Chapter : 1

Introduction to Software Engineering

(1) What is software engineering ? Explain about myths.

-> Software engineering is a systematic, diciplined, cost-effective technique for software devlopment.

-> In Software engineering a systematic and organized technique is adopted.

-> Based on the nature of problem and devlopment constraint various tools and techniques are applied in order to devlope quality software.

-> The software engineer has to adopt systematic and organized approach in order to produce high quality software. He is also responsible for selecting the most appropriate method for software devlopment.

* Myths :

Managment

(i) Myth : Using collection of standards and procedures one can build software.

Reality : We have all standard and ~~protoc~~ procedures with us for helping the devloper to build software.

It is not possible for software professional to build desired product. because the collection which we have should be complete, it should reflect modern techniques and more importantly it should be adaptable.
It should help software professional to bring quality in the product.

(ii) Myth : Add more people to meet deadline of project.

Reality : Adding more people in order to catch the schedule will cause the reverse effect on project.

Software project get delayed because we have to spend more time on educating people and informing them about project.

(iii) Myth : If a project is outsourced to a third party then all the worries of software building are over.

Reality : When a company need to outsourced the project then it indicates that the company does not know how to manage projects.

Sometimes, the outsourced project require proper support for devlopment.

Customer's Myth

(i) Myth : If the software requirments are changing contineously it is possible to accommodate these changes in software

Reality : It is true that software is a flexible entity but if contineous changes in the requirments have to be incorporated then there are chances of introducing more and more error. Similarly, additional resources and more design modifications may be demanded by the software.

(ii) Myth : A genral statment of objectives is suffecient to begin writting programs - we can fill the details later.

Reality : It is not possible each time to have comprehensive problem statment.

We have to start with genral problem statments : however by proper communication with customer the software professional can gather useful information.

Practitioner's Myth

(i) Myth : Once program is running then it's over.

Reality : Eventhough we obtain that the program is running major part of work is after delivering it to customer.

(ii) Myth : Working program is only work product for the software project.

Reality : The working program is major component of any software project along with it there are many other

elements that should be present in the software project such as documentation of software, guidlines for software support.

(iii) Myth : There is no need to documenting the software project. It slow down the devlopment process.

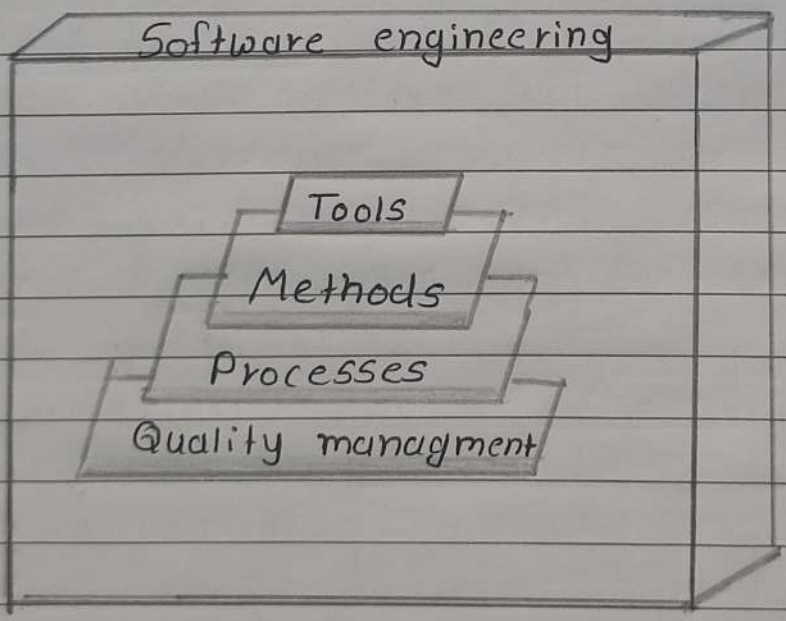
Reality : Documenting the software project help in establishing ease in use of software.

It helps in creating better quality.

Hence, documentation is not wastage of time but it is a must for any software project.

(2) Explain Software Engineering - A Layered technology.

-> Software engineering is a layered technology. Any software can be developed using these layered approaches. Various layers on which the technology is based are quality focus layer, process layer, method layer, tool layer.



-> Quality Managment : It is a backbone of Software engineering technology.

-> Process layer : It is a foundation of software engineering. Basically, process defines the framework for timely delivery of software.

-> Method layer : It is the actual method of Implementation is carried out with the help of requirment analysis, designing, coding using desired programming constructs and testing.

-> Software tools : It is used to bring automation in software devlopment process.

-> Thus Software engineering is a combination of process, method and tools for devlopment of quality Software.

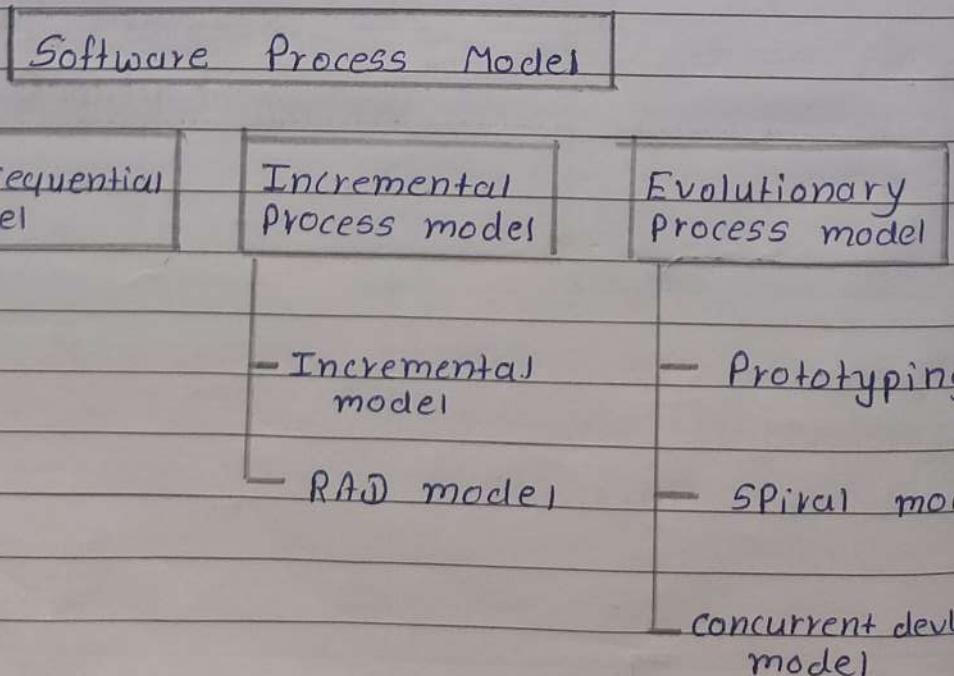
(3) Explain Software Process model. Compare Incremental and prototype process model.

-> The process model can be defined as the abstract representation of process. The appropriate process model can be choosen based on abstract representation of project.

-> The software process model is also known as software devlopment life cycle (SDLC) Model or Software paradigm.

-> These model are called prescriptive process model because they are following some rule for correct usage.

-> In this model various activities are carried out in some specific sequence to make the desired software product.



	Prototype Model	Incremental Process model
→	Some requirments are gathered initially, but there may be change in requirments when working prototype is shown to customer.	The requirements are precisely defined and there is no confusion about the final product of the Software.
→	Suitable for high risk projects.	Incremental model can't handle large project.
→	Cost of this model is low.	Cost of this model is also low.
→	Flexibility to change in this model is easy.	Flexibility to change in this model is easy
→	doesn't support automatic code genration.	It support automatic Code genration.
→	User involment is high	User involvment is only at begining.

(4) Compare Spiral and Prototyping process model.

->	Spiral Model	Prototyping Model
->	It is a risk driven software devlopment model and is made with features of incremental, waterfall, evolutionary prototyping model.	It is a software devlopment model in which a prototype is built, Tested and then refined as per customer need.
->	It is also preferred to as a rapid and close-ended prototype.	It is also reffered to as a meta model.
->	It does not emphasize risk analysis.	It takes special care about risk analysis and an alternative solution is undertaken
->	It is best suited when requirment of the client is not clear and is supposed to be changed.	It is best suited when customer requirments are clear.
->	Cost-effective quality improvment possible.	Cost-effective quality improvment not possible.
->	Large-scale project is maintained	Low to medium project size is maintained.

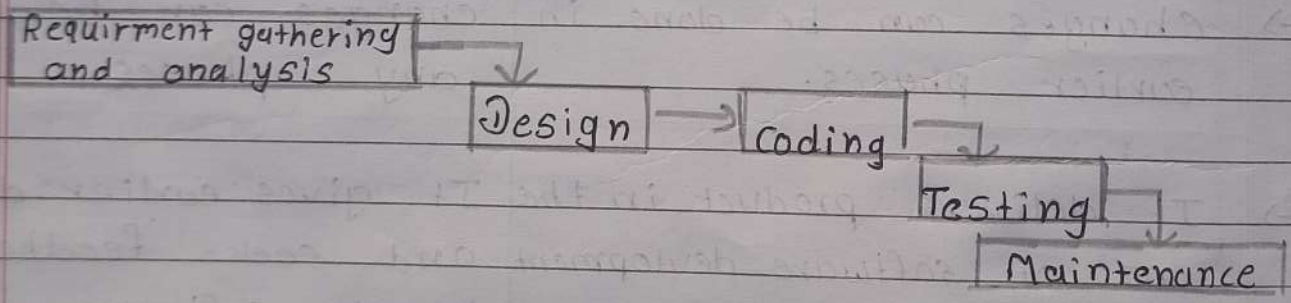
(6) Compare Waterfall and RAD Model

	Waterfall Model	RAD Model
→	It is also named as a classic or traditional model.	Rapid devlopment model is also named as Iterative model.
→	It is high-risk model.	It is low-risk model.
→	It requires large team to start software devlopment.	In this team size can be increased and decreased in devlopment progress.
→	Changes can be done in earlier phases.	changes can be done in any phases.
→	It delivers product in the end of software devlopment cycle.	It gives earlier deliveries and seeks feedback to Update software as needed
→	There is no user involvement in all the phases	There are user involment in all the pheses.

(6) Explain Linear Sequential Model. or Explain Waterfall Model.

→ The linear sequential model is also called as waterfall model or classic life cycle model. It is oldest software paradigm. This model suggest systematic sequential approach to software devlopment.

→ The software devlopment start with requirments gathering phase. Then progresses through analysis, design, coding, testing and maintanance.



→ Requirment Gathering and analysis : In this phase basic requirment of system must be understood by software engineer, who is also called Analyst.

- The information domain, function, behavioural requirments of the system are understood. All these requirment are then well documented and disussed further with the customer for reviewing.

→ **Design**: It is an intermediate step between requirment analysis and coding.

Design focus on program attribute such as : Data structure , Software architecture, Interface representation , Algorithmic details.

- The requirments are translated in some easy to represent form using which coding can be done effectively and efficiently.

→ **Coding**: It is a step in which design is translated into machine readable form.
- If design is done in sufficient detail then coding can be done effectively.
- Programs are created in this phase.

→ **Testing**: Testing begins when coding is done While perfoming testing the major focus is on logical internals of software.
- The purpose of testing is to uncover errors, fix the bugs and meet the customer requirments.

→ **Maintenance**: It is longest life cycle phase. When the system is installed and put in practical use then error may get introduced, correcting such error and putting it in use is the major purpose of maintenance activity. Similarly, enhacing system's service as new requirment are discovered is again maintenance of the system.

(7) Explain Spiral Model.

→ Spiral model is one of the most important software devlopment life cycle which provides support for risk handling.

→ In its diamatrical representation, it's look like a spiral with many loop.

→ The exact number of loop in spiral is unknown and can vary from project to project.

→ Each loop in spiral is called a phase of software devlopment process.

→ As the project manager dynamically determine the number of phases so the project manager has important role to devlope a product using spiral model.

→ The radius of spiral at any point represent the cost of the project so far, and angular dimension represents progress made so far in the current phase.

(i)	(ii)
objective and determination and identify alternative Solution.	Identify and resolve risk
(iii)	(iv)
Review and plan for the next phase	Devlope next version of product

→ Each phase of spiral model are divided into four parts.

(i) Objective determination and Identfy alternative solution : Requirments are gathered from customer and objectives are identified, elaborated, and analyzed at the start of every phase. The alternative solution possible for the phase are proposed in this quadrant.

(ii) Identify and resolve risk : During the second quadrant all the possible solution are evalute for select best possible solution. Then the risk associated with that solution are identified and risk are resolve using best possible strategy. At the end of this quadrant prototype is built for best possible solution.

(iii) Devlope next version of product:

During the third quadrant, the identified features are devloped and verify through testing.
At the end of third quadrant, the next version of software is avalaible.

(iv) Review and ~~next~~ plan for next phase:

In fourth quadrant, customer evalute the so far devloped version of the software. in the end planning for next phase is started.

-> Risk Handling: The most important feature of spiral model is handling these unknown risk after project started.
Such risk resolutions are easier done by devloping prototype.

* Advantage

-> Risk handling: Spiral model is the best devlopment model to follow due to risk analysis and risk handling at every phase.

-> Good for large Product : It is recommended to use Spiral model for complex and large product.

-> Flexibility in requirments : Change request in requirments at later phase can be incorporated accurately by using this model.

-> Customer satisfaction : Customer can see the devlopment of the product at early phase of the software devlopment.

* Disadvantage

-> Complex : The spiral model is much more complex then other SDLC model.

-> Expensive : Spiral model is not suitable for small project as it is expensive.

-> Dependibility on Risk Analysis : The sucessfull completion of project is very much dependent on Risk analysis. Without very higher experienced experts, it is going to be failure to devlope proje

-> Difficulty in time managment : As the number of phases unknown at the start of the project, so time estimation is very difficult.

(8) Compare product and process

	Process	Product
→	while process is a set of sequence steps that have to be followed to create project	It is final production of project.
→	Process is focused on completing each step being devloped.	Product focus on final result.
→	Process consistently follows guidlines	Product are followed firm guidlines.
→	Process tends to be long-term.	Product tends to be short-term.
→	Purpose of this is to make quality of project better.	Main goal of project completing work succesfully
→	Devloped by process engineers.	Devloped by software engineers.
→	output of proces is product.	Devlopment occurs by following process.
→	Ex. Program devloped for Parsing the input.	Ex. A word processing software.

Chapter : 2

Agile Devlopment

(1) Explain Agility, Process model and Principal.

-> Angility

- Agility is dynamic, content specific, aggresively change embracing and growth oriented.
- It encourage team structure and attitude that makes communication among team member, between technologist and bussiness people, between Software engineers and their managers more simply.
- It emphasizes rapid delivery of operational software and de-emphasize the importance of intermediate work product.
- It recognize that planning and uncertain world has its limits and that a project plan must be flexible.
- Agility can be applied to any software.

→ Agile Process

- It is difficult to predict in advance which software requirment will persist and which will change.
  It is equally difficult to predict how customer priority will change as the project proceeds.

- For many types of software design and construction are interleaved.
  It is difficult to predict how much design is neccessary before construction is used to prove the design.

- Analysis, design, construction and Testing are not as predictable as we might like.

→ Aglie Process Model

- Extream programming (XP)
- Adaptive Software Devlopment (ASD)
- Dynamic System devlopment Method (DSDM)
- Scrum
- Crystal
- Feature Driven devlopment (FDD)
- Agile modeling (AM)

-> Principles.

- Satisfy customer by early and contineous delivery of valuable software.
- Welcome to changing requirment, even late in devlopment.
- Deliver working of software frequently. within shorter time span deliver the working limit.
- Bussiness people and devlopers must work together throughput the project.
- Build project around motivated individual. Give them enviroment and support they need.
- Working software is primary meausure of progress
- Agile process promote sustainable devlopment. The sponser, devloper and user should be able to maintain and a constant pace indefinitely.
- Attention to technical excellence and good design enhances agility.
- Simplicity must be maintained while devloping the project using this approach.
- The best architectures, requirments and design must be self organizing team.
- At regular interval team reflect on how to become more effective.

(2) Explain Extreme Programming

→ It is most widely used agile process model.

→ XP uses an object oriented approach as its prefer devlopment paradigm.

→ XP defines (4) frame work activities Planning, Design, Coding and Testing

* Planning

→ Begins with the creation of a set of stories.

→ Each story is written by customer and is placed on an index card.

→ Customer assign value to story.

→ Agile team asseses each story and assigns a cost.

→ stories are grouped to for deliverable increment.

→ A commitment is made on delivery date

→ Divide the work into small parts. Keep the part of nearly constant length.

→ Stand up meeting must be conducted for current outcome of project.

* Designing

→ Simple design always take less time than complex design. It is always good to keep thing simple to meet requirment.

→ For answering the tough technical problem create spike solution. Goal of this solution should be to reduce technical risk.

→ Refactoring means reduction in the redudancy elimination of unused functionality, redesign the design. This will improve quality of project.

* Coding

In extereme programming the customer not only helps the devloper team but it should be the part of the project.

All the code to be include in the project must be coded by groups of two people working at the same computer. This will increase quality of coding.

By having collective code owenership approach the everyone contributes new ideas and not only single person become the bottleneck of the project. Anyone can access code and change any line of code to fix a bug.

* Testing

- The test framework that contains a automated test case suit is used to test the code.
  All the code must be tested using using unit testing before release.

- As soon as one task is finished integrate it into the whole system. Again after such integration unit testing must be conducted.

- Working overtime lose the spirit and motivation of the team. Conduct the release planning meeting to change the project scope or to reschedule the project.

(3) Explain Scrum Aglie devlopment model.

→ Scrum is an agile process model which is used for devloping the complex software system.

→ It is a light weight process framework that can be used to manage control the software devlopment using iterative and incremental approach.

→ In SCRUM emphasize is on software process pattern. The software process pattern defines a set of devlopment activites.

(i) Backlog : It is basically a list of project requirments or feature that must be provided to the customer. The iteams can be included in the backlog list at any time. Product manager analyses this list and update priorities as per requirments.

(ii) Sprint : These are the work units that are needed to achieve the requirments mentioned in the backlogs. Typically the sprints have fixed durantion or time box Thus sprints allow the team members to work in stable and short term enviroment.

(iii) Meetings : These are 15 minutes daily meetings to report the completed activities and plan for next activities. Following questions are mainly discussed.

- What are task done since last meeting?
- What are issues that team is facing?
- What are next activity that are planned?

(iv) Demo : During this phase, the software increment is delivered to the customer. The implemented functionality which is demonstrated to the customer. Note that demo focuses on only implemented functionalities of the software product.

* Roles

(i) Scrum Master : It leads the meeting and analyses the response of each team member. The potential problems are discussed and solved in meeting with the help of master.

(ii) Team member : These are the persons working in a team to devlope the software solution.

* Principles

- There are small working teams on the software devlopment process. Due to this there are maximum communication and minimum overhead.
- The task of ~~partition~~ people must be partitioned into small and clean partition.
- The process must be accommodate the technical or business changes if they occure.
- During the product building the constant testing documentation must be conducted.
- The SCRUM process must be produce the working model of the product whenever required.

* Advantage

- It brings transparency in project.
- It provides flexibility towards the changes.
- Productivity can be improved.
- There is improved communication and minimum overhead in devlopment process.

* Disadvantage

- Some decision are hard to track in fixed time.
- There are problems to deal with non-functional requirments of the system.

Chapter : 3

Managing Software Project

(1) Explain Software project managment and W5HH principle.

-> Software project managment is an art and science of planning and leading software project. It is sub-discipline of project managment in which software projects are planned, implemented, monitored and controlled.

* W5HH Principle

-> Why is the system being devloped ?

- Enable all parties to asses the validity of business reason for software work.
It also justify the expenditure of people, time and money.

-> What will be done ?

- Answer to this question will help the software team member to identify the project task and milestone.

→ When will be done?

- Answer to this question will help to prepare the project schedule with identified project task and milestones.

→ Who is responsible?

- Roles and responsibilities required to devlope the system can be defined.

→ Where are they organizationally located?

- All the roles can not be defined within the software team itself.
There are customer, user and stack holders holding some responsibility.

→ How will be job will done technically and managerially?

- Managment and technical strategy must be defined.

→ How much of each resource needed?

- Deriving the project estimate or cost of the project.

(2) Explain Product Metrics.

-> Product metrics are computed from data collected from analysis and design models, source code and test case.

- Measure : It is quantative indication of the extent, ammount, dimension or size of some attribute of product or process.
- Metrics : It is the degree which a system, component, or process a given attribute. The software metrics relate serval measures. Ex. average number of errors found per review.
- Indicators : Indicators means combination of metric that provides insight into the software process, project or product.

-> Attributes of Software Metrics.

- Simple and computable : Metric should be easy to compute and should not be time consuming activity.
- Empirically and conveincing : It should be immediate and can be derived based on observations.

- Consistent and Objective: Metric should produce clear result. Anybody should get the same result by using these metric when same set of information is used.

- Consistent in units and dimensions: The mathematical units and dimensions used for metric should be consistent.

- Programming Independent: Metric should be based on analysis model, design model and program structure. It should be independent of programing language.

- High quality feedback: Metric should provide a way to produce high quality software product.

(3) Explain software Project planning.

→ The first step to be taken in project managment is project planning.
There are five major activities :
(i) Project estimation
(ii) Project Scheduling
(iii) Risk analysis
(iv) Quality managment planning.
(v) change managment planning.

→ Software estimation begins with description of the scope of software product.

→ For the meaningful project devlopment scope must be bounded. The problem for which the product is to be built is then decomposed into a set of smaller problems. Each of these estimated using historical data and previous experiences as a guide.
The two important issue complexity and risk are considered before final estimate is made.

→ Estimation of resources, cost and schedule for a software engineering efforts requires :-
- Experience
- Access of good historical information
- Courage to commit to quantitative prediction when quantitative information is avalaible.

(4) Risk Analysis and Managment.

→ Risk: Risk denotes the uncertainly that may occure in the choice due to past actions. and risk is something which causes heavy loss.

→ Risk Managment: It is refer to the process of making decisions based on an evalution of the factor that threats to the business.

* Types of Risk

(i) Project Risk: It arise in software devlopment process and then they basically affect budget, schedule, staffing and resources. when project risk become sever then the total cost of project increased.

(ii) Technical Risk: It affect quality and timeliness of project. If technical risk become reality then potential design implementation, verification and maintanance problem get created.

- It occurs when problem become harder to solve.

(iii) Business Risk : When feasibility of software product in in suspect then business risk occure.

- Market Risk : When a quality software built but if there is no customer for this product then it is called market risk.
- Strategic Risk : When product is built and if it is not following the company business polices then such a product brings strategic risk.
- Sales Risk : When product is built but how to sell is not clear then such a situation brings sales risk.
- Managment Risk : When senior managment and responsible staff leaves the organization that it occurs.
- Budget Risk : Lossing overall budget of the project is called budget risk.

⇒ Risk Analysis : Risk analysis in project managment is a sequence of process to identify the factor that may affect project sucess. These includes risk identification, analysis of risk and risk managment control.

(5) Explain RMMM.

→ RMMM stands for risk mitigation, monitoring and managment.

→ Risk Mitigation

- It means preventing the risks to occure. Following steps to be taken for mitigation the risk.

1. Communicate with concerned staff to find of probable risk.

2. Find out and eliminates all those cause that can create a risk.

3. Devlope policy in an organization which will help to continue the project even though some staff leaves the organization.

4. Maintain corresponding document in timely manner.

5. Conduct timely reviews in order to speed up the work.

6. For conducting every critical activity during software devlopment, provide additional staff if required.

→ Risk Monitoring

- In this following things must be monitored by project manager.

1. The approach and the behaviour of team member as pressure of project varies.
2. The degree in which the team perform with the spirit of Team-work.
3. The type of co-operation among team member
4. The types of problem that are occuring.
5. Avalaibility of jobs within and outside an organization.

→ Risk Managment

- Project manager performs this task when risk becomes reality. If project manager is succesful in applying project mitigation effectively then it becomes very much easy to manage risk.

- Ex. Consider that many people are leaving company then if sufficient additional staff is avalaible, if current devlopment activity is known to everybody in team, if latest and systamatic documentation is avalaible then any any new comer can easily understand current devlopment activity.

(6) Explain Project scheduling process and time line chart (Gantt chart).

-> While scheduling a project the manager has to estimate the time and resource of the project.
All activities in the project must be arranged in coherent sequence.

-> The schedules must be continually updated because some uncertain problem may occur during the project life cycle.

-> During the project scheduling the total work is seprated into small activities. And time required for each activity must be determine by project manager. For efficient perfomance some activities are conducted in parallel.

-> The project manager should be aware of the fact that: Every stage of project may not be problem-free. Some typical problems in project development stage are:

- people may leave or remain absent.
- Hardware may get failed.
- Software resource may not be avalaible.

-> Various resources required for project are:

- Human effort
- Sufficient disk space on server
- Specialized hardware
- Software technology
- Travel allowance required by project staff.

* Time Line chart

-> In software project scheduling the timeline chart is created. The purpose of timeline chart is to emphasize the scope of individual task. Hence set of task are given as input to time line chart.

-> It is also called Gantt chart.

-> The timeline chart can be devlope for entire project or it can be developed for individual function.

-> In time line chart:

- All the tasks are listed at leftmost column.
- The horizontal bar indicates time required by task.
- When multiple horizontal bar occure at the same time on the calendar, then than concurrency can be applied for perfoming task.
- Diamonds indicates the milestone.

Task	Sep			Oct			Nov			Dec			Jan		
	6	13	20	3	10	17	7	12	20	9	18	26	1	10	26
1.2															
1.2.1															
1.2.2															
1.2.3															
1.2.4															
1.3															
1.3.1															
1.3.2															
1.3.3															
1.4															
1.4.1															
1.4.2															

[Time line chart]

Task	Planned start	Actual start	planned end	Actual end	Effort assign
1.2.1	6-Sep	6-Sep	13-sep	13-Sep	Harsh
1.3.1	7-Nov	7-Nov	12-Nov	12-Nov	Yash
1.4.2	9-dec	9-dec	26-Jan	26-Jan	Suchi

[Project table]

(7) Difference between Coupling and cohesion

Coupling	Cohesion
→ It is a concept of Inter module	It is a concept of Intra - module.
→ It represent relation b/w modules	It represent relation within a module.
→ Increasing coupling is avoidad for software.	Increasing cohesion is good for software.
→ It represent independence among modules.	It represent functional strengths of module.
→ losely coupling gives the best software	Highly cohesive gives the best software.
→ Modules are connected to other modules.	The module focuses on a single thing

Chapter : 4

Requirments Analysis and Specification

(1) What is software Requirments specification (SRS) ? Explain characterstics.

→ The software requirment document is the specification of the system.
It should include both a definition and specification of requirment.
It is not a design document.
As far as possible, it should set of what the system should do rather than how it should do it.

→ The software requirment provide a basis for creating the SRS.

→ The SRS is useful in estimating cost, planning team activities, perfoming task and tracking team progress throught the development activity.

* Characterstic

-> Correct : The SRS must be correct. That means all the requirments must be correctly mentioned, or the requirment must be realistic by nature.

-> Complete : To make SRS complete, it should be specified what a software designer wants to create a software.

The SRS is said to be complete only if :

(i) When SRS consists of all the requirments related to functionality, perfomance and design.

(ii) When labels and corresponding reference are mentioned for all the figures.

(iii) When expected response to the input data is mentioned by considering validity and invalidity of an input.

-> Stability : In SRS, it is not possible to specify all the requirments. The SRS must contain all the essential requirments. Each requirment must be clear and explicit.

-> Unambigeous : When requirments are understood correcty then only unambegious SRS can be written. Unambegious specification means only one interpretetion can be made from specified requirments. In other words, there should be an unique interpretation of each statment in SRS.

-> Consistent : If there are no conflict in specified requirments then SRS is said to be consistent.
Three types of conflict they may occure:
(i) Logical and temporal conflict.
(ii) characterstic conflict of real world object.
(iii) Two different description about same real world object.

-> Verifiable : The SRS should be written in such manner that the requirments that are specified within it must be satisfied by software.

-> Traceable : If origin of requirment is properly given or reference of the requirments are correctly mentioned then such a requirment is called as traceable requirment.

(2) Difference between Verification and Validation:

→

	Verification	Validation
→	It includes cheking documents, design, code and programs.	It includes testing and validating the actual product.
→	Verification is the static testing.	Validation is dynamic testing.
→	It does not include execution of code.	It includes the execution of code.
→	Method use in this are reviews, inspection and desk-cheking.	Method used in validation are black and white box testing.
→	It can find the bug early stage of devlopment.	It can find bugs which can not find by verification
→	It comes before Validation	It comes after verification.

(3) Explain SRS documentation for Library Managment System.

→

Library Managment System
Prepared by : Harsh Porwal

(1) Introduction

1.1 purpose: This document gives the detail functional and non-functional requirments for Library managment system. The purpose of this document is that the requirments mentioned in it should be utilized by the software devloper to implement the system.

The hotel and user will be able to use this SRS as a test to see if the software engineer will be constructing the system to their expectations.

1.2 Scope

The library managment system will automate the major library options. various sub system will be student account managment system, Book issueing System, Stock maintanence system and managment system.

1.3 Overview

This system is provide an easy solution for the students, staff and librarian for accessing and managing the books in the library.

(2) Overall Description

The library managment system is developed for handling the activities for various users such as student, staff, liberarian and library staff.

2.1 Product Perspective

This is standlone system.

2.2 Hardware Interface

This system is placed in personal computer in the library.

2.3 Software Interface

There are various database in the system. These database can be created in Oracle. or MySQL. The hotel books and student/staff information is maintain by the LMS.

The book database include the Title of book, ISBN number, publisher, edition, number of copies, price, status.

The student/staff database will include information such as First name, last name, Address, phone number, class, name of book issued, expected date of return book.

(3) Functional Requirment

Functional requirment will define the fundamental functioning that the system should perform.

- Record all the details of student, staff, book.
- Display the search result, if the user searches for some books or article.
- Allow to select a book for issuing.
- Record the issue and return date of book.
- Allow modification in student/staff information.
- Allow to create a free account for new student.
- Allow to delete account if student leave college.
- Display the avalaibility of book.
- Allow to assign a strong password to user.
- keep track of purchased iteams stock.
- Allow to reserve book if currently unavaible.
- Genrate daily and monthly record for stock.
- Allow add and delete book in records.

(4) Interface Requirments

4.1 GUI

GUI 1: Login screen will be displayed so that user will enter user name and password.

GUI 2: Main menu will be displayer for create account, search, Book maintanence and report genration.

GUI 3: When the create account button is clicked then the information form will be displayed. which contains details such as first name, last name, phone number, class, etc.

GUI 4: If the user select for the search then user will be allowed to search for book.

4.2 Hardware Interface

The system should be embedded in the computer in library.

4.3 Software Interface

The system must be interfaced with Oracle or access database.

(5) Perfomance Requirment

- It defines the response time for system functionality.
- Load time for GUI should not be more than 4 seconds.
- Login must be verified within 7 second.
- Search query may be procced within 3 second and response time must be given.

(6) Design constraint

- The system must be design as a standlone system and must be run on windows based system. System can be devloped in Java or visual basic. The database can be implemented in Oracle or MySQL.

(7) Non-Functional Attribute

7.1 Security

System administrator provide password to log on the system. Password can be changed by system administrator.

7.2 Reliability

System must be reliable for unauthorized access.

7.3 Availability

System should be avalaible during college hours.

7.4 Maintainability

There should be the facility to modify information about student, book and staff.

(8) Operational Scenarious

If new student arrives then librarien creates account : issueing him library card, user name , password.

If student already have an account he searches for Books . if the book is avalaible then he makes request liberarian to issue book.

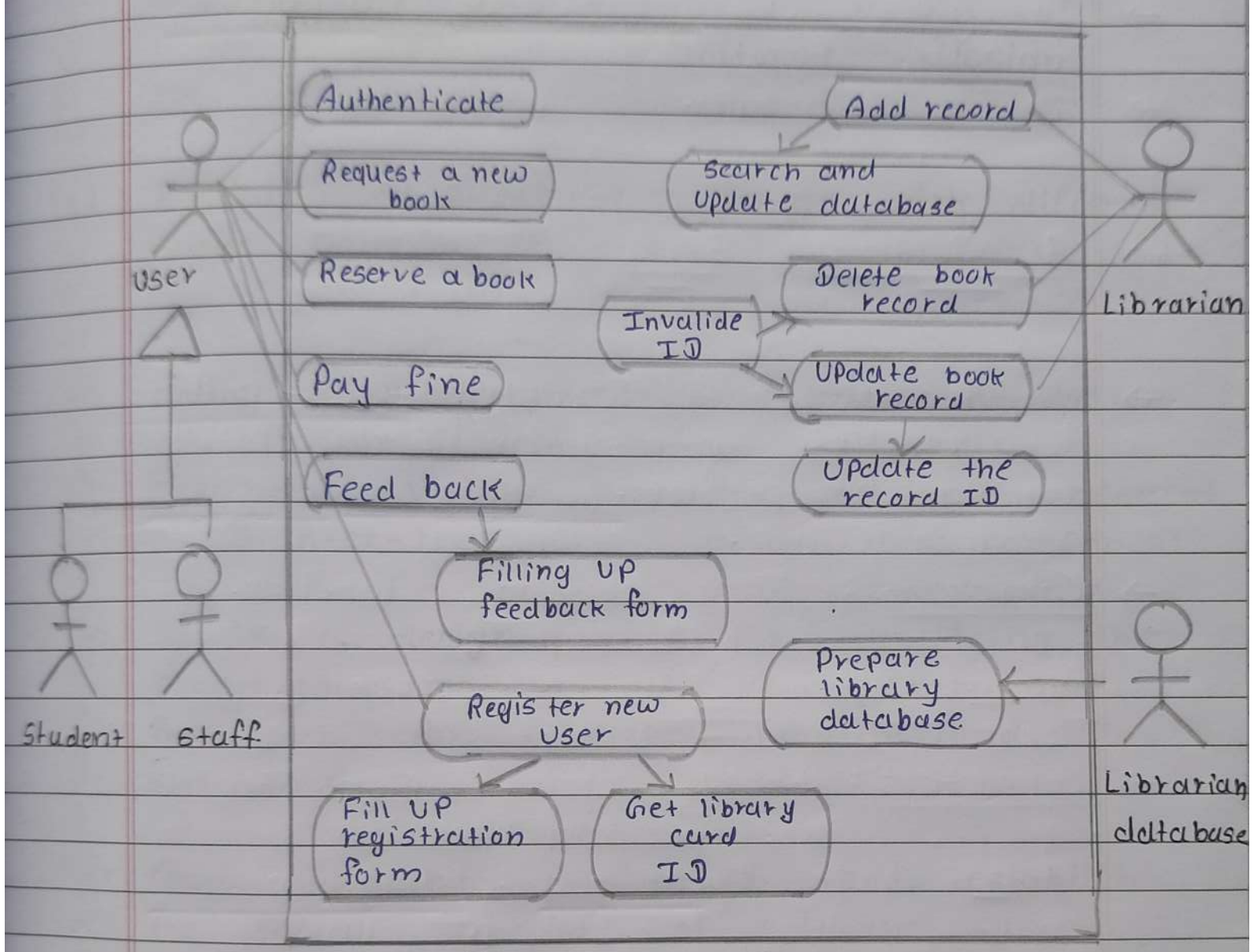
If book is not avalaible then member reserve the book . On avalaibility of book, It is issued to a student.

If the student does not return book on specified date then he has to pay fine.

Librarian update the stock and genrate monthly record.

(9) Preliminary Schedule

System must be implemented within 6 months.



[Use Case Diagram]

(4) Explain SRS for College managment

(5) Explain SRS for Student Result managment.

Chapter : 5

Software Design

(1) Explain architectural design with different style pattern.

-> Software architecture is a structure of system which consist of various components, externally visible properties of these component and inter-relationship among these component.

-> It is a design process for identifying the subsystem making up the system and framework for subsystem control and communication.

-> Goal of architectural design is to establish the overall structure of software system. It represent the link blw design specification and actual design process.

-> It gives the representation of the computer based system that is to be built. Using these system model even the stack holders can take part in software devlopment

-> Some early design decision can be taken using software architectural hence system perfomance and operations remains under control.

→ It gives clear cut idea about the computer based system which is to be built.

* Architectural Style

→ Architectual model or style is a pattern for creating the system architecture for given problem.

→ Components : They perform a function.

Ex. Database, client, server

→ Connectors : Enable communication. They define how the component can co-ordinet.

Ex. Calls, Event broadcasting, pipes

→ Constraints : Define how system can be integrated.

→ Semantic - Models : Specify how to determine a system's overall properties from the properties of its part.

→ Commonly used Style :

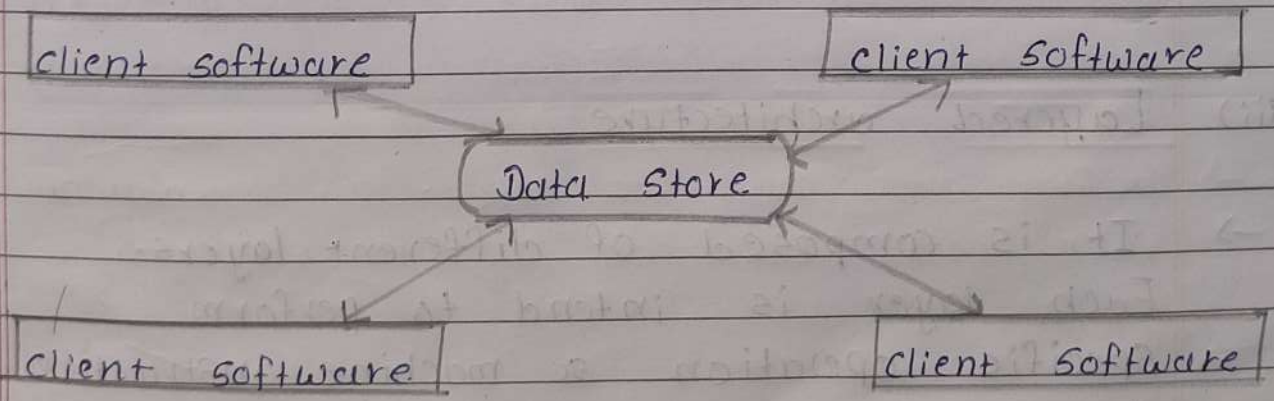
(i) Data centered
(ii) Data Flow
(iii) call and Return
(iv) Object - oriented
(v) Layered.

(i) Data centered architecture

→ In this the data store at center of architecture and other component frequently access it by perfoming add, delete and modify operation.

→ The client software request data from centeral repository. without any change in data or without any change in action of software action.

→ Data centered architecture posses the property of Interchangability.
Interchangability means any component from the architecture can be replaced by a new component without affecting the working of other component.

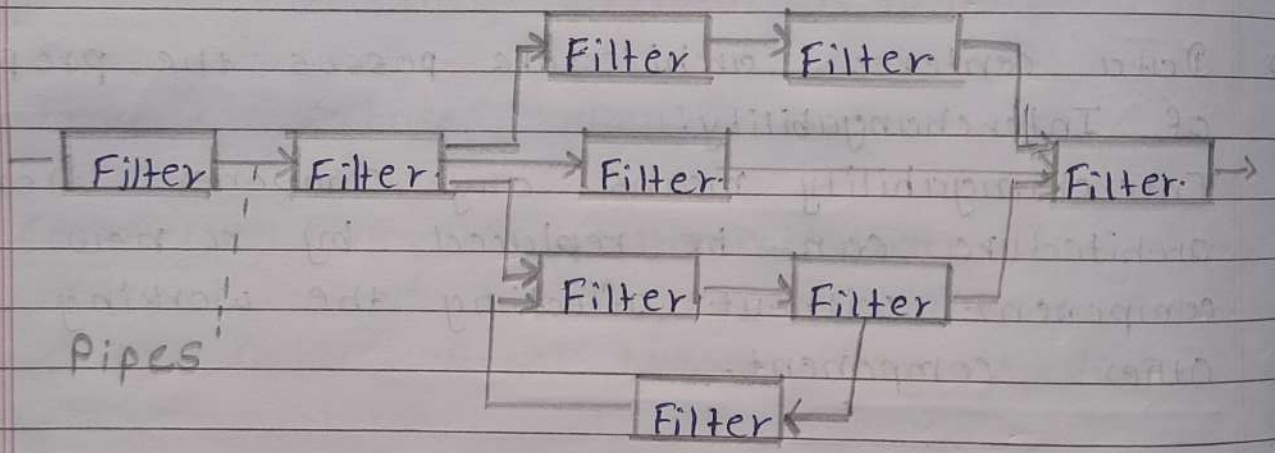


(ii) Data flow architecture

→ In this architecture series of transformation are applied to produce the output data.

→ The set of component called filters are connected by pipes to transform the data from one component to other.

→ This filters work independantly without bothering about the working of neighbour filter.



(iii) Layered architecture

→ It is composed of different layers. Each layer is intend to perform specific operation so machine instruction set can be genrated. various component in each layer perform specific operation.

-> Outer layer is responsible for perfoming the user interface operation.

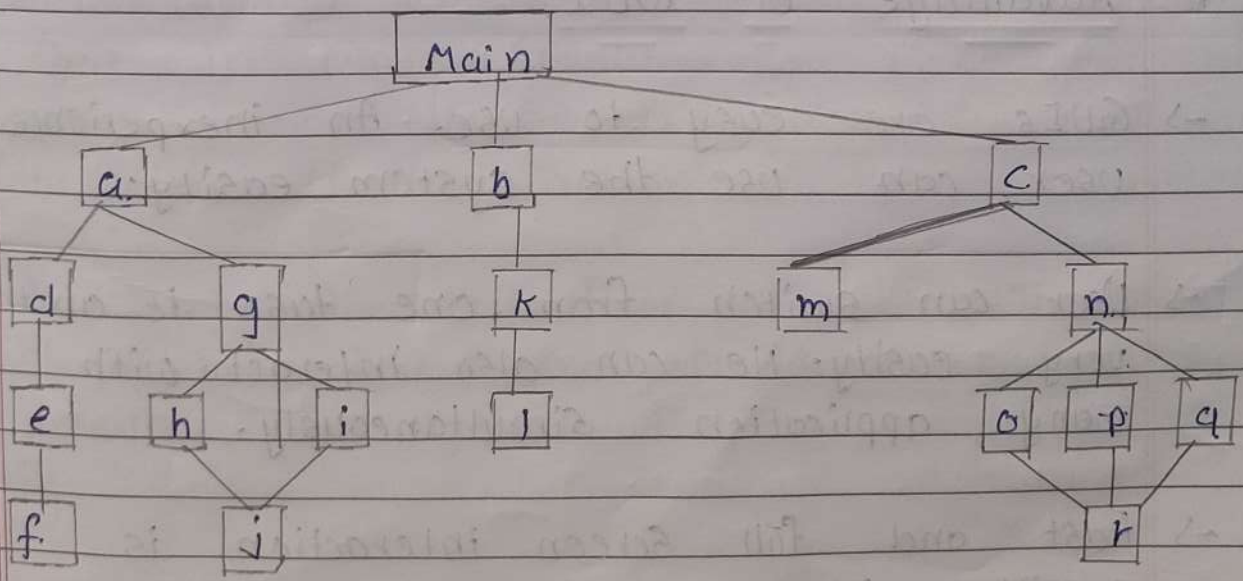
-> Components in intermediate layer perform utility services and application of software function.

(iv) Call and Return Architecture.

The program structure can be easily modified or scaled. The program structure is organize into modules within the program.

In this architecture how modules call each other. The program structure decompose the function into control hierarchy when a main program invokes number of program component.

In this architecture the heirarchy control for call and return represented.



(2) Explain User Interface design and Discuss Golden rules.

→ In user interface design the effective interaction of system with user is provided.

→ Typically system user judges a system by its interface rather than its functionality. A poorly designed interface leads improper use of system. Many software system can not be used because poorly designed user interface.

→ Most of the user in business systems interact with the system using graphical user interface. However text based interface is also used.

* Advantage of GUI

→ GUIs are easy to use. An inexperienced user can use the system easily.

→ User can switch from one task to another very easily. He can also interact with many application simultaneously.

→ Fast and full screen interaction is possible with user.

* Golden Rules

→ There are three golden rules for UID :-

(i) Place the user in control.
(ii) Reduce the user's memory load.
(iii) Make the interface consistent.

(i) Place the user in control

→ Define interaction mode in such way that does not force to user into unnecessary and undesired action : The user should be able to easily enter and exit the mode.

→ Provide flexible interaction : Different people will use different interaction mechanism, some might use keyboard, mouse, command, etc. hence all interaction mechanism should be provided.

→ Allow user to customize interaction : It is observe that in while handling user interface certain action need to be done repeatedly. It saves the time if these action are collected in Macro.

→ Hide technical details from the user : The user should not be aware of the internal technical details of system.

(ii) Reduce the User Memory load.

→ Reduce demand on short term memory : When users are involved in some complex tasks the demand on short term memory is significant.

→ Establish meaningful defaults : Always initial set of default should be provided to the average user, if a user needs to add some new feature then he should be able to add some new required features.

→ Define shortcuts that are intitive : For quick handling of system such short cuts are required in the user interface.

→ The visual layout of the interface should be realistic : Anything you represent on screen if it is a metaphor for real word entity then user would easily understand.

→ Disclose the information gradually : There should not be bombarding of information on user. It should be presented to the user in systamatic manner.

(iii) Make the interface consistent

→ Allow the user to put current task into a meaningful context : Many interface have dozens of screen. So it is important to provide indicators consistently so that the user know about the doing work. The user should also known from which page has navigated to current page and from the current page where can navigate.

→ Maintain consistency across family of application: The development of some set of application all should follow the and implement same design ; rules so that consistency maintained among applications.

→ If certain standards are maintained in previous model of application do not change it untill and unless it is necessary.

(3) Explain cohesion and coupling.

→

* Cohesion

→ With the help of cohesion the information hiding can be done.

→ A cohesive module perform only 'one task' in software producer with little interaction with other modules.

→ Different type in cohesion

(i) Coincidentally cohesive: The modules in which the set of tasks are related with each other loosely then such modules are called coincidentally cohesive.

(ii) Logical cohesive: A module that performs the tasks that are logically related with each other is called

(iii) Temporal cohesion: The module in which the task need to be executed in some specific time span is called

(iv) Communication cohesion: when processing elements of a module share the data then such module is called

* Coupling

-> Coupling effectively represents how the modules can be connected with other module.

-> Coupling is a measure of interconnection among modules in a program structure.

-> Coupling depends on interface complexity between modules.

-> The property of good thing coupling is that it should reduce or avoid change impact and ripple effect.
It should also reduce the cost in program changes, testing and maintenance.

-> Different types of coupling

(i) Data coupling : Data coupling is possible by parameter passing or data interaction.

(ii) Control coupling : The modules share related control data in control coupling.

(iii) Common coupling : Common data or a global data is shared among this module.

(iv) Content coupling : It occurs when one module makes use of data or control information maintained in another module.

(4) Difference beetween procedural and object oriented design.

Procedural	Object oriented
→ The basic abstraction, which are given to the user, are real world function.	The basic abstraction are not the real world function.
→ State information is often represented in a centerlized shared memory.	state information is not often represented in a centerlized shared memory.
→ We decompose in the function.	We decompose in class level.
→ Top-down approach	Bottom-Up approach.
→ Begins by considering the use case diagram and scenarious.	Begins with identify object and classes.

Chapter : 6

Software Coding and Testing

(1) What is software testing? Why it is important?

-> Software testing is a critical element of software quality assuarance and represent the ultimate review of specification, design and coding.

-> Testing is a process of executing a problem with the intend of finding an error.

-> A good test case is one that has high probability of finding an undiscovered error.

-> A successful test is one that uncovers an as yet undiscovered error.

* Principles

-> All test should be traceble to customer requirment

-> Test should be planned long before testing begin.

-> The pareto principle can be applied to software testing - 80 % of all error uncovered during testing will likely be traceble to 20 % of all program modules.

→ Testing should begin "in the small" and progress towards testing "in the large".

→ Exhaustive testing is not possible.

→ To be most effective, testing should be conducted by independent third party.

* Importance

→ Testing is a process that requires more efforts than any other activity.

→ Testing is a set of activity that can be planned in advance and conducted systematically.

→ If it is conducted haphazardly, then only time will be wasted and even more error may get introduced.

→ Hence perfoming testing by adopting systematic strategies is very much essential in during devlopment of software.

* Testing Strategies

-> We begin by 'testing in the small' and move towards 'testing in the large'.

-> There are four various testing strategies :

- Unit - Testing : In this technique are applied to detect the errors from each software component individually.
- Integration Testing : It focuses on issues associated with verification and program construction as component begin interacting with one another.
- Validation Testing : It provides assucarance that the software validation criteria meets all functional, behavioural and perfomance requirments.
- System Testing : All system elements forming the system is tested as a whole.

(2) Explain System Testing.

→ The system test is a series of test conducted to fully computer based system.

→ Various types of system testing :

(i) Recovery Testing
(ii) Security Testing
(iii) Stress Testing
(iv) Perfomance Testing

(i) Recovery Testing : It is intended to check the system's ability to recover from failures.

→ In this type of testing the software is forced to fail and then it is verified whether the system recover properly or not.

→ For automated recovery then reinitialization, checkpoint mechanisms, data recovery and restart are verified.

(ii) Security Testing : It verifies that system protection mechanism prevent improper penetration or data alternation.

-> It also verifies that protection mechanism built into the system prevent unauthorized internal or external access or damage.

(iii) Stress Testing : Determines breakpoint of a system to establish maximum service level.

-> In this system is executed in a manner that demands resources in abnormal quality, frequency or volume.

-> Sensitive testing in which it is tried to uncover data from a large class of valid data that may cause instability or improper processing.

(iv) Perfomance Testing : It evalutes the run time perfomance of the software.

-> In this resource utilization such as CPU Load, throughput, response time, memory usages can be measured.

-> For big system involving many user connecting to server perfomance testing is very difficult.

-> Beta testing is usefull for perfomance testing.

(3) Explain White box Testing and Black box testing.

*> White-Box Testing

-> In white box testing the procedural details are closely examined.

-> In this testing the internals of software are tested to make sure that they operate according to specifications and designs.

-> Thus major focus of white box testing is internal structure, logic paths, control and data flow, condition, loop, etc.

-> Working :

- Input : Requirments, Functional specification, design documents, source code
- Processing : Perfoming risk analysis for guiding through entire process.
- Proper test Planning : Designing test case so as to cover entire code.
- Output : Preparing final report of entire testing process.

→ Techniques :

- Statment Coverage : This is aimed at execising all programing statments with minimal tests.
- Branch Coverage : This is running a series of tests to ensure that all branches are tested atlest once.
- Path Coverage : This technique corresponds to testing all possible path which means that each statment and branch is covered.

→ Advantages :

- Forces test devloper to reason carefully about implementation.
- Reveals error in hidden code.
- Spots the dead code or other issue with respect to best programing practices.

→ Disadvantages :

- Expensive as one has to spend both time and money to perform white box testing.
- Every posibility that few lines of code are missed accidentally.
- In-depth knowledge about programing language is necesary to perform white box testing.

* Black box Testing

→ Black box testing is use to determine that the software functions are operational.

→ It is tested wheather input is accepted properly and output is correctly produced.

→ Major focus of black box testing is on functions, operations, external interfaces, external data and information.

→ Types :

- Functional Testing : It is related to functional requirments of a system. it is done by software tester.
- Non-Functional Testing : It is related to non-functional requirment such as perfomance, scability, Usability.
- Regression Testing : It is done after code fixes, upgrades or any other system maintanance to check new code has not affected the existing code.

→ Techniques

- Equivalance class Testing : It is used to minimize the number of possible test case to an optimum level while maintain reasonable test coverage.
- Boundary value Testing : It is focused on the value testing is focused at boundaries.

This technique determines wheather a certain range of values are acceptable by the system or not. It is very useful in reducing possible test cases.

- Decision table Testing : A decision table put causes and their effects in matrix.

There is unique combination in each column.

→ Advantages :

- Tester does not need to have more functional and programing knowledge.
- It is efficient for implementing test in larger system.
- Tests are executed from user's and client's point of view.
- Test cases are easily reproducible.

→ Disadvantages :

- Without function knowledge it is difficult to implement.
- Sometime reason of test failure can't be detected.
- Some programs in application are not tested.
- Working large sample space consume lots of time.

(4) Explain Integration testing.

→ Integration testing is a process of testing the interface blw two modules.

→ A group of independent components are tested together to ensure their quality of their integration unit.

→ The objective is to take unit tested components and build a program structure that has been directed by software design.

→ It can be done by using 3 approach.

(i) Big - Bang approach
(ii) Top - down approach
(iii) Bottom - up approach

(i) Big Bang approach : Combining all the modules once and verifying the functionality after completion of individual model testing.

(ii) Top - down approach : Testing take place from top to bottom.

- High level modules are tested first then low level modules and finally integrated the low level to high level module to ensure the system is working as intend.

- Stubs used as a temporary module, if a module is not ready for integration testing.

(iii) Bottom-up approach : Testing take place from bottom to up.

- Lowest level modules are tested first then high level module and finally integrated the high level modules to low level modules to ensure system is working as intend.
- Drivers are used as a temporary module, if module is not ready for integration testing.

(5) Difference between black box and white box testing.

→

Black box Testing	White box Testing
→ In this internal structure or the program or code is hidden and nothing is known about it:	In this the tester has knowledge about the internal structure or code of the software.
→ Implementation of code is not needed.	Code implemantation is necessary.
→ Mostly done by software tester.	Mostly done by software devloper.
→ It is functional test of the software.	It is strutural test of the software.
→ No programing knowledge required	programing knowledge is required.
→ It is behaviour testing of the software.	It is logic testing of the software.
→ It is applicable to higher level testing of software	It is applicable to lower level of software testing
→ least time consuming	most time consuming
→ Not suitable for algo testing	Suitable for algo testing

Chapter : 7

Quality assurance and Managment.

(1) Explain software quality assurance. (SQA)

-> It is a simply way to assure quality of software. It is the set of the activities which ensure processes, producers as well as standards are suitable for project and implemented correctly.

-> SQA is a process which works parallel to devlopment of software.
It focuses on improving the process of devlopment of software so that problem can be prevented before they become major.

-> SQA has :
(i) Quality managment approach
(ii) Formal technical reviews
(iii) Multi testing strategy
(iv) Effective software engineering technology
(v) Measurment and reporting mechanism

-> SQA activities :

(i) SQA Managment plan : Make a plan for how you carry out sqa throughout. project. Think about which set of software engineering activities are the best for project.

(ii) Set the check Points : SQA team should set checkpoints.
Evalute the perfomance of the project on the basis of collected data on different check points.

(iii) Multi testing strategy : Do not depend on single testing approach.
When you have lots of testing approach available use them.

(iv) Measure change impact : The changes for making correction of an error sometimes re introduce more errors keep the measure of impact of change on the project.
Rest new change to change check the compatibility of this with whole project.

(v) Manage good relation : In the working enviroment managing good relation with other teams involved in the project devlopment is mandatory.
Bad relation of sqa team with progyramers team will impact directly and badly on the project. Don't play politics.

(2) Explain Formal Technical Reviews. (FTR)

-> FTR is a software quality assurance activity perfomed by software engineer.

* Objectives

-> FTR is useful to uncover errors in logic, function and implemantation for any representation of the software.

-> Purpose of FTR is to ensure that software meets specified requirments.

-> It also ensure that the software is represented according to predefined standard.

-> It helps review the unformity in software devlopment process.

-> Its make the project more manageable.

-> Purpose of FTR is to enable junior engineers to observe the analysis, design, coding and testing approaches more closely.

* The Review meeting

→ Every review meeting should be conducted by considering the following constranit :

- Involvment of people: Between 3 and 5 people should be involved in the review.
- Advance preparation : It should occur but it should be very short at the most 2 hours of work for each person can be spent in this preparation.
- Short duration : The duration of the review meeting should be less than 2 hour.

→ Rather than attemping to review the entire design, walkthroughs are conducted for modules or for small groups of modules.

→ The review meeting is attended by the review leader, all reviewers and producer.

→ The review leader is responsible for evaluting the product for its readiness. The copies of product material is then distributed to reviewers.

* Review Reporting and Record keeping

-> During the FTR, the reviwer actively records all issues that have been raised.
At the end of meeting these all raised issues are consolidated and review issues list is prepared.
Finally, FTR summary report is produced.

* Review Guidlines

-> Review the product not manufacture.
-> Take written notes for record purpose.
-> Limit the number of participant and insists upon advance preparation.
-> Devlope a checklist for each product that is likely to be reviewed.
-> Allocating resources and time schedule for FTRs in order to maintain time schedule.
-> Set an agenda and maintain it.
-> Seprate the problem areas, but do not attemp to solve every problem notes.
-> Limit debate and rebuttal.

(3) Explain CMM and Six Sigma. (or) Explain quality standards.

→ * CMM

→ The capability Maturity Model (CMM) is used in assessing how well an organization's process allow to complete and manage new software projects.

→ Various process maturity levels are :

Level 1: Initial : Few processes are defined and individual efforts are taken.

Level 2: Repeatable : To track cost schedule and functionality basic project managment processes are established. Depending on earlier success of projects with similar applications nesecarry process discipline can be repeated.

Level 3: Defined : The process is standardized, documented and followed. All the projects use documented and approved version of software process which is useful in devloping and supporting software.

Level 4: Managed : Both the software process and product are quantatively understood and controlled using detailed measures.

Level 5 : Optimizing : Establish mechanisms to plan and implement change. Innovative ideas and technologies can be tested.

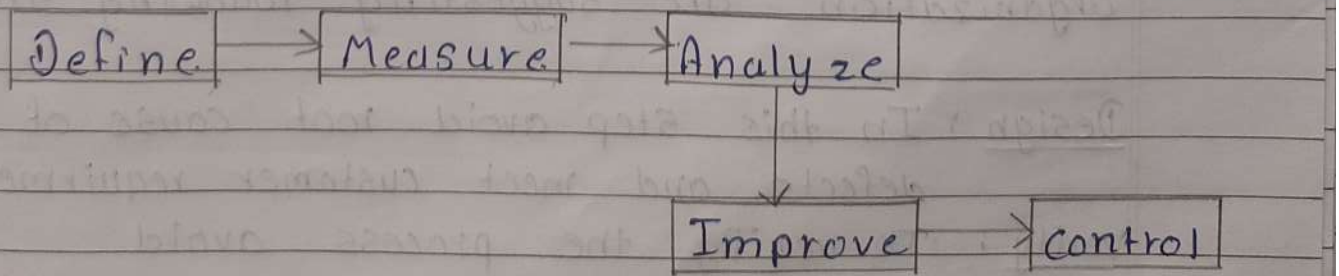
→ Thus CMM is used for improving software project.

* Six sigma

→ It is widely used statical software quality assurance strategy.

→ It is a business driven approach to process improvment, reduced cost and increased profit.

→ The word six sigma is derived from six standard deviation occurrence.



→ Define : The customer requirments, project goals and deliverable are defined by communicating the customers.

→ Measure : The existing process and its output is measured in order to determine current quality perfomance.

→ Analyze : In this phase defect metrics are analyzed in order to determine the few cause.

→ Improve : By eliminating the root cause of defects the process can be improved.

→ Control : The process can be controlled in such a way that the cause of defect can not be re-introduced.

→ For newely devloping software, some organization are suggesting following steps:

Design : In this step avoid root cause of defects and meet customer requirments.

Verify : To verify the process avoid defects and meet customer requirments.

(4) What is software quality ? List down different software quality metrics.

→ In, software engineering software managment is done based on some software metrics where these software metrics are reffered as the measure of various characterstics of software.

→ In software engineering SQA assures the quality of the software.
Set of activities in SQA are continuously apply throughout the software process.
Software quality is measured based on some software quality metrics.

→ There is number of metrics avalaible based on which software quality is measured.

→ There are few most useful metrics :

(i) Code quality
(ii) Reliability
(iii) Perfomance
(iv) Usability
(v) Correctness
(vi) Maintainability
(vii) Integrity
(viii) Security

Chapter: 8

Software Maintenance and Configuration Managment.

(1) Define software maintanance and ity types.

→ Software maintenance is an activity in which program is modified after it has been put into use.

→ In software maintanence usually it is not preffered to apply major software changes to system's architecture.

→ Maintanence is a process in which changes are implemented by either modifying the existing system architecture. or by adding new components to the system.

→ Usually the system requirments are changing and to meet these requirments some charges are incorporated in the system.

→ There is a strong relationship between system and its enviroment. When a system is installed in an environment, it changes that enviroment.

→ The maintain system remains useful in their working enviroment.

* Types

(i) Corrective Maintenance : Means the maintenance for correcting software faults.

(ii) Adaptive Maintenance : Its for adopting the change in enviroment.

(iii) Perfective Maintenance : Means modifying and enhancing the system to meet new requirments.

(iv) Preventive Maintenance : Means changes made to improve future maintainbility.

(2) Explain Re-Engineering and Reverse engineering.

→* Re-Engineering

→ It means re-constructing and re-writting part or all the software engineering system.

→ It is needed for application which require frequent maintenance.

→ Process activities :

Source code translation : In this phase code is converted into new language.

Reverse Engineering : Under this activity program is analyzed and understood throughly.

Program structure Improvment : Restructure automatically for understandability.

Program modularization : Program structure is recognized.

Data re-engineering : Finally clean up an restructure system data.

→ Advantage

Reduced risk : Re-engineering allows the devloper to eliminate certain constraint on system. This help reducing the risk of failures.

Reduced cost : The cost of re-engineering is often significantly less than the cost of devloping new software.

* Reverse Engineering

→ Reverse engineering is a process of design recovery. In reverse engineering the data, architectural and procedural information is extracted from source code.

→ There are three important issues.

(i) Abstraction Level : This level helps obtaining the design information from the source code. It is expected that abstraction level should be high in reverse engineering. High abstraction level helps software engineer to understand program.

(ii) Completness level : It means detailing of abstract level. The completness decrease as abstraction level increased.

Ex. From a given source code listing one can easily devlope a complete procedural design representation. But it is very difficult to devlope complete set of ~~procedural design~~ data flow diagram or entity relationship diagram. The completness in reverse engineer devlops the interactivity.

(iii) Directionality level : It means extracting the information from source code and give it to software engineer.
The directionality can be one way or two way.
The one way directionality means extracting all the information from source code and give it to software engineer.
The two way directionality means the information taken from source code is fed to a re-engineering tool that attemps to restructure or regenrate old programs.

(3) Explain SCM with it's process.

-> SCM stands for software configuration managment.

-> SCM is a set of activity carried out for identifying, organizing and controlling changes throughout the life cycle of computer software.

-> During the devlopment of software change must be managed and controlled in order to improve quality and reduce risk.

-> The primary objective of SCM are :

Configuration Identification : Identify the items that define software configuration.

Change control : Manage changes to one to more items.

Version control : Facilitate to create different versions of application.

Configuration Authentication : To ensure that the quality of the Software is maintained as the configuration envolves over the time.

* Change control

-> Changes in any software projects are vital.

-> Sometimes, introducing small changes in the system may lead to big problems in product.

-> Similarly, introducing some changes may enhance the capabilities of the system.

-> For a large software engineering project, uncontroled change create lot of chaos. For managing such changes, human procedure or automated tool can be used.

* Change control Process

Step 1: First of all there arises a need for the change.

Step 2: The change request is then sumbited by user.

Step 3: Devlopers evalutes this request to asess technical merit, potential side effects, and overall impact on system function and cost of the project.

Step 4: A change report is then genrated and presented to the change control authority.

Step 5: The change control authority is a person or a group of people who makes final decision on status and priority of the change.

Step 6: An engineering change order is genrated when the change gets approved. In ECO the change is described the restrictions and criteria for review and audit are mentioned.

Step 7: The object that needs to be changed is checked out of the project database.

Step 8: The changes are then made on the corresponding object and appropriate SQA activities are then applied.

Step 9: The changed object is then checked in to the database and appropriate version control is made to create new version.

* Version Control

-> Version is an instance of a system which is functionally distinct in some way from other system instance.

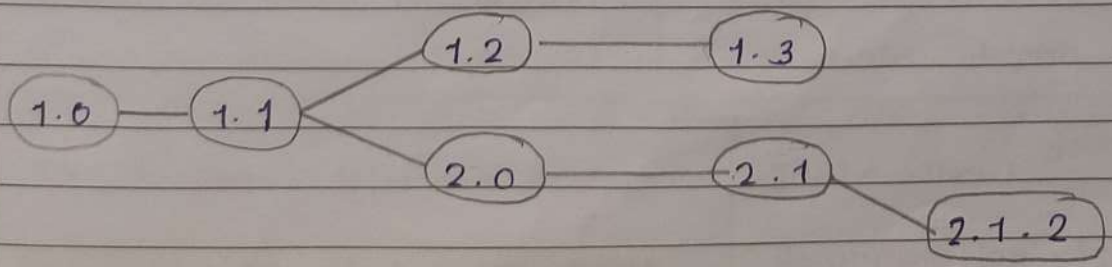
-> Version control works to help manage different version of configuration times during the devlopment process.

-> Configuration managment allows a user to specify the alternative configuration of the software system by selecting appropriate version.

-> Certain attributes are associated with each software version. These attributes are useful in identifying the version.
Ex. attribute can be date, creator, customer.

-> In practice version needs an associated name for easy reference.

-> Different version of software can be shown by an evolution graph as each version of software system is collection of software configuration items.



Chapter : 9

DevOps

(1) What is devOps ? Explain it's principle and practice.

-> DevOps is a practice in which devlopment and operation engineers participate together in entire lifecycle activities of system devlopment from design, implementation to product support.

-> The term devOps is derived from Software DEvelopment and Information technology OPreationS.

-> DevOps promotes a set of processes and method from the three departmant Devlopment, IT operation, Quality assuarance. that communicate and collabrate together for devlopment of software system.

-> DevOps enhances the organization's perfomance, improves the productivity and efficiency of devlopment and operation teams.

-> Bringing the two team together centerlize the responsibility on the entire team and not specific individuals working.

* Principles

(i) Customer centric action : Continuous feedback of end user or real customer should get reflected in all the development activities. The product and services must be developed in such a manner that the customer satisfaction is always protected.

(ii) Focus on end result : The developers and engineers must work by keeping the complete picture of the end product in mind.

(iii) End to End responsibility : All the service required to develope, maintain and monitor must be done by the same team. This help in improving the quality of end product.

(iv) Cross functional autonomous teams : If same team work on all the phases such as development, maintenance and monitoring then it helps in improving the quality of end product.

(v) Continuous improvment : It is normally done for optimizing the speed of processing, reduced cost and minimized waste. The objective of continuous improvment is to bring best quality product.

(vi) Use of automated tools and techniques : It is use in improving the speed and reducing the devlopment efforts. Hence automate the things wherever possible.

* Practices

(i) Prespective consideration : The difference beetween the devloper and participant prespective must be considered.

(ii) Flexibility : Organization must have flexibility to switch between the delivery of product using Devops value and without Devops values.

(iii) Integrate changes : It must be possible to accommodate the required changes in the system.

(iv) Agility : The process as well as tool section needs to be agile according project needs.

(v) Transparency : There must be transparency about the goals, delivery plan and activities between devloper and operational engineers.

(2) Explain the 7 C's of DevOps lifecycle.

→ (i) Planning : During this phase the planning for identifying skills, resources required and outcome is made.

(ii) Devlopment : In this phase, devlopment sketch plan is prepared and programing techniques are identified. Before integration devlopment team would write bunch of code for three or four months.

(iii) Integration : It is the practice of quickly integrating newly devloped code with main body of code that is to be released. Integration saves a lots of time when team is ready to release code.

Continuous integration process from a DevOps involve checking your code in, compiling it into usable code and running some basic validation testing.

(iv) Deployment : It is practice of deploying all the way into production without any human intervention. Team that utilize continuos delivery don't deploy untested code, code runs through automated testing before it get pushed out to production.

(v) Testing : Continuous testing is the process of executing automated test cases as part of software delivery pipeline. It's goal is to obtain immediate and continuous feedback on the business risk associated with a software release candidate.

(vi) Delivery and monitoring : With continuous, delivery, every code change is built, tested and then pushed to a non-production testing or staging environment. There can be multiple, parralel test stages before a production deployment.

Continuous monitoring is a process of monitoring the applications continuously to check its service, perfomance and security. It can be done using different tools.

(vii) Continuous feedback : This is process which allows for an immediate response from your customers for your product and its features and helps you modify accordingly.

(3) Difference between DevOps and agile.

Agile.	DevOps
→ Agile is to devlope software in small iteration and be thus able to adapt to the changing customer need.	Deliver technology to business units in a timely fashion and ensure the technology runs without interruption.
→ Rapid devlopment approach.	It is not rapid devlopment approach.
→ Focus on software devlopment and release.	Focus is not only software devlopment, its release but on it safest deployment in working environment.
→ Communication in this is informal and in the form of daily meetings.	Communication, specifications, document are involved and it is formal and not occur on daily basis.
→ Team is small in nature.	Large team size and multiple team required.
→ It is about software devlopment.	It is about software devlopment and managment
→ Documentation is not important.	Documentation is very much important.

(4) Challenge with DevOps implementation.

-> (i) Cultural change : Any organization must promote the collaborative culture for effective implementation of DevOps. The leaders should bring transparency in the work process. There must be positive atmosphere in an organization.

(ii) Bringing silos and sector together : There is a tendancy of maintaining silos and sector within the team. While devlopers are constantly writing piece of code in order to build a system, tester perform a through analysis to ensure product stability before the final delivery to the customer.

(iii) Giving up legacy System : Organization must give up the old or outdated systems and must adopt modern and efficient systems. Handling m new system along with the old existing system in the organization can be challenging many times.

(iv) Tool selection confusion : There are number of tools available in market. Which tempt the devOps devloper to choose different tool. This lead to changing and updating the tool frequently. Changing and updating tool becomes difficult with change in strategy.

(v) Different metrics : During product development process each team measure their perfomance using different metrics. When the project is to be implemented using DevOps technology, there must be common metric to be used to measure the perfomance.

(vi) Resistance to change : The teams are normally unwilling to accept the changes. They are resistance to change their prefered working style. They do not open up the silos to other teams.
This make difficult to other teams to work and may create an unhealthy environment.

(vii) Process challenges : DevOps strategy does not define specific rules to implement the process or to use the tool. The only restriction is to follow the project goal. Although this gives lot of flexibility and chances to innovative methodologies, it may lead to challenging situation like confusion and dispute among the team member of some strategies.

Chapter : 10

Advance Topics in Software Engineering

(1) Component based software engineering (CBSE)

-> The CBSE is an approach of defining, implementing, integrating lossely coupled independent component into system

-> Software component is a software element. conforms to a component model and can be independantly deployed and composed without modification according to a composition standard.

* Characterstics

-> **Standardized** : The components confirms to some Standardized component model. These Standards are for defining component interfaces, meta data, deployment and documentation.

-> **Independent** : Component must not depend upon other component while deployment is made.

-> **Deployable** : Component must have an ability to operate as standlone entity and it should be self 'contained.

→ Composable : For a component to be composable, all external interaction must take place through publicly defined interfaces.

→ Documented : Component must be fully documented so that any user can decide wheather particular component is useful to meet system requirments.

* Advantage

→ It becomes easy to construct understable and maintable software.
→ Component are independent entities and they don't interfere in other component operation.
→ Component implementaion is hidden.
→ Communication among the component is well defined.

* Disadvantage

→ It is difficult to verify component without source code.
→ Quality of component can not be verified.
→ It is not possible to predict the emergent properties of component composition.
→ It is difficult to make trade offs between the feature of various components.

(2) Explain Computer Aided software engineering (CASE)

→ Case is an implementation of computer facilited tools and method in software devlopment.

→ Case is used to ensure a high-quality and defect free software.
Case ensures a check pointed and disciplined approach and helps designers, devloper, tester, manager and others to see project milestone during devlopment.

→ Case also helped as a warehouse for documents related to project, like business plan, requirment and design specification.

→ Case is used in conjuction with process model that is choosen.

* Types of CASE Tool:

(i) Diagramming Tools : It helps diagrametic and graphical representation of the data and system. It represent system elements, control and data flow among different software components and system structure in pictorial form.

(ii) Computer Display and Report genration :

It helps in understanding the data requirments and relationship involved.

(iii) Analysis tools : It focuses on inconsistent, incorrect specification involved in diagram and data flow.
It checks in collecting requirment, automatically check for any irregulity, imprecision in the diagram.

(iv) Central Repository : It provides the single point of storage for data diagram, report and document related to project managment.

(v) Documentation Genrator : It helps genrating user and technical documentation as per standards. It creates documents for technical user and end user.

(vi) Code Genrator : It aids in auto genration of code, including definitions, with the help of design, document and diagrams.

(3) Explain client server software engineering.

→ It is the application in such a way that the server consists of a set of service which are demanded by clients. That means client demands for the service and server provides these service to client. The client and server are seperate the project.

* Two-tier architecture

→ In this, one server might be connected more than one client. The simplest client server architecture is called two tier client server architecture in which the application executes on two layer client layer and server layer.

→ Thin client model : In this data managment and application logic is implemented on the server and the client is responsible for running the presentation software.

→ Fat client model : In this model the server is responsible for only data managment. The application logic and presentation software is executed on client it self.

Ex. ATM is an example of this client model.

- The ATM is connected to the server! User operate ATMs and the information is proceed at the client side.
- Data managment part handled by server.

* Three-tier architecture

→ In this architecture the presentation application processing and data managment are logically seperate process and execute on different processor.

Ex. Internet banking is an example of this.

- In this client browse the web page and request for banking transaction.
- Then the application processes the request and communicate the database server to verify the request for transaction.
- The database server store bank database and executes the queries.
- Result of this query return to application server.