b) TEST SUITE:

The set of test cases is called a test suite. We may have a test suite of all possible test cases. We may have a test suite of effective / good test cases.
Hence, any combination of test cases may generate a test suite.

Date : / /
Page No.

[b] Differentiate between
(i) Alpha and Beta testing 2010 E
c) Alpha and Beta testing 2018 S

The terms alpha and beta testing are used when the software is developed for anonymous customers. Hence formal acceptance testing is not possible in such cases.

	ALPHA TESTING	BETA TESTING
Location	Developer's site	Customer's or end user's site
Environment	Controlled environment	Real, uncontrolled environment
Developer?	Developer is present	Developer is absent
Reporting failures	failures reported to developer or company typically	failures reported by customers to company
Purpose	Its done at the end of formal testing to get views of potential customers	Sent Product to potential customer for four month to fix bugs for final release.
Reputation	Reputation of company may be at stake	Company's reputation is not affected significantly

Date : / /

Page No.

Q6.a) Design the equivalence class test suite for a function named for largest amongst three numbers taking three integer numbers (a, b, c) as input in range [1-200]. 2018S [4]

output domain equivalence classes are:

$O_1 = \{\langle a,b,c\rangle : a \text{ is largest}\}$

$O_2 = \{\langle a,b,c\rangle : b \text{ is largest}\}$

$O_3 = \{\langle a,b,c\rangle : c \text{ is largest}\}$

$O_4 = \{\langle a,b,c\rangle : \text{invalid i/p}\}$

first we partition the domain of input as valid input values and invalid values, so we get the following classes.

$I_1 = \{\langle a,b,c\rangle : 1 \le A \le 200\}$

$I_2 = \{\langle a,b,c\rangle : 1 \le b \le 200\}$

$I_3 = \{\langle a,b,c\rangle : 1 \le c \le 200\}$

$I_4 = \{\langle a,b,c\rangle : a < 1\}$

$I_5 = \{\langle a,b,c\rangle : a > 200\}$

$I_6 = \{\langle a,b,c\rangle : b < 1\}$

$I_7 = \{\langle a,b,c\rangle : b > 200\}$

$I_8 = \{\langle a,b,c\rangle : c < 1\}$

$I_9 = \{\langle a,b,c\rangle : c > 200\}$

∴ Test cases are :

TC ID	a	b	c	Expected Result	Classes Covered
1	26	50	72	C is larger	I_1, I_2, I_3
2	0	91	55	Invalid input	I_4
3	203	63	101	Invalid input	I_5
4	177	0	99	Invalid input	I_6
5	123	201	81	Invalid input	I_7
6	7	9	0	Invalid input	I_8
7	98	115	208	Invalid input	I_9

another set of eq classes can be

$I_1 = \{\langle a,b,c\rangle : a > b, a > c\}$

$I_2 = \{\langle a,b,c\rangle : b > a, b > c\}$

$I_3 = \{\langle a,b,c\rangle : c > a, c > b\}$

$I_4 = \{\langle a,b,c\rangle : a = b, a + c\}$

$I_5 = \{\langle a,b,c\rangle : b = c, a + b\}$

$I_6 = \{\langle a,b,c\rangle : a = c, c = b\}$

$I_7 = \{\langle a,b,c\rangle : a = b = c\}$

Date : / /
Page No.

∴ test cases are.

Testcase ID	a	b	c	Expected Resut	Eq Classes Covered
1	50	26	26	a is larger	I_1, I_5
2	25	40	25	b is larger	I_2, I_6
3	72	72	101	C is larger	I_3, I_4
4	50	50	50	All are equal	I_7

2 a) Consider a program for the determination of the nature of roots of a quadratic equation. Its input is a triple of positive integer (say a, b, c) and values may be from interval [0,50]. The program output may have one of the following words: 2020E
[Not a quadratic equation; Real roots; Imaginary roots; equal roots]
Design the equivalence class test cases .

output domain equivalence class test can be identified as follows:

$O_1 = \{ \langle a, b, c \rangle$: Not a quadratic equation if $a = 0 \}$

$O_2 = \{ \langle a, b, c \rangle$: Real roots if $(b^2 - 4ac) > a \}$

$O_3 = \{ \langle a, b, c \rangle$: Imaginary roots if $(b^2 - 4ac) < 0 \}$

$O_4 = \{ \langle a, b, c \rangle$: Equal roots if $(b^2 - 4ac) = 0 \}$

The no of test cases can be derived from above such and shown below:

Test Case	a	b	c	Expected output	Eq Class
1	0	50	50	Not a quadratic Eqn	O_1
2	1	50	50	Real roots	O_2
3	50	50	50	Imaginary roots	O_3
4	50	100	50	Equal roots	O_4

We may have another set of test cases based on input domain:

$I_1 = \{ a : a = 0 \}$

$I_2 = \{ a : a < 0 \}$

$I_3 = \{ a : 1 \leq a \leq 50 \}$

$I_4 = \{ a : a > 50 \}$

$I_5 = \{ b : 0 \leq b \leq 50 \}$

$I_6 = \{ b : b < 0 \}$

$I_7 = \{ b : b > 50 \}$

$I_8 = \{ c : 0 \leq c \leq 50 \}$

$I_9 = \{ c : c < 0 \}$

$I_{10} = \{ c : c > 50 \}$

In these classes, our basic assumption is single faulty theory. Hence one value is at an extreme and other values are nominal values.

Input domain test cases are:

P.T.O.

Date : / /
Page No.

test case ID	a	b	c	Expected output	Eq class covered
1	0	50	50	Not a quadratic Eqn	I_1, I_5, I_8
2	-1	50	50	Invalid input	I_2, I_5, I_8
3	50	50	50	Imaginary roots	I_3, I_5, I_8
4	101	50	50	Invalid input	I_4, I_5, I_8
5	50	-1	50	Invalid input	I_3, I_6, I_8
6	50	106	50	Invalid input	I_3, I_7, I_8
7	50	50	-1	Invalid input	I_3, I_5, I_9
8	50	50	101	Invalid input	I_3, I_5, I_{10}

Q.4 Attempt any two:

(a) Let us consider an example of grading the students in an academic institution. The grading is done according to the following rules:

Marks Obtained	Grade
80-100	Distinction
60-79	First Division
50-59	Second Division
40-49	Third Division
0-39	Fail

2022E

Generate test cases using equivalence class testing technique. [4][CO5]

marks obtained = m

$O_1 = \{ \langle m \rangle ;$ distinction $\}$
$O_2 = \{ \langle m \rangle ;$ First division $\}$
$O_3 = \{ \langle m \rangle ;$ second division $\}$
$O_4 = \{ \langle m \rangle :$ third division $\}$
$O_5 = \{ \langle m \rangle :$ fail $\}$
$O_6 = \{ \langle m \rangle ;$ Invalid marks $\}$

$I_1 = \{ m : m > 100 \}$
$I_2 = \{ m : 80 \leq m \leq 100 \}$
$I_3 = \{ m : 60 \leq m \leq 79 \}$
$I_4 = \{ m : 50 \leq m \leq 59 \}$
$I_5 = \{ m : 40 \leq m \leq 49 \}$
$I_6 = \{ m : 0 \leq 39 \leq 39 \}$
$I_7 = \{ m : m < 0 \}$

test cases generated.

T.C. id	m	Expected output	Eq Class covered
1	171	invalid input (O_6)	I_1
2	93	distinction (O_1)	I_2
3	66	first division (O_2)	I_3
4	54	second division (O_3)	I_4
5	49	third division (O_4)	I_5
6	7	fail (O_5)	I_6
7	-2	invalid input (O_6)	I_7

Date : / /
Page No.

5 a) Consider a program for determining the previous date. Its input is a triple of day, month and year with the values in the range
$1 \le$ month ≤ 12
$1 \le$ day ≤ 31
$1900 \le$ year ≤ 2025
The possible outputs would be previous date or invalid input date. Design the Equivalence class test cases.

2018E

output domain equivalence classes are:
$O_1 = \{ \langle D, M, Y \rangle$: previous date if all are valid inputs $\}$
$O_2 = \{ \langle D, M, Y \rangle$: invalid date if any inputs makes the date invalid $\}$

test case ID	M	D	Y	Expected output
1	6	15	1962	14 June, 1962
2	6	31	1962	Invalid date

we may have another set of test cases which are based on input domain

$I_1 = \{$month : $1 \le m \le 12\}$
$I_2 = \{$month : $m < 1\}$
$I_3 = \{$month : $m > 12\}$
$I_4 = \{$day : $1 \le D \le 31\}$
$I_5 = \{$day : $D < 1\}$
$I_6 = \{$day : $D > 31\}$
$I_7 = \{$year : $1900 \le Y \le 2025\}$
$I_8 = \{$year : $Y < 1900\}$
$I_9 = \{$year : $Y > 2025\}$

Input domain test cases are:

Testcase	M	D	Y	Expected output	Equivalence Class covered
1	6	15	1962	14 June, 1962	I_1, I_4, I_7
2	-1	15	1962	Invalid input	I_2, I_4, I_7
3	13	15	1962	Invalid input	I_3, I_4, I_7
4	6	15	1962	14 June, 1962	I_1, I_4, I_7
5	6	-1	1962	Invalid input	I_1, I_5, I_7
6	6	32	1962	Invalid input	I_1, I_6, I_7
7	6	15	1962	14 June, 1962	I_1, I_4, I_7
8	6	15	1899	Invalid input (value out of range)	I_8, I_4, I_8
9	6	15	2026	———	I_1, I_4, I_9

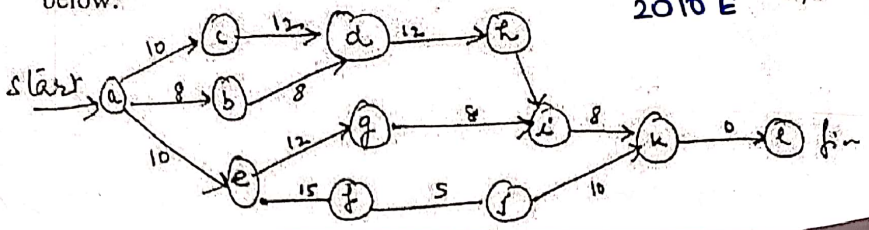
Date : / /
Page No.

(c) Explain the significance of independent paths. Is it necessary to look for a tool for flow graph generation, if program size increases beyond 100 source lines? [4][CO5] 2022 E

Independant paths are significant because they help to ensure that all possible program states are covered by test cases, This is important because it helps to ensure that the program is working as intended and that it won't produce unexpected behaviour.

If the program size ↑ beyond 100 LOC, it may be necessary to use a tool for flow graph generator because it will be difficult to find all possible paths manually which may also result in mistakes.
A (FGG) can help automate this process - ~~easy~~ easy.

4[a] Identify the critical activities and critical path for the problem given below: 2010 E 7



Date : / /
Page No.

b) Cyclomatic Complexity 2018E

Cyclomatic complexity (also called structural complexity) is a software metric used to measure the complexity of a program.
It is a quantitative measure of the number of linearly independent paths through a program's source code.
This provides us with an upper bound for the number of tests that must be conducted to ensure that all the statements have been executed atleast once and all conditions are executed on its true and false side.

It was developed by Thomas J. Mcabe in 1976 and can be computed with the help of a Control Flow Graph
∴ Given a CFG, G of a program, the cyclomatic complexity V(G) can be computed as:

$$V(G) = E - N + 2$$

E → # edges | N → # nodes.

6 a) Draw the control flow graph of the following code and also find the cyclomatic complexity. 2020E

```
1. Int A
2. IF (A%2=0) THEN
3. PRINT("No is Even )
4. ELSE
5. PRINT("No is Odd")
6. ENDIF
```

Clearly, cyclomatic complexity,

$$V(G) = E - N + 2$$

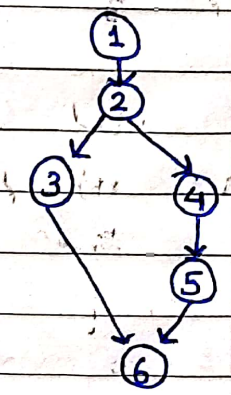
Here,

$E = 6$

$V = 6$

$$\therefore V(G) = 6 - 6 + 2 = 2$$

It can be represented as



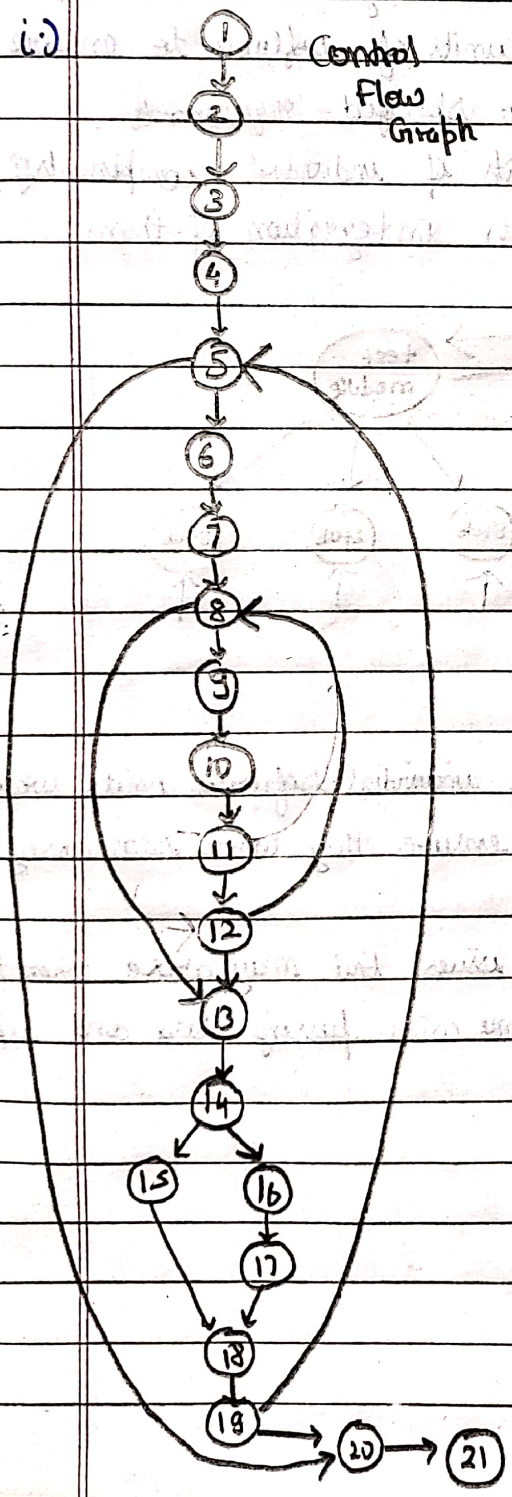
Date : / /
Page No.

3[a] Consider the following C program : 2010E

```
1  main ()
2  { int num-students, marks, subjects, total
3        float average;
4        num-student =1;
5        while (num-student <=25)
6  { total = 0;
7        subject :=1
8        while (subject <=5)
9        {scanf("%d",&marks);
10 total =total + marks;
11 subject ++;
12 }
13 average = total / 5;
14 if (average < 50)
15       printf ("Fail")
16 else
17       printf("\n pass..Average marks are %f\n", average);
18 num-student++;
19       }
20    printf("End of program")
21 }
```

(i) Draw flow graph for above code.
(ii) Compute cyclomatic complexity.
(iii) Identify all independent paths. 10

(i) Control Flow Graph



(ii) Cyclomatic complexity

E = 25

V = 21

$\therefore V(G) = 25 - 21 + 2$

$= 6$

$\therefore$ complexity is 6

(iii) independant paths

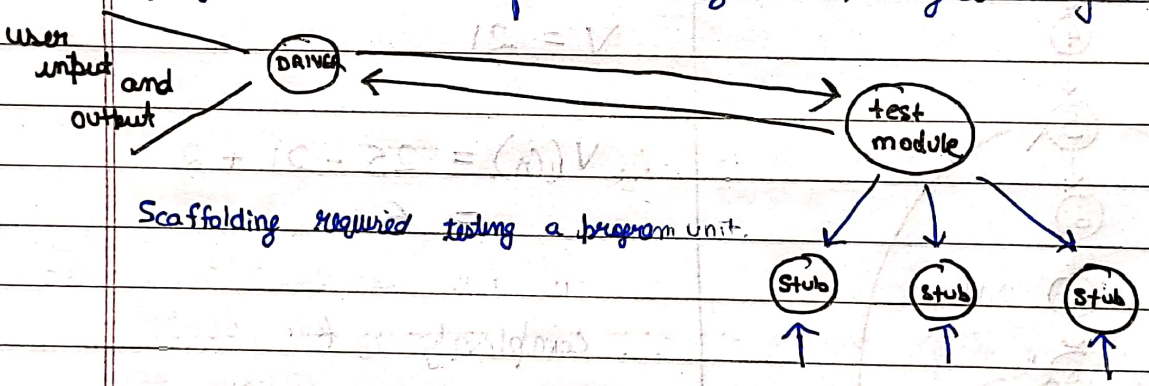
Date : / /
Page No.

[b] Level of Testing 2018E

Testing may be defined at different lvls of SDLC. The testing process runs parallel to software dev. Before jumping on the next stage, a stage is tested, validated and verified.
Testing separately is done just to make sure that there are no hidden bugs or issues left in the software. Its various lvls are :-

- UNIT TESTING

It is conducted by developers before handing over the code to the testing team. It involves testing individual units of software to ensure they function correctly according to specified requirements.
It is a white box testing to identify if individual components of program are error free - allowing for integration of them.



Scaffolding required testing a program unit.

- INTEGRATION TESTING

Integration testing involves combining individual software units which are tested as a whole group to ensure they work seamlessly together.
It aims to uncover any potential issues that may arise when these units are integrated, such as problems with passing data and updates

Its of three types:
→ top down integration testing
→ bottom up integration testing
→ Sandwich integration testing

Date : / /
Page No.

SYSTEM TESTING

It is the closest level to everyday experience where products are evaluated based on user expectations rather than strict specifications. It focusses on demonstrating performance and functionality rather than finding faults.

Software is subjected to special tests, considering hardware configs and various specifications such as performance, reliability and uptime. Choosing testcase should prioritize catastrophic failures over minor issues, replicate real-world usage and testing existing capabilities of modifying.

The main goal is to avoid disrupting current functionality while ensuring dependability and operational correctness.

(b) What is the difference between white and black box testing? Is determining test cases easier in black or white box testing? Is it correct to claim that if white box testing is done properly, it will achieve close to 100% path coverage. 2022 E [4][CO5]

WHITE-BOX TESTING	BLACK-BOX TESTING.
→ The developers can perform it	→ The test engineers perform it
→ "What the software is supposed to do, also aware of how it does it"	→ "What the s/w is supposed to do but is not aware of how it does it"
→ To perform WBT, we should know the programming language	→ To perform BBT, there is no need to have information of programming language.
→ We look into code and test logic	→ Verify functionality based on requirements.
→	
→ Test Design Techniques: Control flow testing, Data flow testing, Branch testing, Statement coverage, decision coverage, path testing	→ Test Design Techniques: Decision table testing, All-pairs testing, Equivalence partitioning, boundary value analysis, Cause-effect graph.
→ Can be applied at mainly Unit testing but can be in integration, system	Can be applied virtually to All levels of Software testing.

Date : / /
Page No.

Determining test cases is typically easier in white box testing because we have access to the internal code and can design tests based on code logic.

Even though most cases are covered, its not guaranteed to reach 100 % path coverage

5 a) What are the automation challenges that SQA (Software Quality Assurance) team faces while testing? 2020E

Automation challenges faced by the SQA team are :-

1.) Mastering the automation tool : SQA teams may struggle with learning and effectively using automation tools, which can vary in complexity, leading to change in tool proficiency.

2.) Reusability of Automation Script : Ensuring the automation scripts are modular and reusable across different test cases or projects can be challenging but is crucial for efficiency.

3.) Adaptability of Test Case for Automation : Adapting manual test cases to automation may require significant modifications, posing a challenge in maintaining test consistency and coverage.

4.) Automating Complex Test Cases : Handling cases with multiple inputs, conditional flow etc can be difficult to automate, and demand deep understanding of app's behavior.

UNIT 6 - Software Maintenance

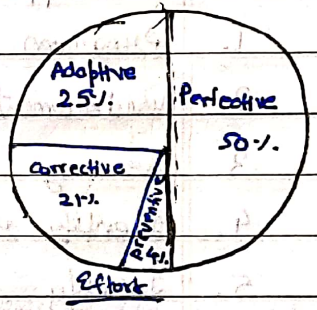
Date : / /
Page No.

b) What is Software Maintenance and why we need it? Also, what ae the categories of software maintenance? 2020E [4+4]

[b] What is software maintenance? Explain its types. 2018E

Software maintenance is a wide range of activites including error correction, enhancement, removal of obsolete features and optimization, all to preserve the value of software after its in operation.

This phase begins when software is delivered and operational, consuming a significant portion of SDLC cost (about 40-70 %). It addresses not only fixing issues but also adapting to changing needs and improving software quality over time, which can span for many years.



Types/Categories of Software Maintenance.

d) Perfective and corrective maintenance 2018S

1.) **CORRECTIVE MAINTENANCE:**
Corrective maintenance of software product may be essential either to rectify some bugs observed while the system is in use, or to enhance the performance of the system.

2.) **ADAPTIVE MAINTENANCE:**
This includes modifications and updations when the customers need the product to run on new platforms, or new OS, or when they need the product to interface with new hardware or software.

3.) **PERFECTIVE MAINTENANCE:**
It is the maintenance that a software product needs to support the new features that the users want or to change different types of functionalities of the system according to the customer demands

4.) **PREVENTIVE MAINTENANCE** (other type)
It includes modifications and updations to prevent future problems of the S/W.
It goals to attend problems, which are not significant at this moment but may cause serious issues in future.

b) The development effort for a software project is 500 person months. The empirically determined constant (K) is 0.3. The complexity of the code is quite high and is equal to 8. Calculate the total effort expended (as per Belady and Lehman model) if E 2018E

(i) maintenance team has good level of understanding of the project (d=0.9).
(ii) maintenance team has poor understanding of the project (d=0.1).
[4+4]

Date : / /
Page No.

K = 0.3
C = 8

Given,
Development effort (P) = 500 PM

i.) when (d = 0.9) → good lvl of understanding.

$M = P + K e^{(C-d)}$
$= 500 + 0.3\, e^{(8-0.9)} = 500 + 363.59$
$= 863.59 PM$

ii.) when (d = 0.1) → poor lvl of understanding.

$M = P + K e^{(C-d)}$
$= 500 + 0.3\, e^{(8-0.1)}$
$= 500 + 809.18 = 1309.18 PM$

Hence, it is clear that effort increases exponentially, if poor SE approaches are used and understandability of project is poor.

Q.5 Attempt any two:
(a) The development effort for a project is 600 PM. The empirically determined constant (K) of Belady and Lehman model is 0.5. The complexity of code is quite high and is equal to 7. Calculate the total effort expended (M) if maintenance team has reasonable level of understanding of the project (d=0.7). 2022E [4][CO6]

Given,
Dev effort, P = 600PM
emp:det. coeff, K = 0.5
complexity, C = 7
understanding, d = 0.7

so effort,
$\therefore M = P + K e^{(C-d)}$
$= 600 + 0.5\, e^{(7-0.7)}$
$= 600 + 0.5\, e^{6.3}$
$= 600 + 272.28$
$= 872.28 PM$

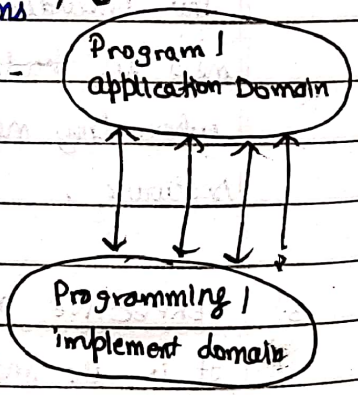
Date : / /
Page No.

6 Write short notes on any two:
a) Reverse Engineering 2018

Reverse Engineering is the process followed in order to find difficult, unknown and hidden information about a software system.
It is important since several software products lack proper documentation, and are highly unstructured, or their structure has degraded through a series of maintenance efforts. (maintenance cant be performed w/o a complete understanding of software system)

It is applicable to / has scope in:
1. Program comprehension.
2. Redocumentation or document generation
3. Recovery of design approach and details at any lvl of abstraction.
4. Identify reusable components.
5. Identify components that need restructuring.
6. Recovering business rules.
7. Understanding HLD.

It encompasses a wide array of tasks related to understanding and modifying software system. This array can be broken into # classes.
→ Mapping b/w application and program domains
→ Mapping b/w concrete and abstract levels.
→ Rediscovering high lvl structures
→ Finding missing links b/w program syntax and semantics
→ To extract reusable component



Levels of RE

Reverse engineering detects low level implementation constructs and replace them with their high level counterparts.
The process eventually results in an incremental formation of an overall architecture of the program.

Date : / /
Page No.

(b) What is the importance regression test selection? Discuss with the help of examples. 2022 E [4][CO6]

Regression testing is the process of retesting the modified parts of software and ensuring that no errors have been made / introduced into previous test code.

REGRESSION TEST SELECTION

Regression testing is very expensive activity and consumes significant amount of effort / cost. Many techniques are available to reduce this effort cost of :

1.) Reuse the whole test ~~case~~ suite
2.) Reuse the existing test ~~case~~ suite, but to apply a regression test selection technique to select an appropriate subset of the test suite to be run

Its important as it can effectively expose ~~ffaws~~ faults in the modified software, while minimizing the risk of omitting test cases which might uncover issues.

Example.)

Total No. of Pages: 01 *Roll No.....*

B. Tech. (CO) **Fifth *Semester***

Mid-Semester Examination ***(September 2023)***

CO-301 SOFTWARE ENGINEERING

Time: 1.5hrs ***Max. Marks: 20***

Note: ***Answer ALL questions. Assume suitable missing data if any.***

Q1. Describe Facilitated Application Specification Technique (FAST) and compare this with brainstorming sessions. [5][CO2]

Q2. Compare the waterfall and the spiral model of software development. [5][CO1]

Q3. Discuss the organization of SRS as per IEEE standard. [5][CO2]

Q4. Suppose a system for office automation is to be designed. It is clear from requirements that there will be five modules of size 0.5 KLOC, 1.5 KLOC, 2.0 KLOC, 1.0 KLOC, and 2.0 KLOC respectively. Complexity and reliability requirements are high. Programmer's capability and experience is low. All other factors are of nominal rating. Use COCOMO model to determine the overall cost and schedule estimated. Also calculate and schedule estimates for different phases.

Table 1: Coefficients for intermediate COCOMO

Project	a_i	b_i	c_i	d_i
Organic	3.2	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	2.8	1.20	2.5	0.32

Multipliers of different cost drivers

Cost Drivers	RATINGS					
	Very low	Low	Nominal	High	Very high	Extra high
Product Attributes						
SOFTWARE RELIABILITY	0.75	0.88	1.00	1.15	1.40	--
DATABASE SIZE	--	0.94	1.00	1.08	1.16	--
PRODUCT COMPLEXITY	0.70	0.85	1.00	1.15	1.30	1.65
Computer Attributes						
EXECUTIME TIME CONSTRAINT	--	--	1.00	1.11	1.30	1.66
MAIN STORAGE CONTRAINT	--	--	1.00	1.06	1.21	1.56
VIRTUAL MACHINE VOLATITY	--	0.87	1.00	1.15	1.30	--
TURNAROUND TIME	--	0.87	1.00	1.07	1.15	--

Cost Drivers	RATINGS					
	Very low	Low	Nominal	High	Very high	Extra high
Personnel Attributes						
ANALYST CAPABILITY	1.46	1.19	1.00	0.86	0.71	--
APPLITYLICATION EXPERIENCE	1.29	1.13	1.00	0.91	0.82	--
PROGRAMMER CAPABILITY	1.42	1.17	1.00	0.86	0.70	--
VIRTUAL MACHINE EXPERINECE	1.21	1.10	1.00	0.90	--	--
PROGRAMMING LANGUAGE EXPERIENCE	1.14	1.07	1.00	0.95	--	--
Project Attributes						
MODERN PROGRAMMING PRACTICES	1.24	1.10	1.00	0.91	0.82	--
USE OF SOFTWARE TOOL	1.24	1.10	1.00	0.91	0.83	--
DEVELOPMENT SCHEDULE	1.23	1.08	1.00	1.04	1.10	--

Lifecycle Phase Values of $\mathcal{M}_p$

Mode & Code Size	Plan & Requirements	System Design	Detailed Design	Module Code & Test	Integration & Test
Organic Small S=2	0.06	0.16	0.26	0.42	0.16

Lifecycle Phase Values of τ_p

Mode & Code Size	Plan & Requirements	System Design	Detailed Design	Module Code & Test	Integration & Test
Organic Small S=2	0.10	0.19	0.24	0.39	0.18

[5] [CO4]

Total no. of Pages: 02 Roll no.....

5th SEMESTER

B.Tech

END TERM EXAMINATION ***NOV/DEC-2023***

CO 301 SOFTWARE ENGINEERING

Time: 03:00 Hours ***Max. Marks: 40***

Note : All questions carry equal marks. Attempt all five questions.
Assume suitable missing data, if any.

Q.1 Attempt any two:

(a) Explain Agile and Rapid Application Development (RAD) life cycle models. [4][CO1]

(b) Discuss the difference between the following:

i. Functional and Nonfunctional requirements

ii. User and System requirements [4][CO2]

(c) What are risk management activities. How do we assess and control the risk? [4][CO4]

Q.2 You have been assigned the task of designing a software system for the Indian Railway Catering and Tourism Corporation (IRCTC). The system is aimed at facilitating online train ticket reservations for passengers.

Requirements are:

i. Users should be able to search for trains based on various criteria such as source, destination, date, and class.

ii. Users can book tickets for a selected train, choose their preferred class, and make reservations for multiple passengers.

iii. Registered users can log in to the system, view their booking history, and cancel reservations.

iv. Administrators should be able to manage train schedules, update seat availability, and generate reports on ticket bookings.

v. The system should handle online payments securely.

Draw the Use Case, Activity, Sequence, Class, State Chart Diagrams for the above software system. [8][CO3]

Q.3 Attempt any two:

(a) Discuss various key process areas of CMM at various maturity levels. [4][CO4]

(b) Explain the Boeh software quality model with the help of a block diagram. [4][CO4]

(c) Consider the program given in part 4 (b). List out the operators and operands along with their count as per token counting rules. Also, calculate the Halstead software science metrics such as program length, program volume, program level, difficulty, effort, and time. Consider the value of *stroud number* as 18. [4][CO4]

Q.4 Attempt any two:

(a) Consider the program for the determination of next date in a calendar. Its input is a triple of day, month, and year with the following range:

1<=month<=12

1<=day<=31

1900<=year<=2025

The possible output would be next date or invalid input date. Design robust and worst test cases for this program. [4][CO5]

(b) Consider a program given below for the selection of the largest of numbers.

```
void main()
{
 float a,b,c;
 printf("Enter three values\n");
 scanf("%f%f %f", &a,&b,&c);
 printf("\n Largest value is");
 if(a>b){
            if(a>c)
               printf("%f\n",a);
            else
               printf("%f\n",c);
         }
 else{
            if(c>b)
             printf("%f\n",c);
             else
             printf("%f\n",b);
      }
}
```

Design equivalence class testing and decision table testing techniques for the above program. [4][CO5]

(c) For the above given program in part 4 (b), calculate the McCabe's cyclomatic complexity. [4][CO5]

Q.5 Attempt any two:

(a) Annual Change Traffic for a software system is 25% per year. The development effort is 500PMs. Compute an estimate for Annual Maintenance Effort (AME). If the life time for the project is 5 years, what is the total effort of the project? [4][CO6]

(b) What is reverse engineering? Discuss various levels of reverse engineering. [4][CO6]

(c) Describe the Taute maintenance model. What are various phases of this model? [4][CO6]