DT:-12.07.16

Software Engineering

software 2 types
(1) system software.
(2) application software.
(FB, whats app)

ent a; - delimeter.

2 civil mall
SE

Software Engineering

Software engineering is a discipline of computer science that attempts to apply operation, modification and maintainance of software component of various systems.

OR

software engineering is an engineering branch associated with development of software product using well defined scientific principles methods and procedures and the outcome is reliable and efficient software product.

- It discusses systematic and cost effecti techniques for development.
- The techniques are results from inovati n and lessions learnt from past experience or mistake.

Engineering

A engineering is an application of well understood of scientific method to the construction, operation, modification and mai tainance of useful devices and systems.

System:

System is an assembly of component that interact on some manner among themselves

DT:- 13.07.16.

Software engineering is neither a form of science nor arts

(1) Past experience systematically arranged
ex $(a+b)^2 = a^2+2ab+b^2$

(2) Unique sol^n.

(3) practical amount
eg:- amount.

(4) Subjective judgement qualitative.

History or evolution of software engineering

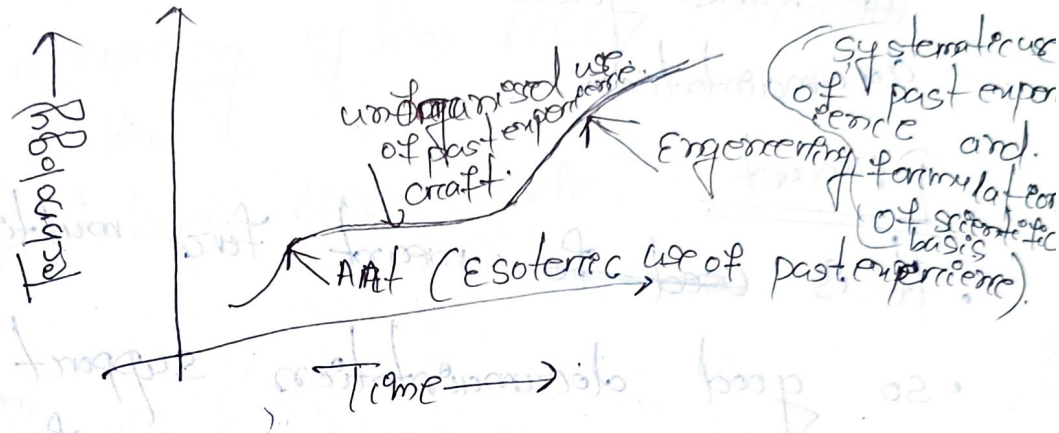
Software eng. is nothing but evolution of art to an engineering discipline.

(Exploratory style / Build and fix)
or
(outsource - without employee attached)

In the built and fix (Exploratory style) normally a poor quality program is quickly developed with out making any specification plan or design. The different implementation that are subsequently noticed while using or testing are fixed.

art (personal knowledge).
↓
craft (group).

Evolution of technology with time.



Software crises

- The factors causing the software crises are rapidly increasing problem size.
- Lack of adequate (sufficient) training in software engineering techniques. increasing skill shortage.
- Low productivity improvements.

Software development projects

① Program Vs product

Program

programs are developed by individuals for their personal use.

- These are small in size and has limited functionality.
- Author is the sole user of the program and he himself maintains the program.
- It lacks good user interface and proper documentation.

Product

- It is ~~used to~~ meant for multiple users.
- so good documentation support is needed.
- As it has a large no. of users so must be systematically designed, carefully implemented and throughly tested.
- It contains code as well as associated documents (printer)
- It is too large to be developed by individuals, hence done by a group of engineers working in a team.

DT:- 14.07.16

Types of software development projects

(1) software product development project.
(2) Outsourced project.

(1) : It may be - generic product (os, database)
or
- custom developed product.

Software product development project

- A software product can either be generic or custom developed.
- Generic products are used by large and diverse range of customers.

Ex:- operating system like microsoft window

Database like oracle.

custom

outsource product.

- It can make good commercial sence. for a company to out source some part of its work as a project to another company.

- perseve development

perceived complexity

Human cognetive knowlege

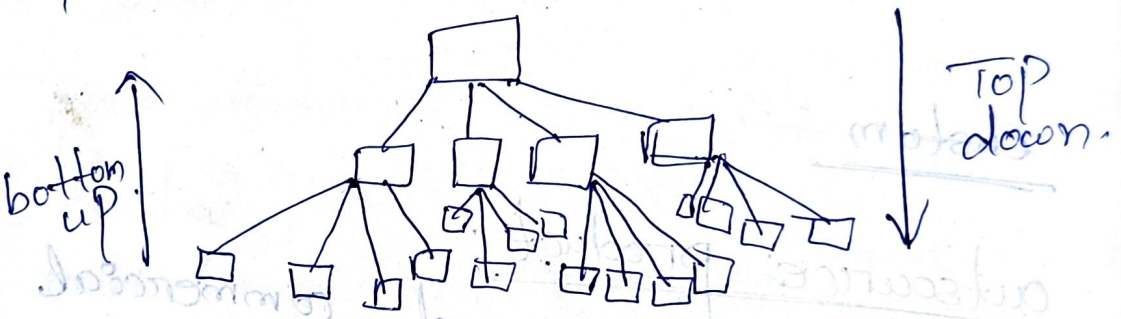
Principles by software engeneering to overcome. Human cognetive limitations:-

(1) Decomposetion.

Decomposetion tells decomposing the problems into many small interdependency

parts.

- The small parts are then taken up one by one and solved separately.
- The idea is each small part should be easy to handel, understand and solve.
- The full problem is solved when all parts are solved.



decomposition may be Topdown or bottom up.

Abstraction

Specifying a problem by omitting unnecessary details is known as principle of abstraction otherwise is known as modeling or model construction.

SE DT:- 18.07.16

- software engineering means to develope a software.
- 2 types application
  - standard based
  - web
- software engineering discusses systematic and cost effective techniques to software develop-ment.
- These techniques have resulted from innovations as well as lessons learnt from past mistakes.

Explain briefly disceplen of software engineering approach is science or art:-

Software engineering approach is neither science nor art. There are no. of issues so that no engineering approach comes under science or art.

- These issues are:-

(i) Engineering disceplens such as software engineering make heavy use of past experiences.

The past experiences has systematically arrange and applied wherever req.

But In science always exact solns are accepted by no. of proofs.

(ii) In engineering disciplines while designing a system several conflicting goals might have to be optimised. In such situations no unique soln may exists and several alternate solns may be purposed. Therefore to arrive at the final soln several iterations and backtracking may have to be performed. On the other hand in science only unique solns are possible.

(iii) In an engineering discipline the cost effectiveness approach is adopted and economic factors are addressed where as science normally doesn't concern itself with practical issues such as cost, maintenance, useability etc.

(iv) Engineering disciplines are based on quantitative principles but where as art based upon qualitative attributes.

The software engineering discipline means its evolution and its impact.

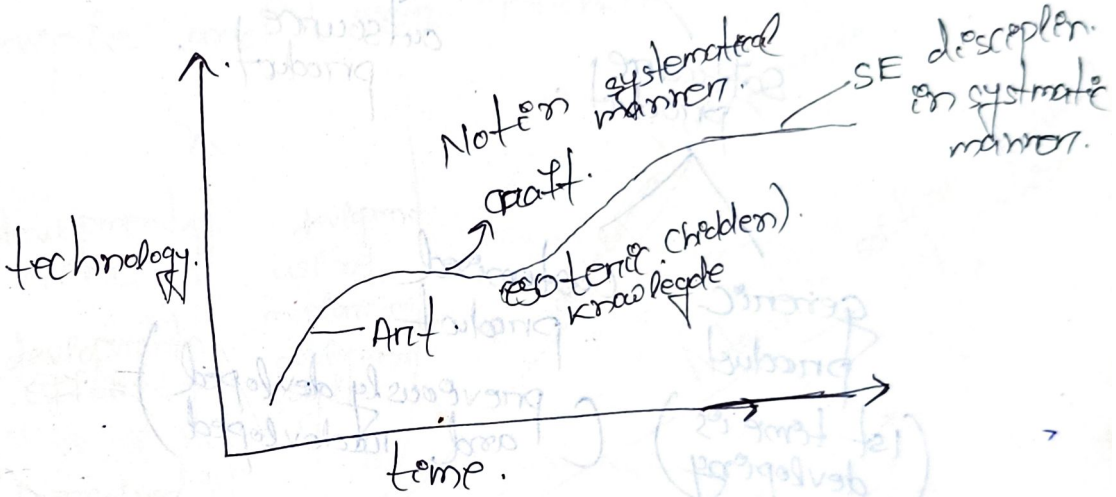
Evolution of an art into an engineering discipline.

- In the build and fix style normally a poor quality program is quickly developed without making any plan, design, specification.

The different imperfections that came subsequently makes whole users on testing app fined.

- There are many stories about programmers in the past who were like proficient and best and could create good programs based on some esoteric knowledge.

DT:- 19.07.16



- The story of the evolution of the software engineering discipline is not much different.

- In the early days there were good programmers and bad programmers. The good programmers knew certain principle they refused to share with bad programmers.

- over along with new research ideas, have systematically organised that forms discipline of the software engineering

Software crises

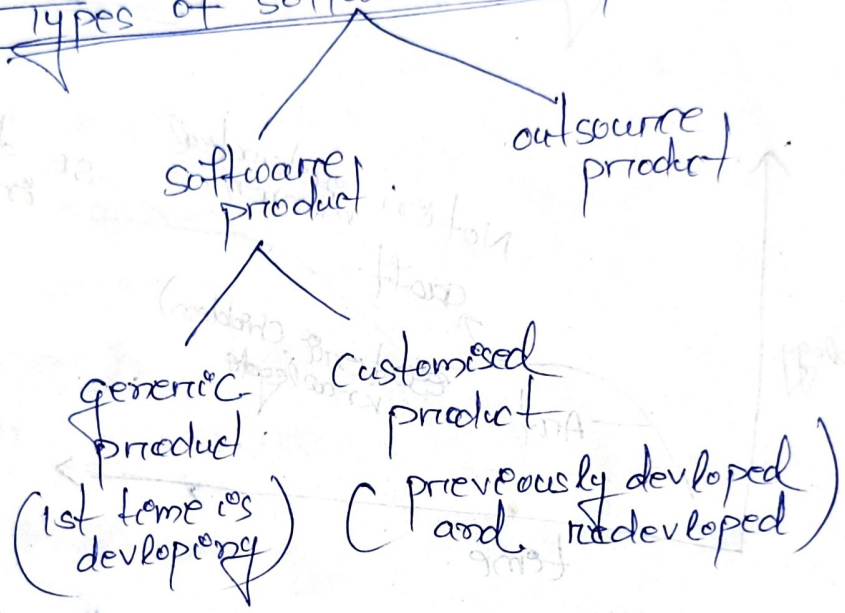
~~Before a day, p~~

Software development projects

Commerical product

It is used by huge no. of customers.

- Program vs project
- Types of software development projects



outsourced product

- Lack of freand, team.
- cost-effect.

Generic

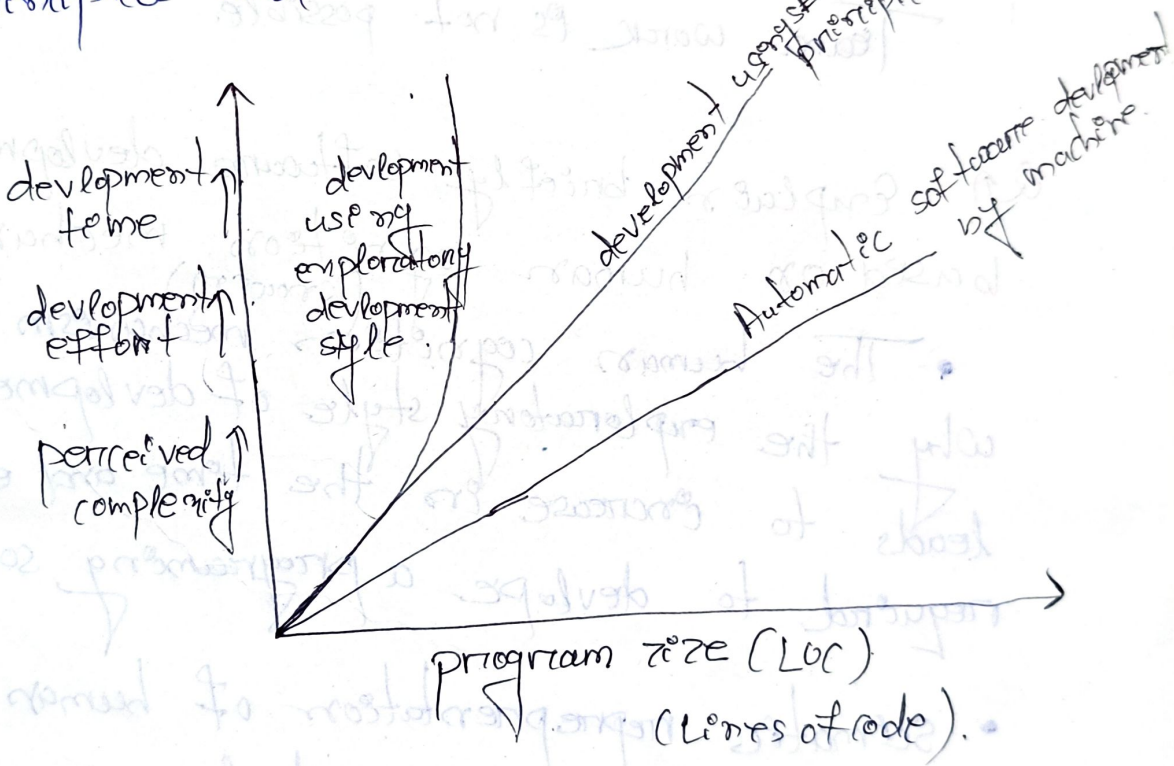
- To develop a generic product large no. of team members is reqd, where as to develop customise product less no. of team member are reqd.
- Generic product is used by huge no. of customer.

Ex of generic product (OS, DBMS, banking or railway software etc) where as customized software. p used by few customers.
(Ex:- LMS (library management system) of an institute)

DT:- 20.07.16

Exploratory style of software development

exploratory style means developing software using only human tricks no SE principle and machine knowledge is applied.



- for large problems it become impossible to develope the product using the exploratory style of development. A problem that requires covering one million lines of source code never be able to complete using exploratory style. of the amount time or the effort.
- involves, to solve it.
- invests.

when the development is carried out using SE principles then it is possible to solve a problem with effort i.e almost linear on product size.

If the problem could be written automatically by the machine then the increase in effort could be even closer to linear pace.

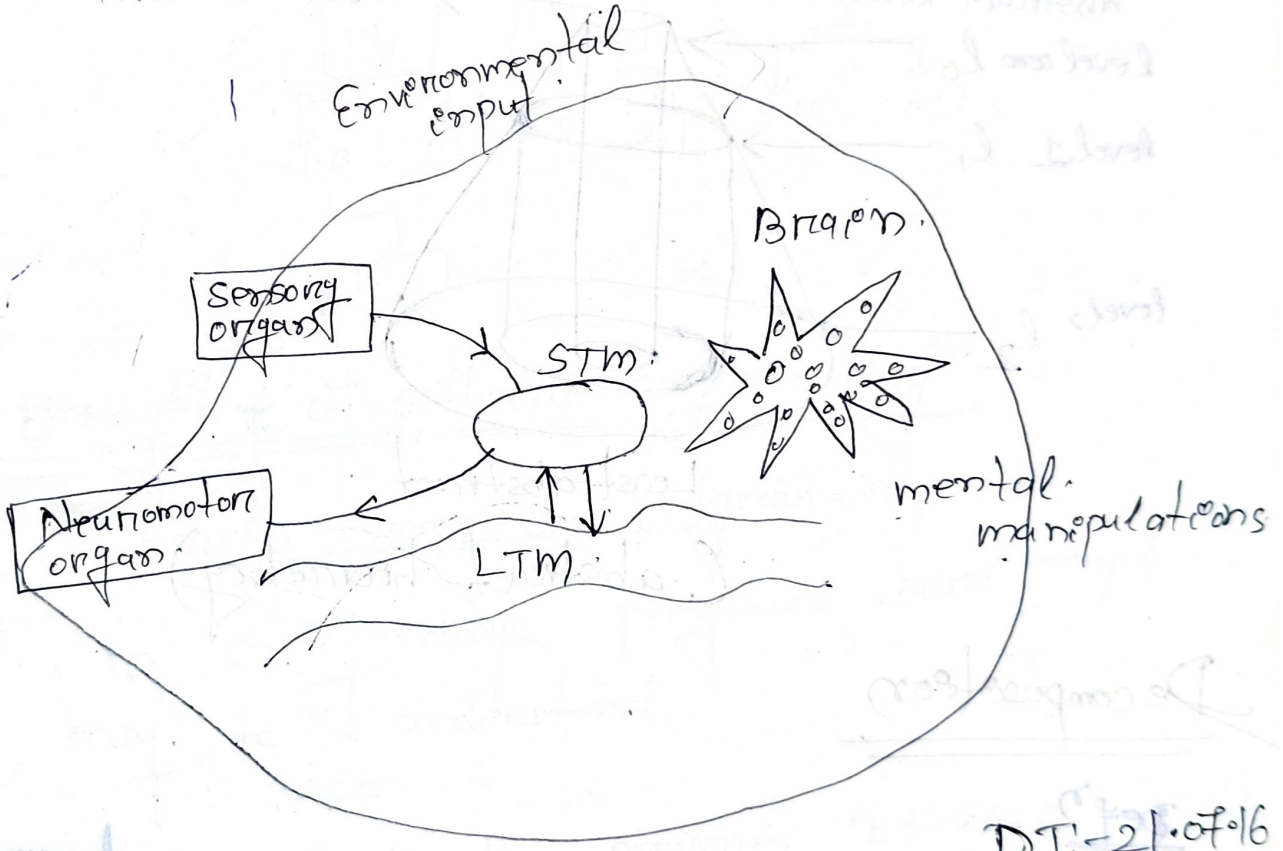
Q) what are the problem arises if we develope software using exploratory style.

Team work is not possible.

Q) Explain briefly software development based on human cognition mechanism (knowing)

- The human cognition mechanism explains why the exploratory style of development leads to increase in the time and effort required to develope a programming sol^n.
- schematic representation of human cognition is explained below.
- Normally human memory consists of two parts one is short term memory another is long term memory.
- short term memory is known as working memory
- Information store in STM for few sec

- STM get information from sensen organs. (Smell, touch, etc)
- It retrieve information from LTM. (yesdaga stored information)
- Brain also uses STM. STM is also used by mental manipulation unit.
- Information store in LTM in permanent manner.
- Item means collection of information from different places.



DT:-21.07.16

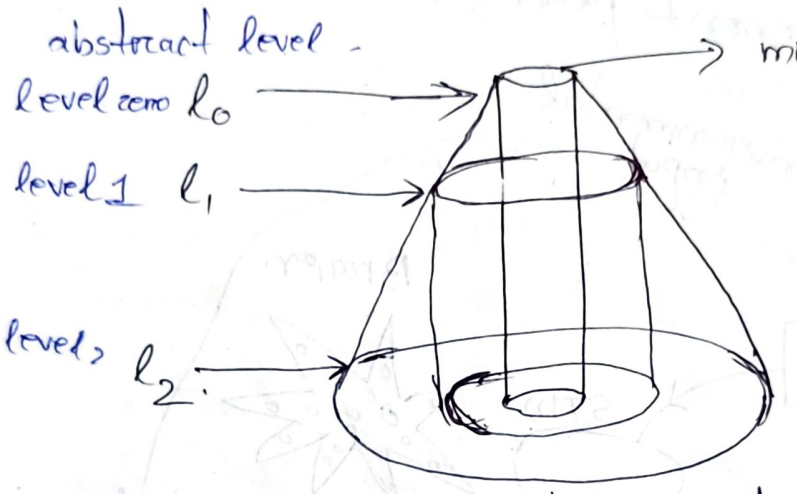
Too important principles are applied by software engineering principles to overcome. from human cognitive mechanism. These are abstraction and decomposition.

Abstraction

Simplifying a problem by omitting unnecessary

details is known as the principle of abstraction.

- To understand a complete problem abstraction or methodology is used. It focused only necessary details.
- Depending upon complicated problem there are various levels of abstraction.
- The top most level gives less information where as the bottom level explains detail.



(abstract heirarchy).

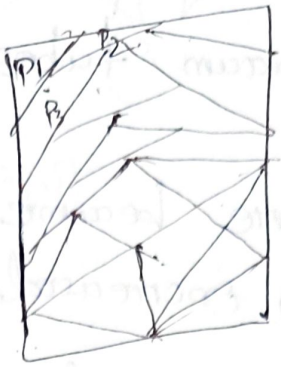
Decomposition

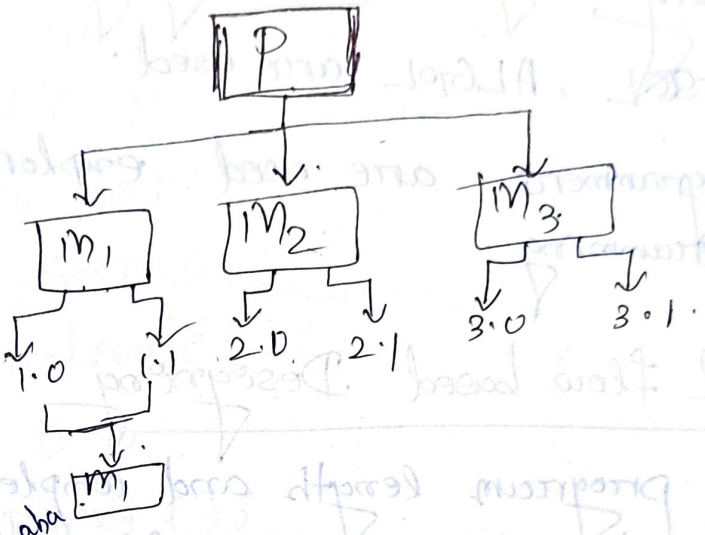
Def?

The decomposition principle says decomposing the problem into many small independent part. The small part are taken one by one and solved separately.

The idea is that each small part could be easy to grasp (understand) and understand and can be easily solved.

The full problem is solved when all the parts are solved





Emergence of software engineering

→ Early computer programming:-

- In early days programs were simple. easy to understand.
- Length of the program 100 lines of code.
- Programmers are solved prob. using their own. tricky knowledge.
- Basically assembly languages are used
- At the end we can say early computer programming used exploratory programming style.

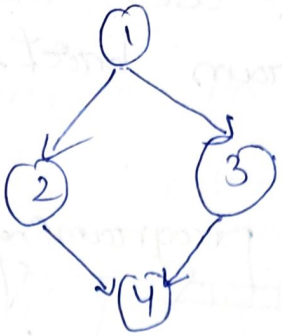
→ High level language programming

- Instead of vacuum tubes transisters are used
- programs are became complicated. Lines of code (LOC) encreased from 100 to 1000.
- Many high level languages like COBOL, PASCAL, ALGOL are used.
- programmers are used exploratory style programming.

Control flow based Desegning

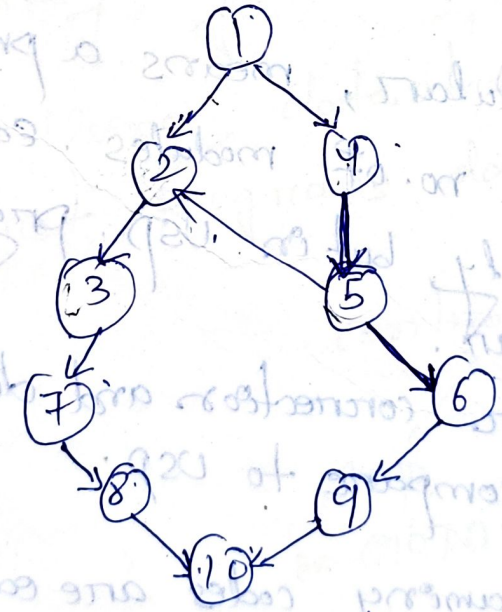
- As program length and complecety encreases then it become very difficult for the program ersto. comete program on cost effective manner. maintainance of programming code also become difficult.
- To overcome this problems programers intro-duced programs control flow str.
- A programs control flow str. endicates the sequence. on which the programs constructors are executed.

```
1. if().
2. call function();
3. else print(..);
4. end
```




```
1. if()
2. 100: issue=T;
3. GO TO 110;
4. else if( ).
5. GO TO 100;
6. else GOTO 120;
7. 110: activate.();
8. GOTO 130;
9. 120: print();
10. 130: end;
```

(GOTO is a jumping stmt
↓
it is very complecated to understand, so it's not written).



- The scientists Dijkstra explained "GOTO statements considered harmful, but in early days to jump from one place to another place only GOTO stmt introduce so though GOTO statement are harmful but at that

some programmers use that construction to explain program briefly.

DT:-22.07.1

Structure programming

In structured programming (SP)

(1) Assignment
(2) selection (if else statement)
(3) iteration (loop statement).

where as in constructed programming (USP) ing statements are there

- In SP there is a good control structu where as in USP there is no good contro structure.
- A SP is modular, means a problem decomposed into no. of modules. Each mod solved separately, but in USP programs are not modular.
- In SP error correction and detection is easier as compare to USP.
- In SP programming codes are easily readable, maintenance is easier, less effort, cost effective as compared to USP

• In sp there is one start and exit point where as in usp it is not possible.

Data structure oriented design.

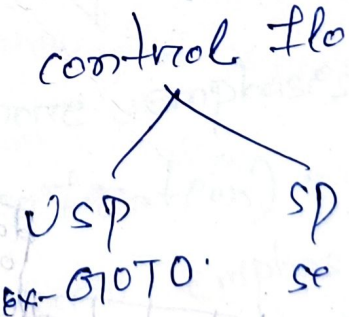
• In early days (1970) VLSI circuits introduced. (very large scale integrated ckt) IC

• Using VLSI computers become faster, problems solved easily.

• complicated software products are solved using this ckt. still programers faceing to solve software which are very lenthy

• programers planned give more emphasis on designing data structure of a problem than designing of control str^r of a problem

• Using data structure oriented technique first programs data structures are designed



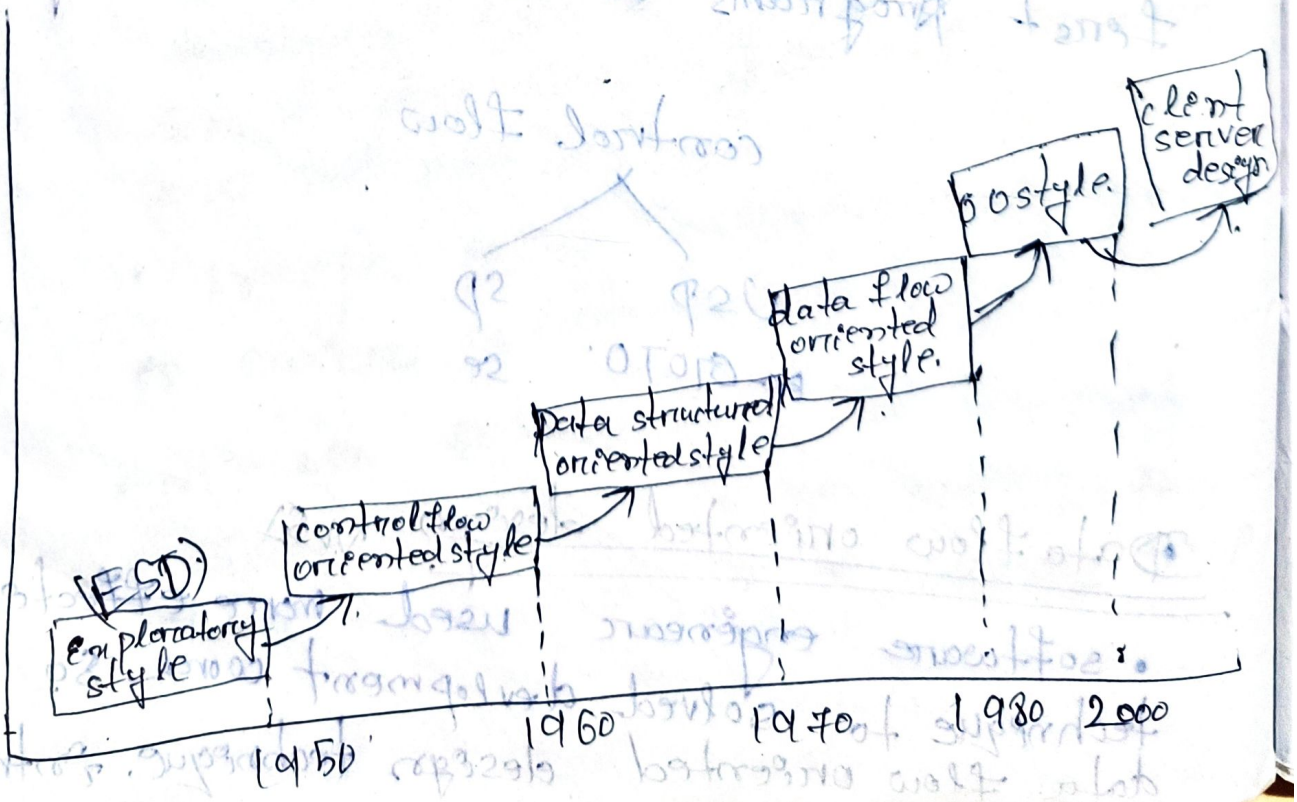
Data flow oriented design VLSI

• software engineers used more effective technique to solved development cocrk. So data flow oriented design technique, introd

- The data flow oriented technique emphasized the major data item handled by a system must first be identified, and then the processing required on these data items to produce the desired outputs should be determined.

Object oriented designing (OOD)

- Data flow oriented design evolved into object oriented design.
- The main work is done by object for a particular problem.
- Essential features of OOD are
  - Data hedding
  - Data abstraction
  - Inheritance (reuseability of the code).



Q. Explain briefly software development using exploratory style vs software engineering principle and practices.

IMP

- ESD programming is based upon error correction where as SE principle based on error prevention.

In ESD after product is ready error detected and correct where as in SE no of stages are there for software development work. Each stage is reviewed for errors there corrected and movement to the next stage.

- In ESD synonym of software development is writing program coding where as in the coding is a small part, there activities like designing testing are there.

- In SE more emphases on SRS (software requirement specification). DT:- 23.07.16

- Nowadays more emphasis on design activities. so different design technologies used to develope software.

- periodical reviews carried out after completion of each phase so there as soon as possible error can be detected and

connected which is known as phase containment errors.

- Now a days different ~~testing~~ techniques are used for testing activities ~~to~~ to test software by the test cases are designed ~~so that~~ customer as well as developer. ~~develope~~ DT-22-01-16

- Nowadays there is a better visibility of the product through various development activities By visibility we mean production of good quality product and peer reviewed documents at the end of every software development activity.

- Now a days to develope a software. different project planning made like total effort, total time, resources are reqd, resource rescheduling etc.

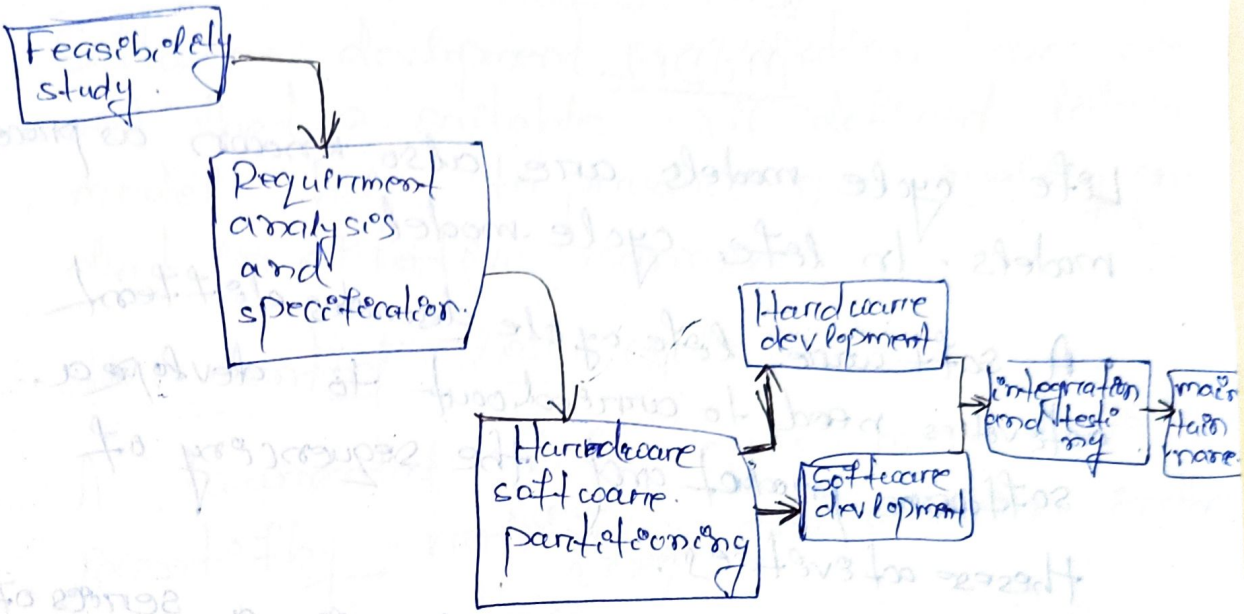
- Now a days different measurements are used to develope software.

computer system engineering

system software:- used to control hardware
ex:- operating system.

apps running on operating system - application software.

project management



SOFTWARE LIFE CYCLE MODEL

Life cycle models are also known as process models. In life cycle model.

- A software life cycle describes different activities need to carried out to develop a software product and the sequency of these activities.
- A software life cycle is a series of stages that a software product undergoes during its life time.
- Every software product starts with a request for the product by the customer. this is called product conception.
- A life cycle model is a descriptive and diagrametic representation of the software.
- A life cycle model graphically represents the different phases of a life cycle and they are ordering maintained by textual description.
- A life cycle model maps the different activities perform on a software product from its conception to retairment onto a set of life cycle phases.

Why use life cycle model. DT:- 25-07-16

Software development organisation have realized that a suitable well defined lifecycle model helps to produce good quality product in effective manner.

Documented life cycle model

A documented life cycle model helps to identify redundancy and omission in the development process.

phase entry and exit criteria

- In a life cycle model it is very important to fix entry point and exit point.
- once a phase completes then next phase starts.

Different phases of Classical waterfall model

Practically classical waterfall model not used by any software houses. though it is not used current life cycle models are based upon this model so it is essential to have a clear understanding working principle of classical waterfall model.

In the following diagram different phase of classical waterfall model explained.

waterfall

Feasibility study

Requirement analysis and specification

Design

coding and unit testing

integration and system testing

maintenance

Drawback

we can't go back to correct any error

Feasibility study

- The main goal of feasibility study actually is, whether it would be physically means financially and technically feasible to develope the product.
- It involves. ~~Persons~~ fact's. like gather all the datas reqd to develope the system, processing of data and output from the system.
- Generally, this phase gives an abstract idea of the problem.
- Formulation of different strategies to solve the problem.
- Evaluation of diffent strategies. Deffent

- strategies are analysed and among the strategies best one is selected to develope the production.

- Because of this feasibility study is the best one of the important phase as compared to any other phase.
- Also we can say it is a decision making phase. All the senior persons of the organisation participated in this phase.

IMP **Requirment analysis and specification**

The main activity of this phase is to understand the exact req of the customer and document them properly. So this phase is devided into two parts.

(a) Requirement analysis gathering and specification.

(a) **Req gathering and analysis**

- 1st gather requirements and analyse the gathered requirements. The main goal is to collect information from customer to develope the product. After gathering information analysis activity started.

Requirments specifications

All the customer requirments gathered are documeted in a document which is known as SRS document (software requirment and specification

SRS document mainly consists of 3 requirement :—

(i) FR
(ii) NFR
(iii) goal for implementation

- Functional requirement
  (i) input data.
  (ii) processing data.
  (iii) output data.

- Non functional req mainly consists of performance standard to be followed.
- Generally SRS document is written using end-user terminology.
- So that customer easily understandable using SRS document. After SRS document is reviewed it is aproved by the customer.
- It is a contract between developer and customer. SRS document also consists of other activities like user manual, system plan, design etc.

Design phase

The Goal of this phase is to transform SRS document contents into a structure. i.e suitable to implement program using any language. technically we can say design phase is known as software architecture phase.

Design phase is possible by two approach (i) Traditional designing approach (TDA)
(ii) object oriented approach

(i) TDA

- TDA followed by all software houses
- This approach is mainly based upon dataflow oriented design that consists of two activities.

→ structure analysis from SRS document
when it completes structure design starts.

→ In structure analysis no. of functions supported by the system examined data flows from one fun to another fun analysed. Decomposition of functions takes place in this phase mainly structure analysis activity represented by dataflow diagram (DFD).

DT:- 26.06.16

ex: fun1 (data1),

```
{
return data1;
}
```

→ fun2 (data1);

- structure des

After structure analysis software design was starts there are two activities.

(i) HLD (High leveled design or known as software architecture)

(ii) LLD (detailed design or low level design)

- High level design specifies total no. of modules of the developed system, interaction among the modules in low level design each individual module design like data structure of a module, algorithm followed by module.

coding and unitesting

The purpose of this phase transform design into source code the end product of this phase consist of no. of programming modules which are implemented using different programming languages. because of this reason this phase is known as implementation phase.

Different organisation used different standards to implement module.

Unitesting :-

After source code unitesting ~~code~~ work starts. each module individually tested on

isolation to other modules. The main cause is individual module errors detected and corrected when a module is under unit test no other module interact with that module.

Integration and system testing

Integration is not possible by single shot it consists of no. of steps one module is integrated with another module dependency upon planned system. After integration system testing activities start. System testing classified into 3 catagories.

(1) α-testing

Testing is done by developer

(2) β testing

Testing is done by a set of friendly customers.

(3) acceptance testing

Testing is done by customers.

Generally system testing is carried out on a planned manner according to system test planned document. After testing is over. The summarize report condition a document known as test report

Maintenance phase

Maintenance requires more effort to develop the software

There are 3 types of maintenance.

(1) corrective maintenance

This type of maintenance envolves corrected errors that are not detected at the time of software development. This type of maintenance envoves correcting error that are not detected at the software develop

perfective maintenance

It envolves improvement of implementation of the system.

Adaptive maintenance

This envoles port software code different platforms.

Disadvange waterfall model

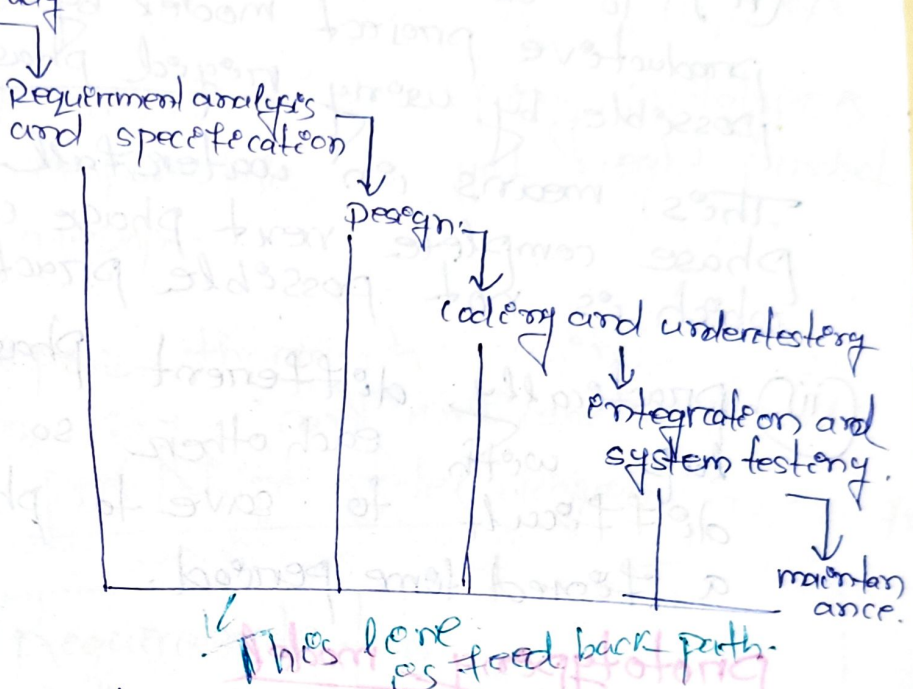
The classical waterfall model assumes that no error is ever commited by the engineers during any of the lifecycle phase and there fore leaps no scopes for errors correction.

This model assumes all the customer requirment defone correctly and complete at the beginning of project.

and on the bases of that development work starts but practically customer req^men always changes which is not supported by the module.

Iterative waterfall model

DT: 28/07/16



(i) Interative water fall model carried out by making necessary changes on classical waterfall model.

(ii) It is used for developing software project.

(iii) In this model a feed back path introduce, It connect every phase to it precedency..... In a phase if error committed and error detected in the next phase then using this path rework can be done preceding phase.

(iv) But there is no feedback path to connect to feasibility study phase this means no rework is possible in this phase.

Disadvantages of Iterative waterfall model

(i) practically different types of risk arises when a software project is developed using waterfall model satisfactory working is not possible.

(ii) To achieve better efficiency and higher productive project model by the developers not possible by using rigid phase sequence model.

This means in waterfall model one phase complete next phase work starts, which is not possible practically.

(iii) practically different phase cannot over lap with each other so it is very difficult to save to phase completion a fixed time period.

Prototyping model

Proto type model is a toy implementation of a system.

A prototype model uses different short cut to implementation.

Advantages

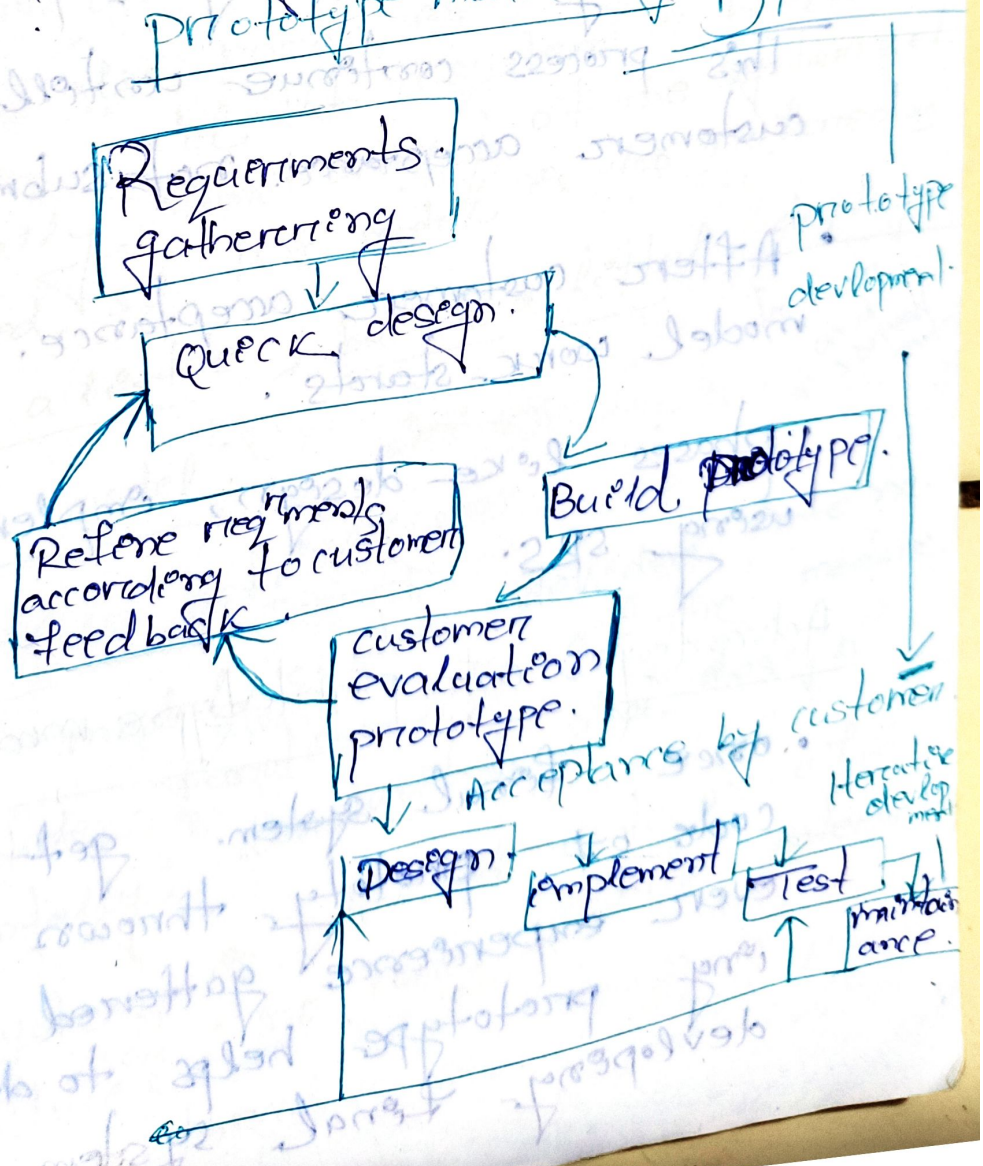
- This model is used full to developed GUI (Graphical user interface) of a system. It is generally used to implement message dialogue box, input data formats, reports according to costmer requirement.
- Another reason when development team as lot of technical issues to developed a project.

Ex :-
Suppose to implement a compelen a common language and a developerz team before never develop compilerz to any particular language. Using prototype model they implement a small compelen for a small language, then the developerz analysesed and cohen they are master they developed final product.

The third reason to developing prototypes that it is impossible to get right product on the 1st attempt.

When developer thiorough using this model then they can implement final. product.

prototype model diagram



1st phase of prototype model to control various risk and second phase implement using alternative water fall model

How prototype model work

- Initially all the customer requirements gather then quick design starts to build prototype of a system.
- After prototype is form, customer evaluate
- Dependency upon customer feedback if regd customer requirement changes are accordingly to that again prototype build. This process continue until and unless customer acceptance not submitted.
- After customer acceptance. water fall model work starts.
- phases like design, implement develop using SRS.

Advantages of prototype model

- once final system get ready the code of prototy throwon away. How ever experience gathered from develop ing prototype helps to deals for developing final system product.

- The total cost to devlope a system using prototype model is less than to devlope a system using water fall model.
- By constructing prototype customer requirements properly implemented and technical essues get resolved or get removed.

Evolutionary model

- Evolutionary model known as successive versions of the model.
- It is also known as incremental model
- In this model 1st a system is built then gradually functionality of the system get improved so this model is also refered as design a little, build a little, deploy a little model.

feedback and improved out.

Life cycle activities

In this model system devided into no. of modules, functionality of the module incrementally constructed and design

1st system is devided into no. of core modules: no core module interact with each other

- Again non core modules developes, non core modules interact with core modules.

- Each version of the product is fully functioning and performing more than previous phases.

Advantages

- In this model users get a chance to experiment with partially developed system before the complete version is released.

- This model helps to implement model according to customer requirements of changes dynamically. As a result when final product released then change of request is less.

Disadvantages

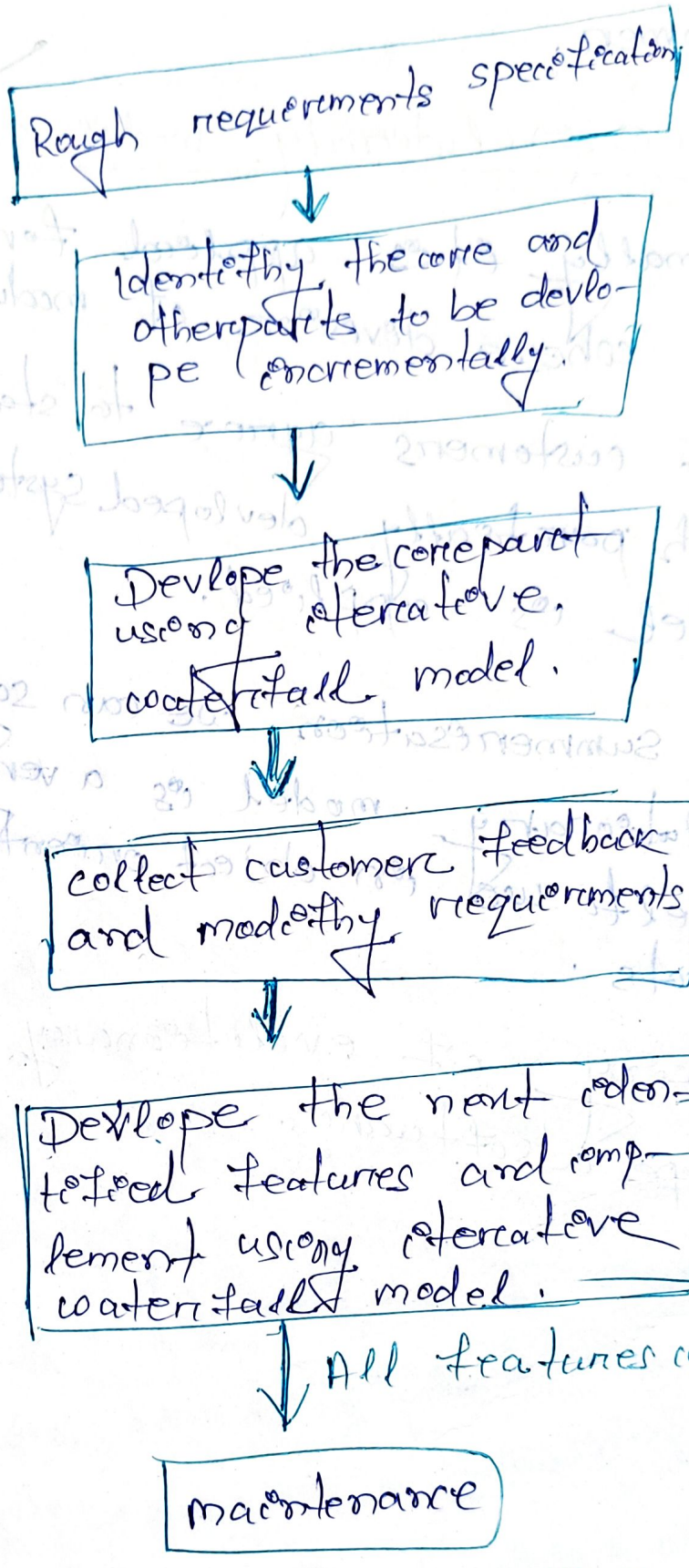
the critical think is how practical problem devide into several versions so that it could be acceptable by the

customer.

Where evolutionary model can apply

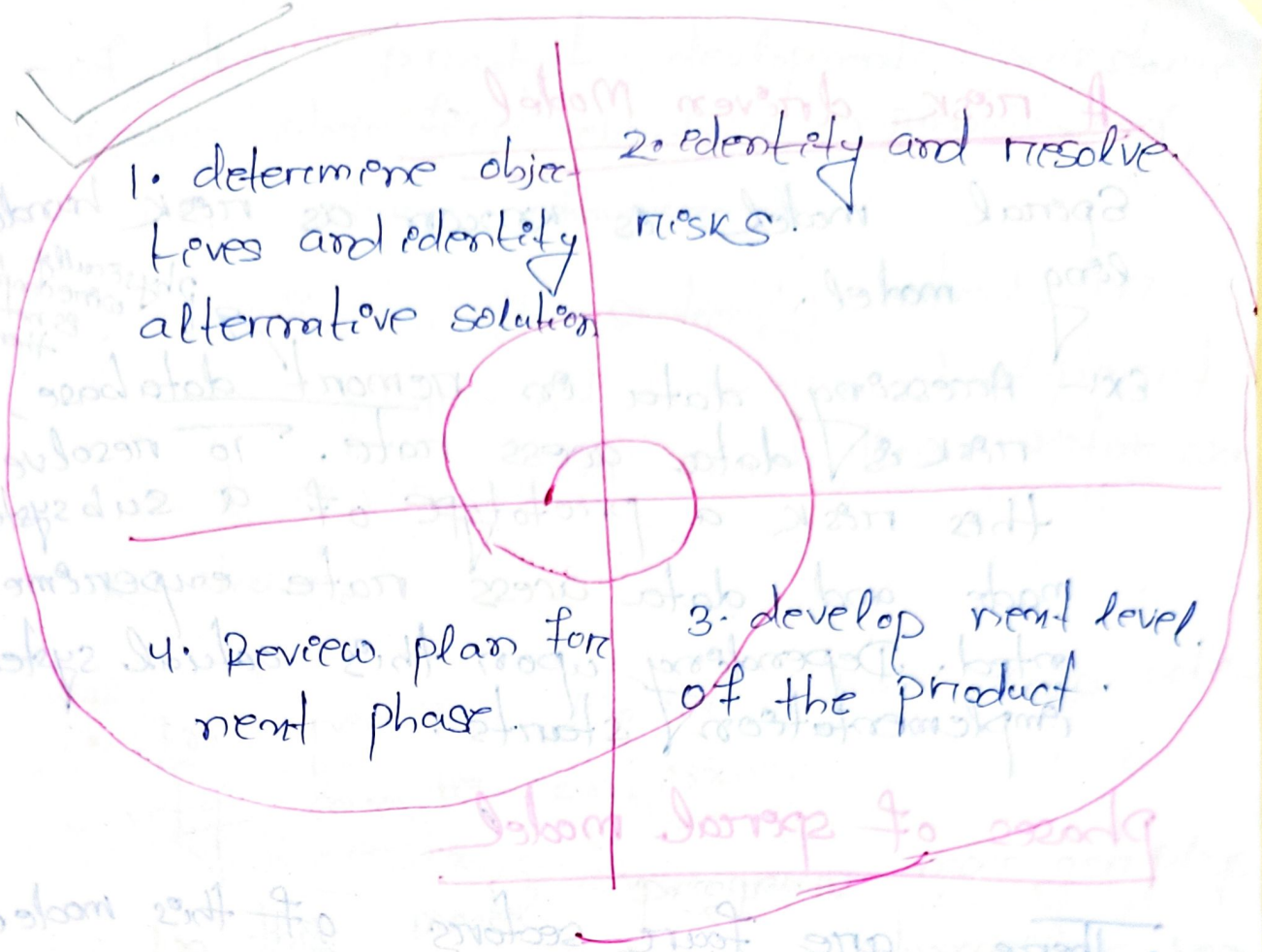
- normally it is applied for large products where division of modules is easier.
- If customers agree to start work with partially developed system then this model is applied.
- In summerisation we can say the evolutionary model is a very natural model to use on object oriented development projects.

working of evolutionary model to develope a software.



DT:- 01.08.16

Spiral ~~boosting~~ model



Spiral model represents no. of loops the diagramatic representation sometimes spiral.

Each loop represents one phase.

No. of loops are not fixed, it's uncertain.

over each loop one or more features of the product are elaborated and analysed and the risk at that point of time are identified and are resolved through prototype model. Based on this identified features are implemented

A risk driven Model.

Spiral model is known as risk handling model.

Ex:- Accessing data in remort database (physically wire conn is not there) risk is data acess rate. To resolve this risk a prototype of a subsystem made and data acess rate experimented. Depending upon this actual system implementation starts.

Phases of spiral model

- There are four sectors of this model
- In the 1st sector objectives of the phase identified.
- objective investigated, Elaborated, Analysed
- The risk of the phase identified.
- Depending upon risk no. of alternative solutions identified.
- In the second sector all the alternative solutions experimented, among the soln best solns selected for implementation. The solutions are experimented using prototype model.
- At the 3rd sector next level

of the product development work starts means identified objectives implementation work starts. ~~to the f~~

- In the fourth section planned for next phase of software development and review work of last three section work starts.

- In sperial Model level loop activity can be continue.

- In this model program manager plays a role because dynamically because program managers dynamically determine no. of phases of the problem.

Advantages and disadvantages

- As compared to any other model it is a complicated model
- It is generally used for complicated problems. To apply this model generally knowledgeable staff required.
- It is generally used for certain categories of project.
- For projects having many unknown risk that might show up as the development process the sperial model would be the most appropriate. Develop

model to follow up.

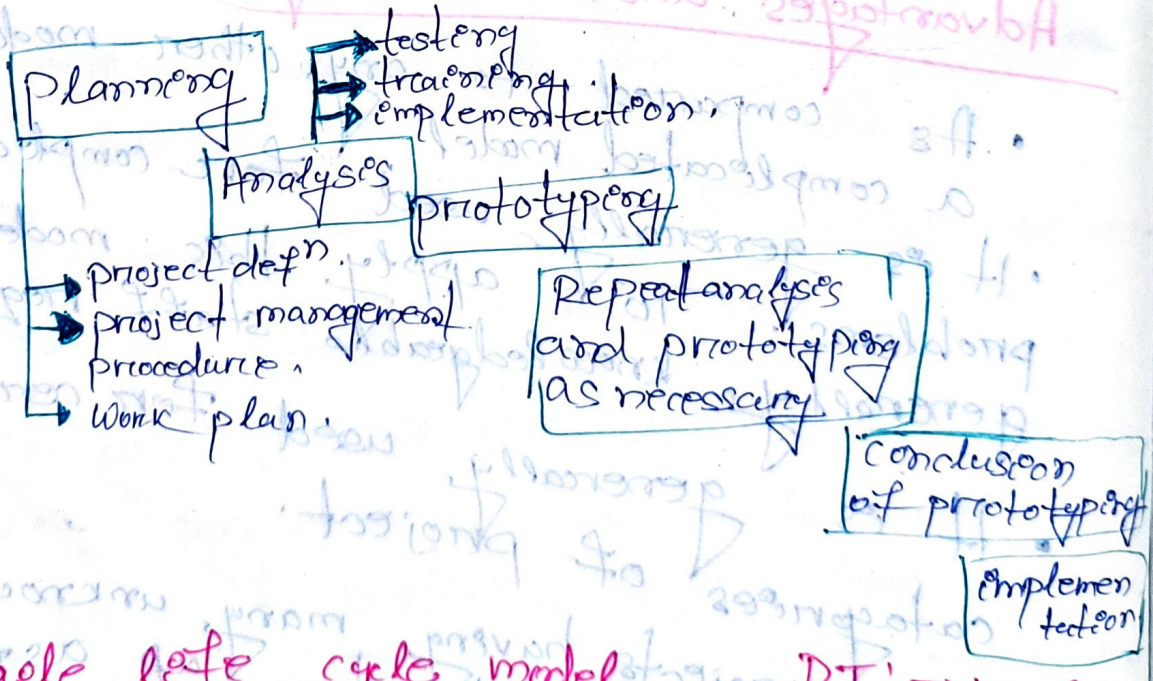
Q) Each spiral model known as meta model?

- Spiral Model is a metamodel. To experiment alternate solution prototype model is used to implement each phase work wise fall model is used..

- Each loop represents one generation each preceeding loop represents part of development of the system as compare to successive loop. So it satisfied proportion of evolutionary model.

DT:- 02.08.16

Rapid application development model (RAD)



Agile life cycle model

DT:- 10.08.16

- Agile lifecycle model is a type of incremental model.

- software is developed in incremental rapid cycles.
- Each release is thoroughly tested to ensure software quality is maintained.
- It is generally used for critical applications.
- Extreme programming is one of the current Agile development cycle.

advantages

customer satisfaction by rapid, continuous, software delivery.

More emphasis on interaction bet^n customer, developer than process and tools.

face to face interaction is more on this life cycle model.

- continuous interaction for technical excelence and good designing if late changes then also applicable in this model.

Disadvantages

- Less work on designing and documentation.
- only senior programmers are working on this life cycle model.
- software is developed rapidly, so if customer representation about requirement then a good project development

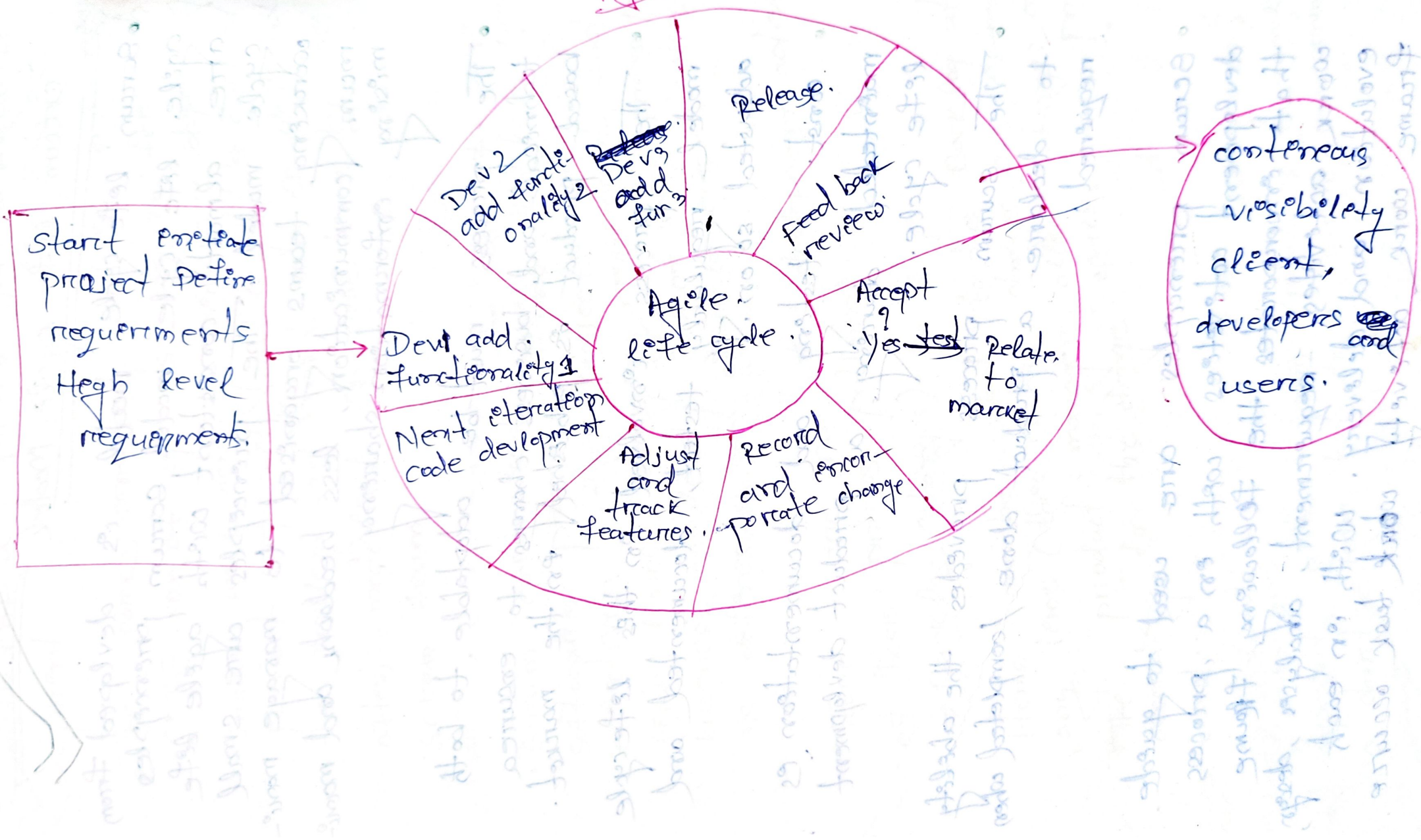
Les not possible

When to use Agile life cycle model:-

- When new changes are needed to be implemented then on this life cycle model are used.
- For new changes developers work is less as compared to any other life cycle model.
- This life cycle model requires very less planning as compare to any other life cycle model like water fall model.
- If rapidly want to develope a software without much more effort on designing, documentation, planning, this life cycle model is useful.
- If customers requirement rapidly changes during software development life period then also this model is useful.

Diagram

Agile lifecycle model



DT:-19.03.16

Scrum life cycle Model

- Scrum life cycle model is developed from agile life cycle model scrum principles are almost consistent with agile life cycle model. The principles are small working teams organised to manage maximum communication, less headache and maximising information sharing.
- The process must be adoptable to both technical and business changes to ensure a based product is published on the market.
- The process involved in this life cycle model regularly tested, documented and adjusted so on.
- const testing and documentation is maintained during a product development life cycle.
- The scrum process provides the ability to declare a product done / completed when required.
- Scrum principles are used to guide development activities with in a process that incorporates the following frame work activities requirement, analysis, design evolution and delivery. With in each frame work activity work task occurs.

coeth in a process pattern called a sprint.

- The coorck conducted coeth in a sprint. (The no. of sprints requere for each frame coork actevety coell vary depending on product complexety (and length) es adopted to the problem and hand and es defined and often modefied en reak teme by the scrum team.

- Scrum emphasised the use of a set of softcoare process patterns that have proven effectiven., for projects coeth tight teme lenes, changeng requerements and business complexety. Each of these process patter ns defines a set of development actevetees.

There are no. of development actevetees of these process patterns.

→ Backlog (changes)

Items can be added to the Backlog at any teme (this es how changes are introduced) The product manager allocs changes and according to the change coorcks are prio- tesed.

→ sprints

sprint allocs team members to coorck in a short team but en a stable enveronment.

→ Scrum meetings

There are three key questions asked and answered by all team members.

(1) what did you do since the last team meeting.
(2) what obstacles are you encountering.
(3) what do you plan for the next team meeting.

A team leader called a scrum master leads the meeting and collect feedback from each person.

→ demos

delever the software increment to the customer so that functionality that has been implemented can be demonstated and evaluated by the customer.

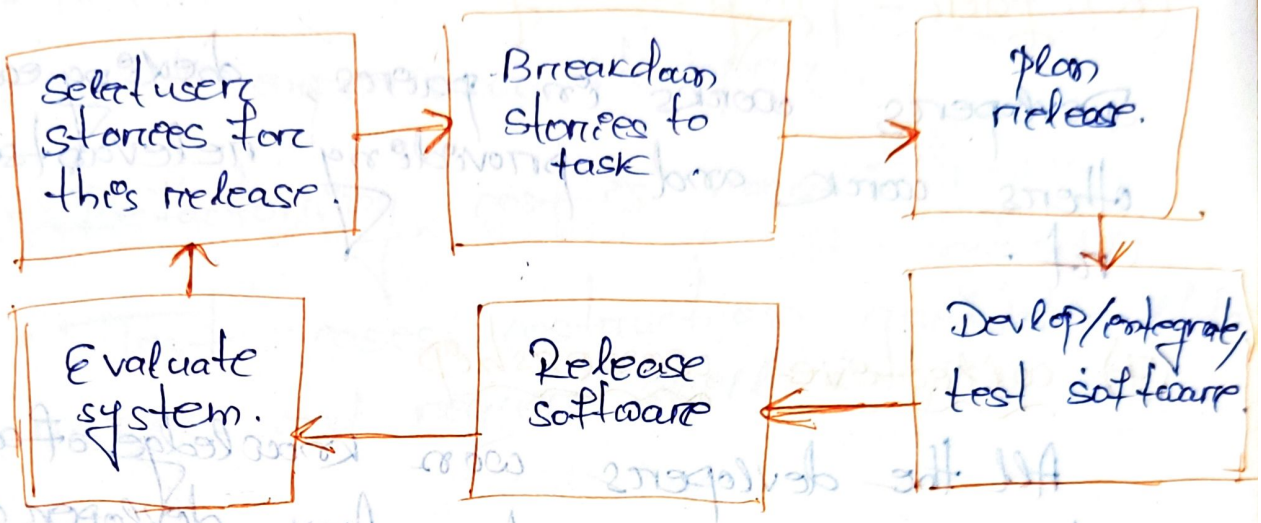
DT:- 22.08.16

Extream programming

Extreme programming principle

- Incremental development is supported through small frequent system releses.
- Customer full involvement
- people not process through pair programming collective coonership and a process that avoids long coorking hours.

- change supported through regular system releases.
- Maintaining simplicity through const refactoring of programming code. (makeeasy)



Extreme programming practices:

(1) incremental planning
collect customers requirements and devide into no. of small parts.

(2) small releases
Releasing of the system are frequent

(3) simple design
More designing pattern is carried out for fulfilling of customer requirements

(4) Test first development
An unit testing is carried out to test

individual parts.

(5) Refactoring

Using refactoring programmers make code as simple as possible.

(6) Pair - programming

Developers works on pairs, checking each others work and providing relevant support.

(7) collective ownership

All the developers own knowledge of all the programming codes - Any developer can change anything.

(8) continuous integration

As soon as on a task is complete it is integrated into the full system. or whole system.

(9) On-site customer

During extreme programming representative from end- user side always available.

Advantages ~~used for small~~

- used for small.
- more importance on final product.
- The technique is iterative

Disadvantage

- Not useful for large project.
- Need experience developer.
- Refactoring cost is more
- Test cases construction are difficult to manage and require skill developer.

Requirement Analyses AND Specification

chapter-2 DT:- 29.08.16

The main goal of requirement analysis and specification is to clearly understand customer requirement and to systematically organize requirement into a specification document which is known as SRS document.

SRS document is the final outcome of requirement analysis and specification phase.

The requirement analysis and specification phase consists of two activities.

(i) Requirement gathering and analysis.

(ii) Requirement specification.

(i) Requirement gathering and analysis

The main goal is to collect all information that are required to develope a system. It looks simple but it is a complicated work so to overcome this problem there are different ways to collect information

→ studying the existing document

Before going to customer site first study all the existing document, the system to be developed.

→ Interview

There are different types of users of a soft coare different categories of users requir. different features of the softcare. So the conduct enterview for different catagories of usere for developing the softwa- re.

→ Task analyses.

The user generally veew a Softcare as a black box that provides no. of ser- vices. Each service is known as a task. To execute a task different speps requir for debugging.

Example: Issue a book from the library.

To debug this task the steps are:-

(i) Authonticate user

(ii) cheak book limitation

(iii) cheak wheather the book is reserve or not

→ Scenario Analysis.

To execute a task there are different types of scenario.

Ex:- Book issue from library.

scenarios are.

(i) user is not authonticated.

(ii) Book is reserved and can't be issued.

(iii) Maxm no. of Book is reached.

→ Form analysis

There are different forms analysed to input data to a system and output data from a system.

(ii) Requirement analysis

The main purpose of the requirement analysis activity is to analyse the collected information to obtain a clear understanding of the product to be developed with a view to remove all ambiguities from the mind of customer of the problem.

There are 3 main activities of requirement analysis phase.

→ ambiguity.
→ inconsistencies.
→ incompleteness.

ambiguity

if a customer not given clear requirements to develope a system then there will be different types of ambiguities for developing a system.

Ex:- The student who will get less mark he will be fail.

Ex:- Suppose from the examination section gate a notice that if a student obtained less mark in a subject then the declare as fail here ambiguity is

there is no crystal information for less mark.

Inconsistencies

Two requirements are said to be inconsistencies if both are contradict to each other.

ex:- From the examination side a notice came that if a student secure less mark then there will be repeat in a semister. From the office side a clerck said there is no provision of repeat in a semister.

Incompleteness

Means set of requirements are skiped from customers side which is visualised from developers side at the end for developing a software.

SRS (Software requirement specification)

DT:- 14.09.16

The important catagories of users of the SRS document are as follows
(i) user, customer and marketing persons.
(ii) software developers.
(iii) Test engineers.
(iv) project managers.
(v) Maintainance engineers.

(vi) **Characteristics of a good SRS document:-**
There are different characteristics of a good SRS document. The characteristics

are → concise.

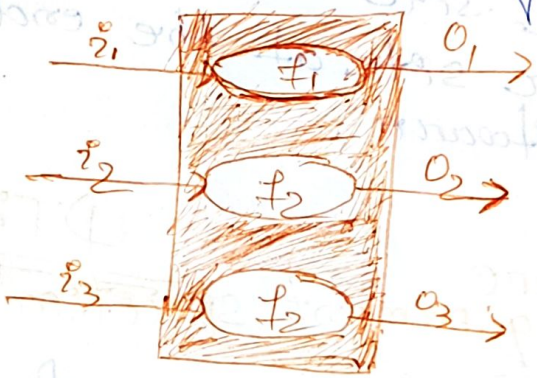
- if a SRS document is unambiguous & complete, then it is a good SRS document.

→ structured

It should be a well structured so that user can be easily understand and modify

→ Black box view

A SRS document should specify no. of input data, functionality of the system and outputs. A SRS document must specify high level functionality of the system.



→ Traceable

It shold be possible to trace a specific requirement to the design elements that implement and vice versa.

→ verifiable or verification is possible

What ever data is available in SRS document for verification phase that data can be again reused here for designing test case.

are → concise.

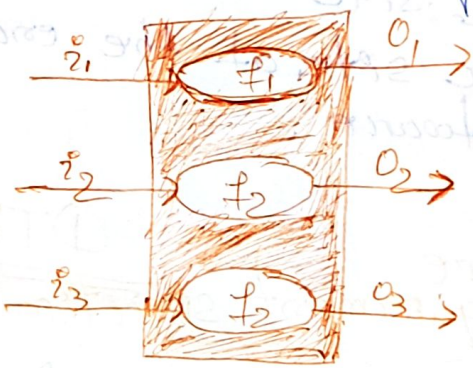
- if a SRS document is unambiguous & complete, then it is a good SRS document.

→ structured

It should be well structured so that user can be easily understand and modify.

→ Black box view

A SRS document should specify no. of input data, functionality of the system and outputs. A SRS document must specify high level functionality of the system.



→ Traceable

It should be possible to trace a specific requirement to the design elements that implement and vice versa.

→ verifiable or verification is possible

What ever data is available in SRS document for verification phase that data can be again reused here for designing test case.

→ Response for undesired events

SRS document helps for exceptional conditions. this means it helps to overcome from different types of errors like system errors.

Bad SRS documents:—

→ Noise

The term noise refers to presence of material not directly relevant to software development.

→ Wishful thinking

Wishful thinking means some information there which are not possible to implement.

→ Forward references

Forward references should responsible to reduce details specification of any functionality of the system. (in exam write unnecessary)

→ over specification

over specification occures when we try to address the how to aspects the SRS documents.

Important Categories of customer requirements:—

A SRS document should clearly document following aspect of the system

(i) functional requirements.

(ii) Non functional requirements.

(iii) Goals of implementation.

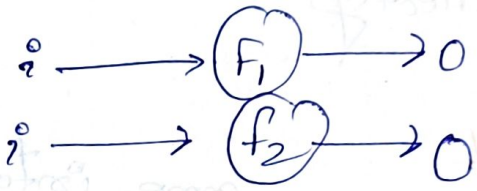
DT:-16.09.16

Functional requirement:-

Functional requirement discuss functionalities required by the users from the system.

Functional requirement same as mathematical function of a problem.

$$f : I \rightarrow O$$



Non functional requirement:-

Non functional requirement deals with characteristics of the system which are not expressed as the functions of the system examples are:-

- System use.
- maintenance
- user and computer interaction
- Reuse of a system

Goals of implementation:-

A goal on contrast to the functional and non functional requirements is not checked by the customer for conformance at the time of acceptance testings.

The goal of implementation may document issues such as reuseability of the

system, revesion to the system functionalities that may required in the future, new devices supported in the future etc.

Details of functional requirements:

A high level function is one using which the user can get some useful piece of work done.

for f_1 : issue a book — high level function.

sub fun { f1.1, f1.2

EX

issue a book from library consists of no. of sub functions like cheaking of library card, book availability, users a student or faculty or any other staffs.

How to document the functional requirements

With draw cash from atm.

R.1: withdraw cash.

Description:

The withdraw cash function first determine the type of account that the user has and the account no. from which the users wishes or wants to withdraw cash.

It cheaks the balance to determine whether the requested amount is available in the account. If balance is available

then its output is required amount of cash otherwise error message.

R1.1: Select withdraw amount option

input: withdraw amount option.
output: User prompted to enter account type.

R2.2: select account type.

input: User option saving/ cheaking/dep site.
output: enter the amount.

R1.3: Get required amount.

input:
outpt:

Organisation of the SRS document:-

To organise SRS document in general IEEE 830 standard followed

The SRS document should be organised into the indicated sections and each of the sections should discuss the items mentioned under the respective section heading.

1. Introduction:
   (a) purpose.
   (b) overview.
   (c) Environmental characteristics.

(i) Hardware.
(ii) peripherals.
(iii) people.

2. Goal of implementation.

3. Functional Requirements

(a) User class 1

(i) Functional requirement 1.1
(ii) Functional requirement 1.2.

4. Non-functional Requirements.

(a) External interfaces.
(b) User interfaces.
(c) software interfaces.
(d) communication interfaces.

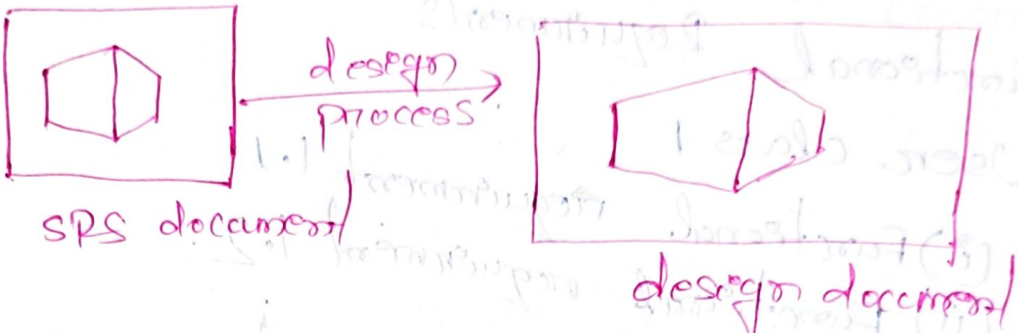
(5) Behavioral accept of the system

(a) Different system states (b) events and actions.

SOFTWARE DESIGN. Dt:-19.09.16

Chapter-IV

The activety is carried out during design phase called design process (Transform the SRS document into design document).



Outcome of a design process

Different modules required

A problem is divided into no. of modules each module consists of no. of function and data shared among the function.

control relationship among modules

It is necessary to control relationship among modules due to function calls betn diff modules.

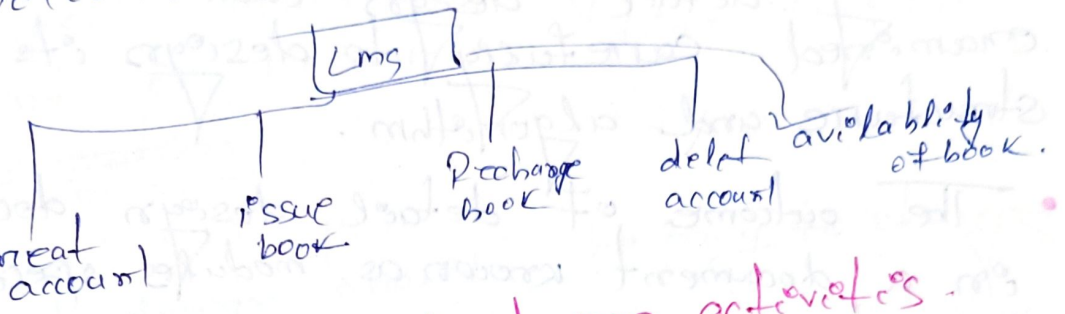
Interfaces among different modules.

Interfaces among different modules identifies exact no. of data items exchange betn modules during function calls.

Data structure of Individual modules.

Each module consists of no. of data items that are shared among different fcn's so it is necessary to properly define

data structure among the modules.
Algorithm need to implement Individual module
Each module has specific task. So different algorithms are req to implement functionalities of modules.



classification of design activities.

- A good software design is arrived at a single step procedure which is known as design activity.
- Through high level design, a problem is decomposed into a set of modules, the control relationships among various modules identified and also the interfaces among various modules are identified.
- The outcome of a high level design is called program structure or software architecture.
- After high level design is made a problem decomposed into no. of sub modules that are cohesive (sticking together) have low coupling among themselves and are arranged in hierarchy manner. Different notations are used to represent high level design, Tree like diagram known as structure chart

- other notations like UML (unified modeling language) used to document object oriented design of a system.
- During detail design each module is examined carefully to design its data structure and algorithm.
- The outcome of detail design documented in a document known as modules specification documents (MSPEC)

Classification of design methodology:-

Two methodology is used
(i) procedural.
(ii) ~~others~~ object oriented approaches.

Analysis vs Design

- The goal of any analysis technique is to elaborate customer requirements.
in case of analysis a model detail documentation part made by using DFD (Data flow diagram) where as design made by using structure chart
- UML diagram is used for both analysis and to design a module.

characteristics of a good software design

correctness

A good software design should is of

all correct.

Understandability

A good software design should be easily understandable.

Efficiency

A good software design should be efficient in resource, time, cost, (obtemsate)

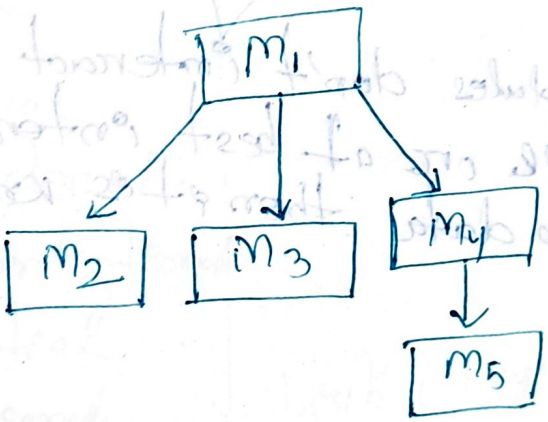
~~obtemsation~~

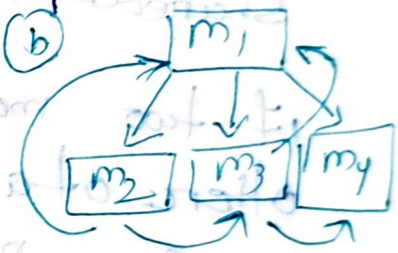
Maintainability

A good design should be easy to change

Understandability of a design D T:- 20.09.16

- A design solution is understandable if it is modular and it is arrange in layers.





A design solution is considered to be highly modular if the different modules in the soln have high cohesion and they are intermodule coupling are low.

Cohesion means coth in a module no. of functions call each other, if strong connection then high cohesiveness. other wise low cohesion. If two modules interact with each other then two modules coupled

5 each other.

Details of cohesion and coupling

IMP Cohesion is a measure of the functional strength of a module whereas the coupling between two modules is a major of the degree of interaction betn the two modules.

Defn of coupling

If the function calls betn two module involves passing of large no. of shared data then the modules are tightly coupled.

If the interactions occur through some shared data then they are highly coupled.

If two modules don't interact with each other at all or at best interact by no data passing then it is known as low coupling.

Cohesion

1. Functional independence:

A module i.e highly cohesive and low coupling with other modules is said to be functionally independent of other modules.

advantages

• Error isolation.

When ever an error exist in an ... independence reduces the

the chances of errors propagating to other mode.

- scope of reuse
  Reuse of the module for the developme- nt other application become easier.
- understandability
  If a a module is independent then, complexicity of design greatly reduce.

V. VIMP

Explain briefly classification of cohesion

coincidental ↓ low cohesion
logical
Temporal.
procedural.
communicational
sequential
functional. ↓ high cohesion

(i) coincidental cohesion

A module is said to have coincidental cohesion if it performs a set of task that relate to each other very losely.

(ii) Logical cohesion

A module is said to be logically cohesive if all elements of the module perform similar operations

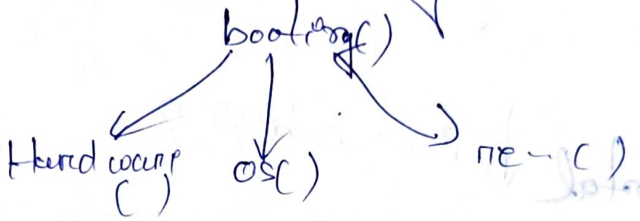
Ex

Print gread sheet, print attendant sheet
print student admission data sheet.

Temporal cohesion

When a module contains functions that are related by the fact that ~~each of~~ these functions are executed in some time duration. Then the module is said to be temporal cohesion.

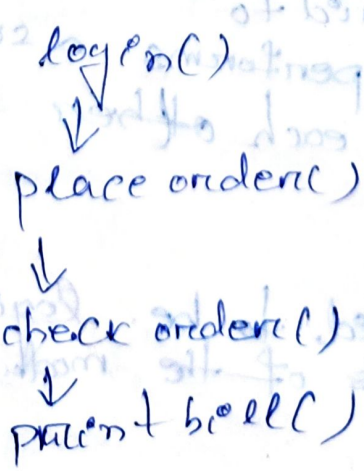
Ex:- computer booting process.



Procedural cohesion

A module is said to procedural cohesion if the set of functions of the module are executed one after other though these functions may work entirely different purposes and operate on many different data.

Ex



3/ communicational cohesion DT:-21.09.16

A module is said to be communicational cohesion if the functions of the module refers or update the same data structure.

ex module Name: student
Table: student records

- Add new student()
- entry mark()
- calculate grade()

Sequencial cohesion

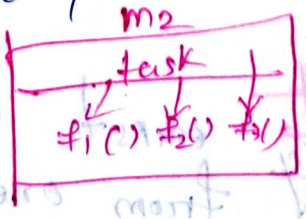
A module is said to sequential cohesion if different functions of the module execute on a sequence and output from one function is input to the next in sequence.

ex

M1
f1()
↳ o1
f2(↓ input)
o2
f3(↓)

functional cohesion

A module is said to functional cohesion if different functions of the module co-operate to complete a single task.



v.imp

Classification of coupling explain briefly?

The degree of coupling between two modules depends on their interface complexity. (Exchanging of data among two modules)

Data	Low coupling
stamp	↓
control	↓
common	↓
content	High coupling

Data coupling

Two modules are data coupled if they are communicating using an elementary data i.e passed as a parameter between the two modules.

EX An integer or a character.

Stamp coupling

Two modules are said to be stamp couple if they communicate using composite data items.

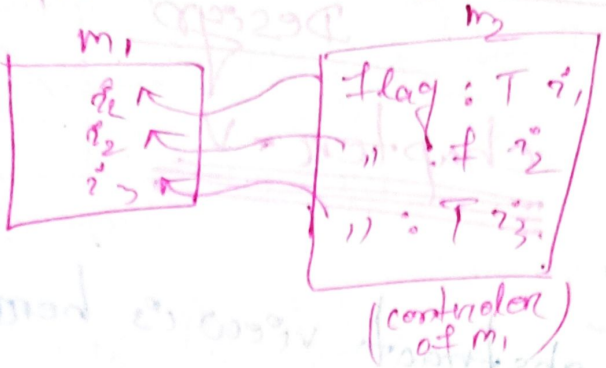
ex:- structure in C.

control coupling

Control coupling exist between two modules if data from one module is

10 used to direct the orders of instruction execution on another.

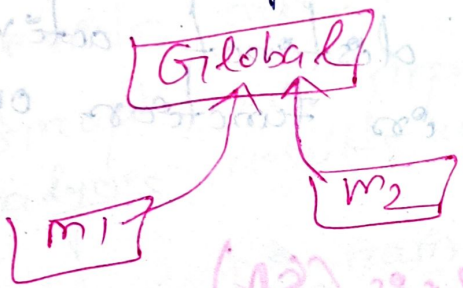
Ex



Common coupling

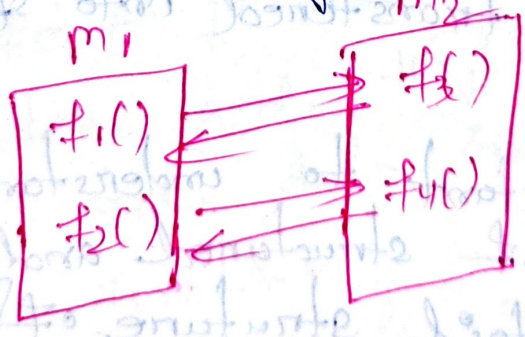
Two modules are common coupled if they share some global items.

Ex



Content coupling

Content coupling exists between two modules if they share programming code

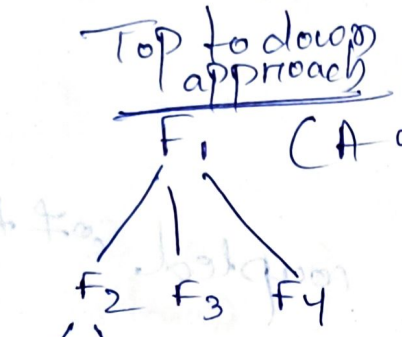


— o —

Function oriented Software Design

chapter - V

Top to down approach



(F1 CA abstract view is here).

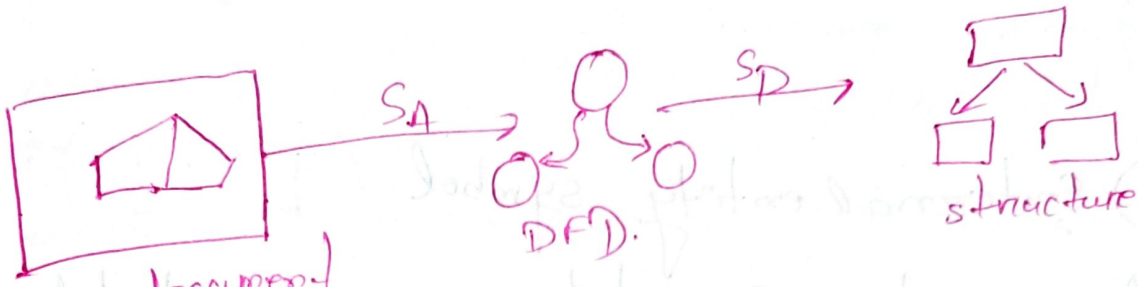
(Here detail functionality is there).

- There are two distinct activities are carried out in function oriented software design.

(i) Structure analysis (SA)

(ii) structure Design (SD)

- In SA SRS document is transfered into DFD model. In structure design DFD model is transfered into structure chart.
- It is important to understand that the purpose of structured analysis is to capture the detail structure of the system as perc receive by the user. while as the purpose of structure design is to define str^{n} of the soln i.e suitable for implementation in some programming language.



Structure Analysis

The principles of structured analysis is explained in the following

(i) Top down approach.
(ii) Devide and concure approach.
(iii) Graphical representation of the analysis results using DFDs.

DT:- 26.09.16

Data flow Diagram

A DFD is a hierarchical graphical model of a system that shows the different processing activities of function and that the system performs and data interchange among these function.

(i) primitive symbols used for constructing DFD

process symbols (◎) ⇒ A function is represented using a circle. This circle is known as process or bubble. This bubble represents function name.

(ii) External entity symbol ( )

An external entity represented by a rectangle.

This represents external to the software system.

Ex:- like human user, external hardware and other software are external with the software being modeled.

(iii) Data flow arrow (→)

A arrow is used as a data flow symbol. A data flow symbol represents flow of data between two processes

(:) Between external entity and process

A data flow arrow represented with a data name.

(iv) Data store symbol ( )

A data store symbol represented by two parallel lines it represents logical file. This symbol represents data store or a physical file on a disk.

Each data store symbol connected by arrows, and arrow indicates flow of data into data store or flow of data out from data store.

(v) output symbol

An output symbol represents hard copy produced by the system.

(vi) Developing DFD Model of a system.

(i) The DFD model of a problem consist of many DFDs and a single data dictionary. A data dictionary lists the purpose of all data items and the definition of all composite data items in terms of their component data items.

(ii) The DFD model of a system is constructed by using hierarchy of DFDs. The top level of DFD is called level 0 DFD or context diagram. level 0 DFD represents abstract of the system.

At successive level DFD is decomposed into level-1, level-2 DFDs. level-1 DFD consist of bubbles upto 7. More than 7 bubbles represented in level-2 DFD.

Imp The context diagram establishes the content in which the system operates i.e who are the users, what data input to the system and what data received by the system.

Imp Decomposition of bubbles known as leveling of bubble.

- The data that flow into or out of a bubble must match the data flow at the next level of DFD. This is known as balancing of a DFD.
- It is important to realise that a DFD represents only data flow and it doesn't represent any control of information in the system.

ex RMS calculating software.

A software is called RMS software read 3 integers and determine RMS of 3 input nos and display it.

Level-0 DFD

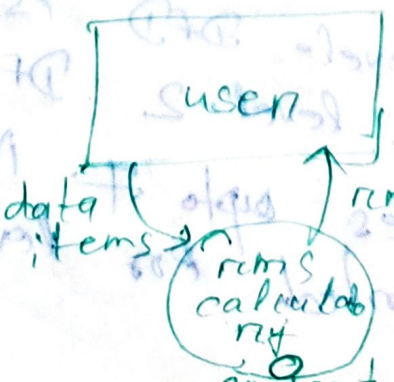
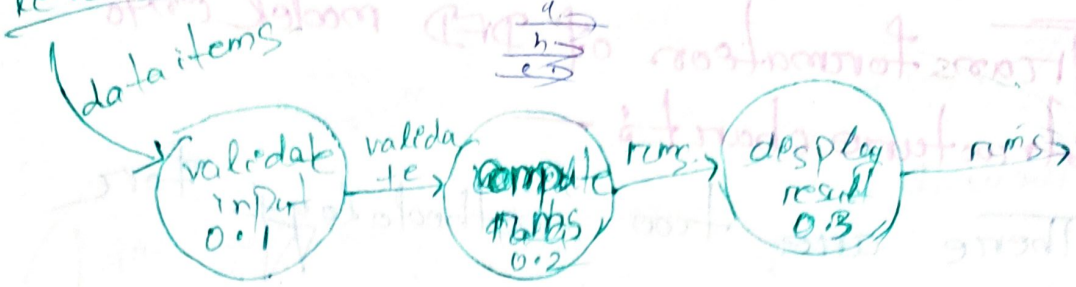
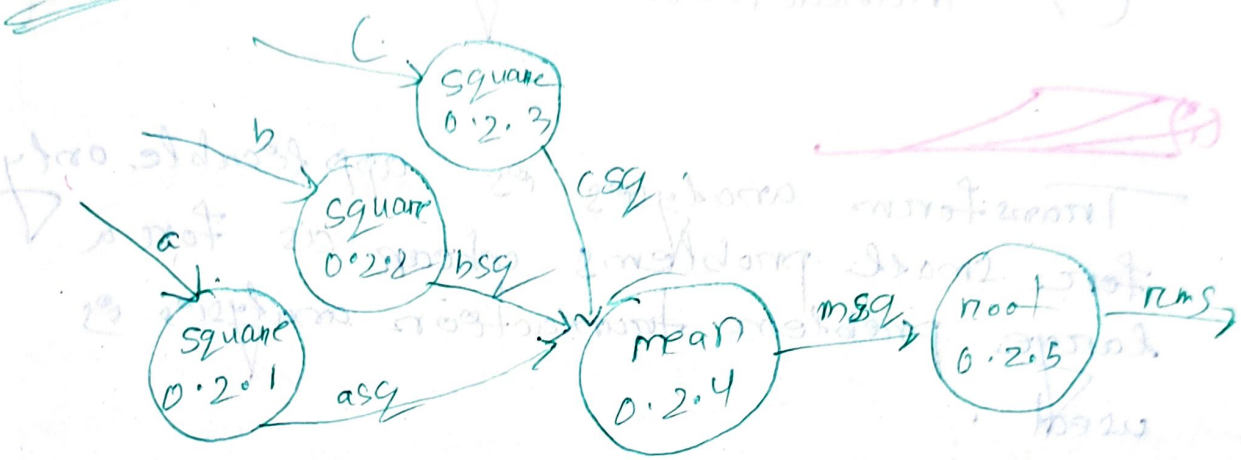


diagram.

level-1



level-2



Data dictionary for the DFD model

data item : {integer}3
rms : float
valid data : data item
a : integer
b : integer
c : integer
asq : integer
bsq : integer
csq : integer
msq : float

DT:- 28.09.18

Transformation of DFD model into structure chart:-

There are two methodology for conversion

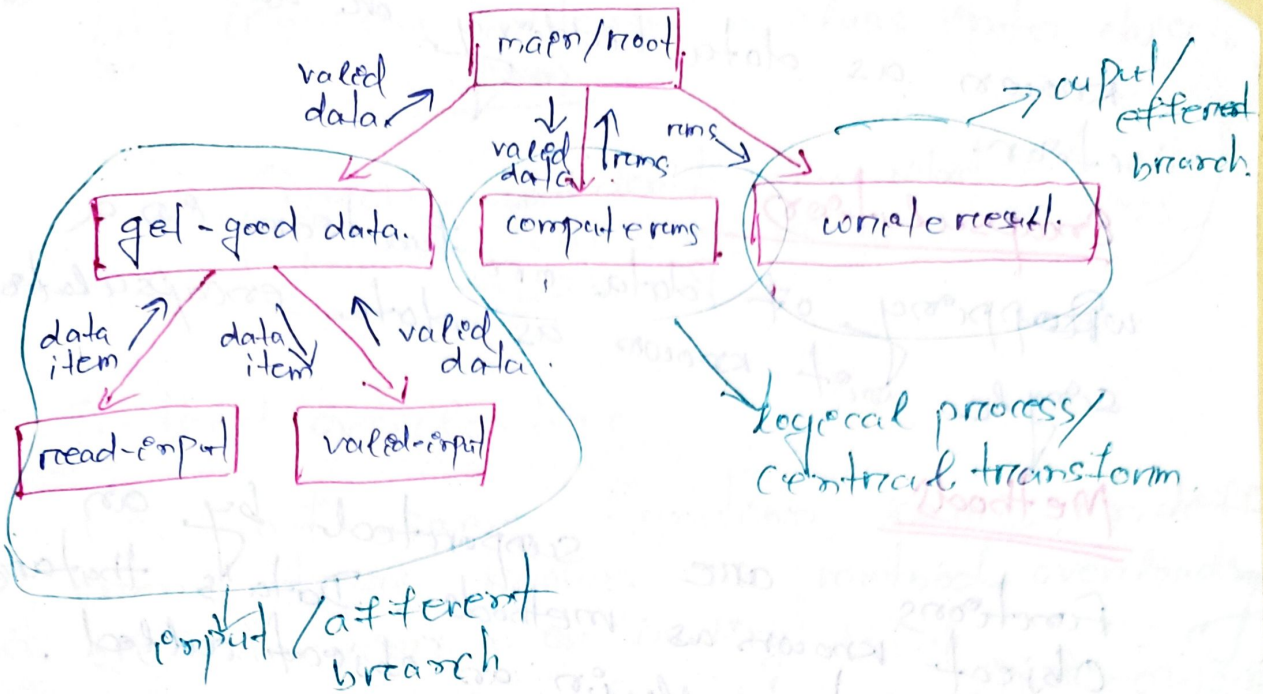
(i) Transform analysis.
(ii) Transaction analysis.

Transform analysis is applicable only for small problems where as for a large problem transaction analysis is used.

(i) Transform analysis

- The first part conversion of DFD into structure chart is divide DFD into 3 parts: Input, logical processing, output.

(a) The input portion in DFD is called afferent branch. The output portion of DFD called efferent branch. The remaining portion of DFD known as central transform.



CHAPTER

Basic concepts of object oriented Analysis and Design

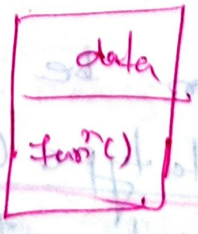
Imp 10 marks

Explain briefly oops concept.

object (object is used to access member of class).

When the system is analysed, developed and implemented in terms of natural objects, it becomes easy to understand the design and implementation of the system

class



O_1, O_2, O_3

Data hiddeing

Hiddeing of data from outside object

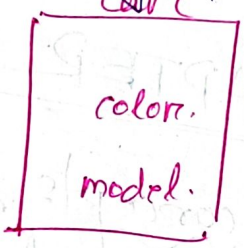
known as data hiddeng or data abstr
elion.

Encapsulation

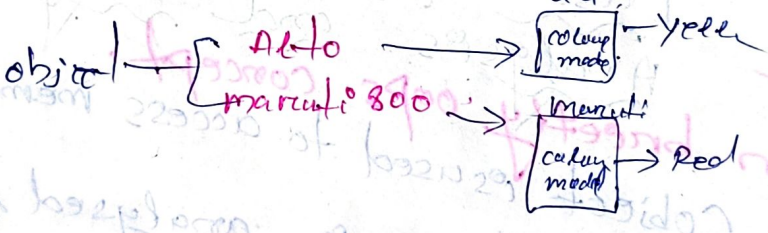
wrapping of data or function in a single unit known as data encapsulation

Methods

functions are supported by an Object known as methods. Data is stored internally in an object called attributes.



attribute = properties are car has pro
colon, model.



Class

- A class consists of no. of objects. objects has similar attributes and having similar methods could prepare a class.
- A class can be considered as Abstract (great) data type (ADT).

(i) The data of an object can be accessed only through its methods. or others controls the copy data of stored internally or the object is abstracted out.

(ii) We can create a class into objects.

(Two objects don't know what is in other but they know they member of same class)

Method overloading

Use of same function name or different forms known as method overloading or function over loading.

D T:-03-10-15

To the implementation of class responsibility through multiple methods known as method overloading.

Ex method overloading

display (solid line) ——————

display (dotted line) .....

display (dash line) - - - -

class relationships

inheritance que (Difference could come) 5 marks

If a class is derive from the existing class then it is known as inheritance.

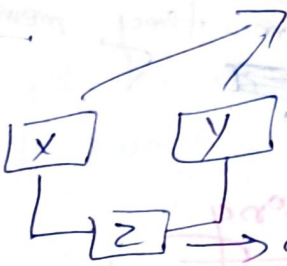
x → base class

↓

y → derive class.

If a class is derive from a single base class it is known as single inheri-

if a class is derive from more than one base class it is known as multiple inheritance.



The important advantage of using inheritance mechanism in programming include code reuse and simplicity of program design.

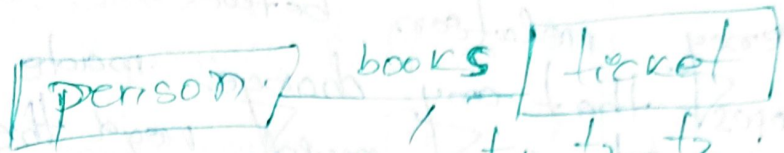
A derive class may give a new definition to a method which already exist in base class redefination of a method which already exist in a base class known as method overriding.

Association

If a class associated with another then corresponding objects of two classes know in to each other. This type of relationship known as association relationship i.e each object of one class in the association relationship knows address of corresponding object of other class. When two classes are associated the relationship bet^n two object of two

corresponding classes known as link.

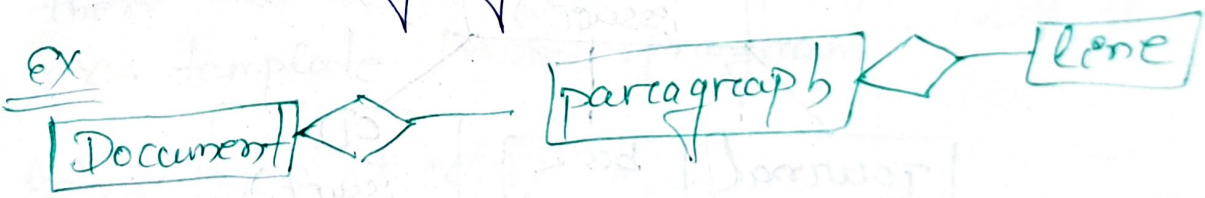
ex



Imp Aggregation (◇)

Aggregation is a special type of association relation where the involved classes are not only associated to each other but a whole-part relationship exist between them i.e the aggregate object not only knows addresses of its parts and therefore invoke methods of its parts, but also takes responsibility for creating and destroying its parts.

ex



Imp Composition

composition is strict form of aggregation in which the parts are existence dependent on the whole. This means life of the parts can't exist outside the whole. when the whole is created, the parts are created, when the whole is destroyed the parts are destroyed.

ex

imp

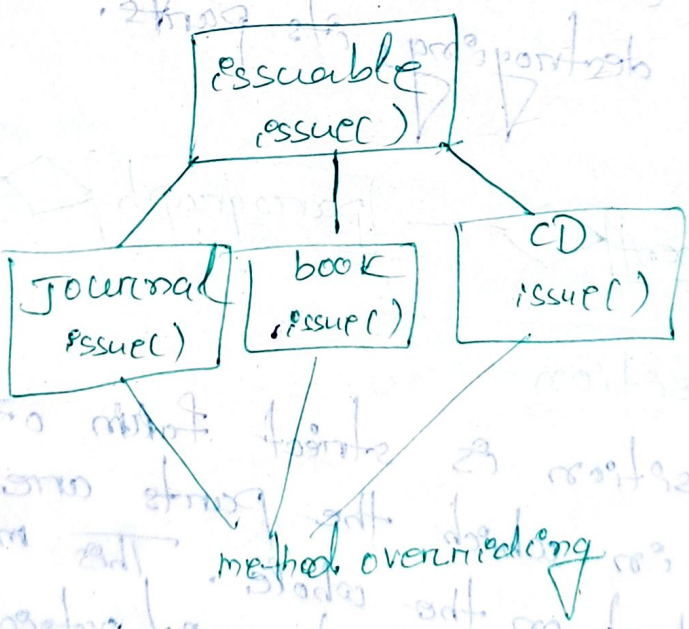
Dependency

A dependency relation between two classes shows that any change made to the independent class would regd the corresponding change to be made to the dependent class

A class implements interface class. If any change made in interface class then it is necessary to make exact change to the class implementing interface as well.

Abstract class

The classes that are not produce. instances (objects) known as abstract class



Polymorphism

It means many forms of the same function name.

polymorphism is of two types:—

1. static polymorphism.
2. Dynamic polymorphism.

At compilation time if compiler know which particular function get executed for a function call then it is known as static polymorphism. Ex:- function overloading.

- At runtime if binding establish between function call and function definition then it is known as dynamic polymorphism.
- Dynamic binding helps for program code reuse and maintenance.

Genericity

We can make a class as generic, a variable or function as generic.

In a program if generic concepts is there then known as generic programming.
Ex:- template based program.

Advantages of OOPD

- programming code and Design reuse productivity increases.
- Testing and maintenance are easier. programming code and design easily understandable.

Unified Modelling Language (UML)

- It is a language used to document object oriented analysis and design results using methodology.

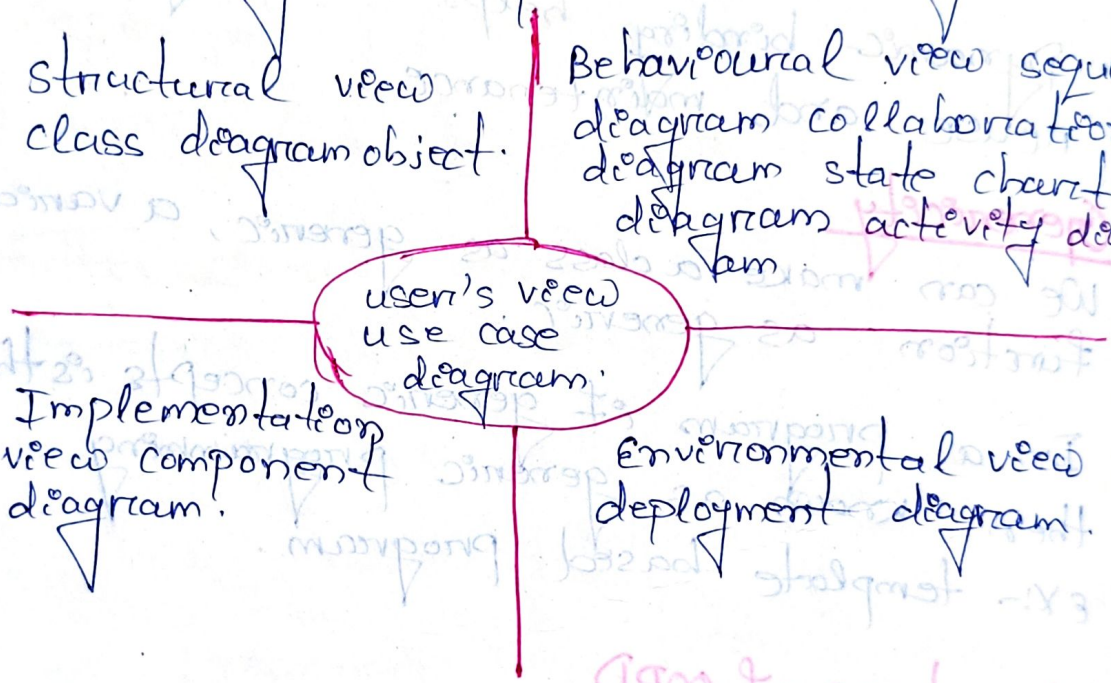
- UML is a language used for creating models and it is not a design methodology

What is a Model

A Model is an abstraction of a real problem and is constructed by leaving out unnecessary details. This reduces the problem complexity and makes it easy to understand the problem.

UML Diagrams

UML diagrams captures following views



1. use case diagram

The user's view captures the view of the system in terms of functionality offered by the system to its users.

2. Structural view

It defines the structure of the kinds of objects, class, important to the understanding of working system and implementation.

3. Behavioural

The Behavioural view captures how objects interact with each other.

4. Implementation view

This view captures important components of the system like graphical user interface/ database part etc (GUI)

5. Environmental view

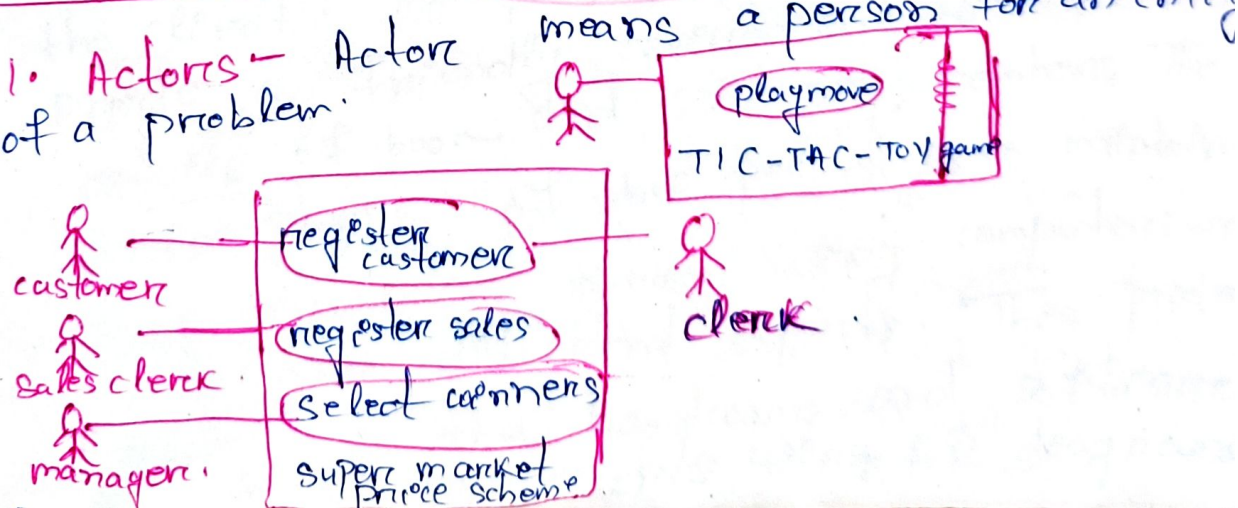
This view models captures how different components implemented on different pieces of hardware.

Use case Model

- The use cases represent the different ways in which a system can be used by the users.
- An use case can be viewed as a set of related scenarios tied together by a common goal the main line sequence and each of the variations are called Scenarios or instance of use case.
- The use case model represent a functional or process model of a system.

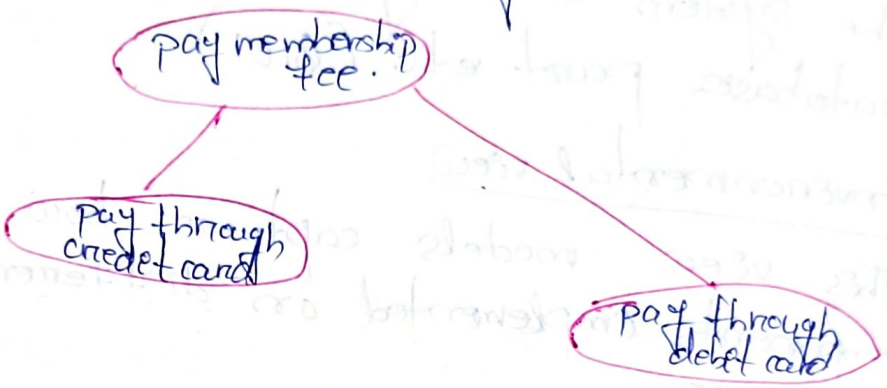
Components of use case:-

1. Actors - Actor means a person for an entity of a problem.



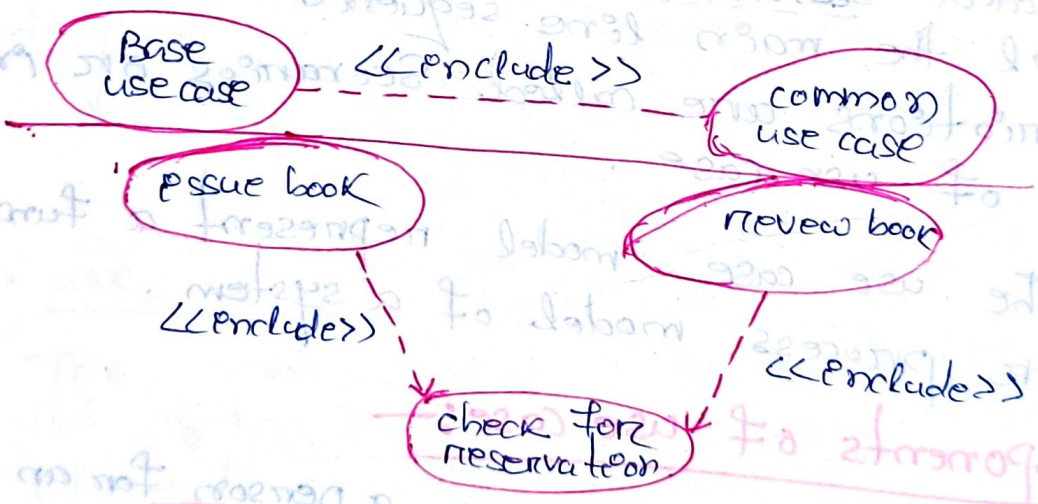
Factoring of use case:-

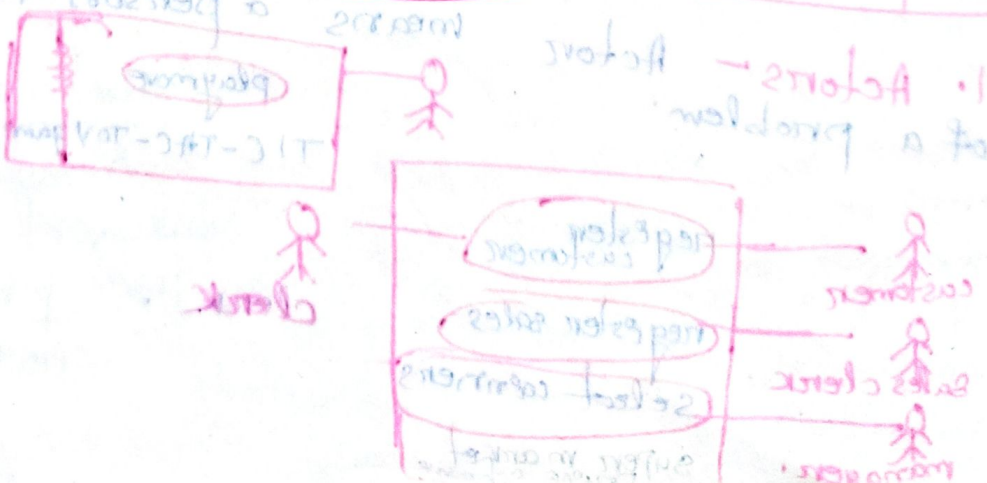
1. Generalisation:- Use case generalisation used when we have one use case that is similair to another but does something slightly different or something more.



2. Includes

In older version of UML it is known as uses relationship. The includes relationship implies one use case includes behaviour of another use case in its sequence of events and actions.





DT:- 05.08.16

Library management system

(i) User registration.

(1) Write SRS on Hospital management system.
(2) Write SRS on online examination system.
(3) Write SRS on student admission system.
(4) Write SRS on Hotel management system.
(5) Write SRS on railway reservation system.
(6) Write SRS on store management system.
(7) Write SRS on motor vehicle management system.

1) Design the SRS on Library information system

Introduction

purpose

The main objective of this document is to illustrate the requirement of the both functional and non-functional requirements purposed by the client. The purpose of this project is to provide a friendly environment to maintain the details of book and library members. The main purpose of this project is to maintain easy circulating system using computers and to provide different reports. This project describes the hardware and software interface requirements using ER diagrams

and UML diagrams.

Overview

- The implementa

Functional requirement

- Administrator menu: In this module one can do the following
  - → Admin should be able to insert, modify and delete books.
  - → can accept or reject a new users according to the library policy.
  - → can get the information of any member who has borrowed a book.
  - → Add and edit book categories and arrange books by categories.
  - → can record books returned by users

Book entry:- In this module we can enter a new book and store it.

Book issue: In this mode we can issue a book to the student

Book record: In this module we can keep the records of various books being issued and returned

Register student: In this module we can keep the record of how many students have we registered.

Non functional requirements

- The user should be simple and easy to understand and use. Also be an interactive interface. The system should prompt for the user and administrator to login to the application. and for proper input criteria.
- The software provides good graphical interface for the user. Any administrator can operate on the system, performing the needs task such as create, update viewing the details of the book. Allows user to view quick reports like Book issues / Returned etc in between particular time.

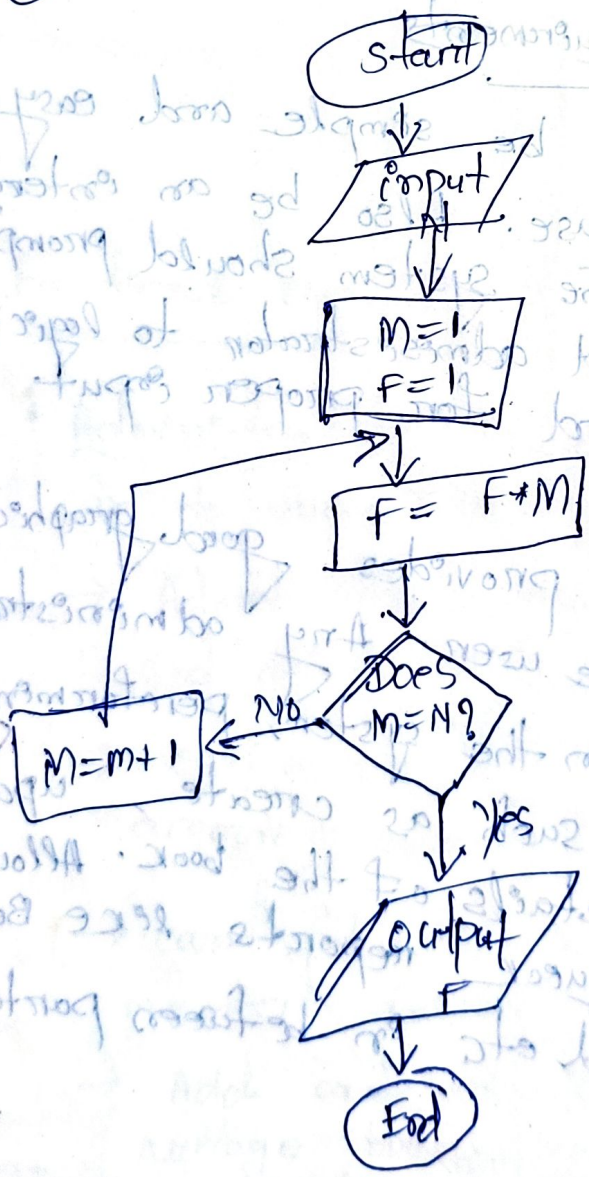
- **Error Handling**

Library management system shall handle expected and non-expected errors in ways that prevent loss of information and long downtime period.

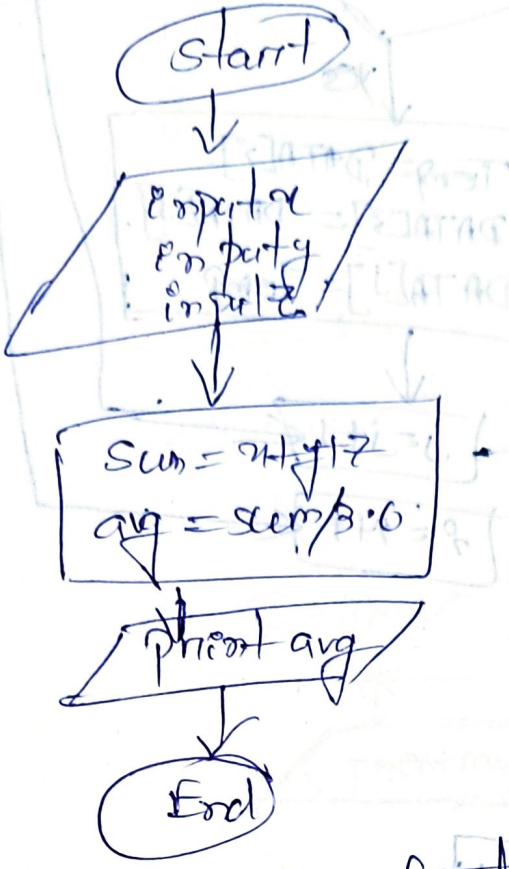
- **Performance Requirements**

The system shall accommodate high no. of books and users without any fault.

(1) Flow chart to find factorial no.



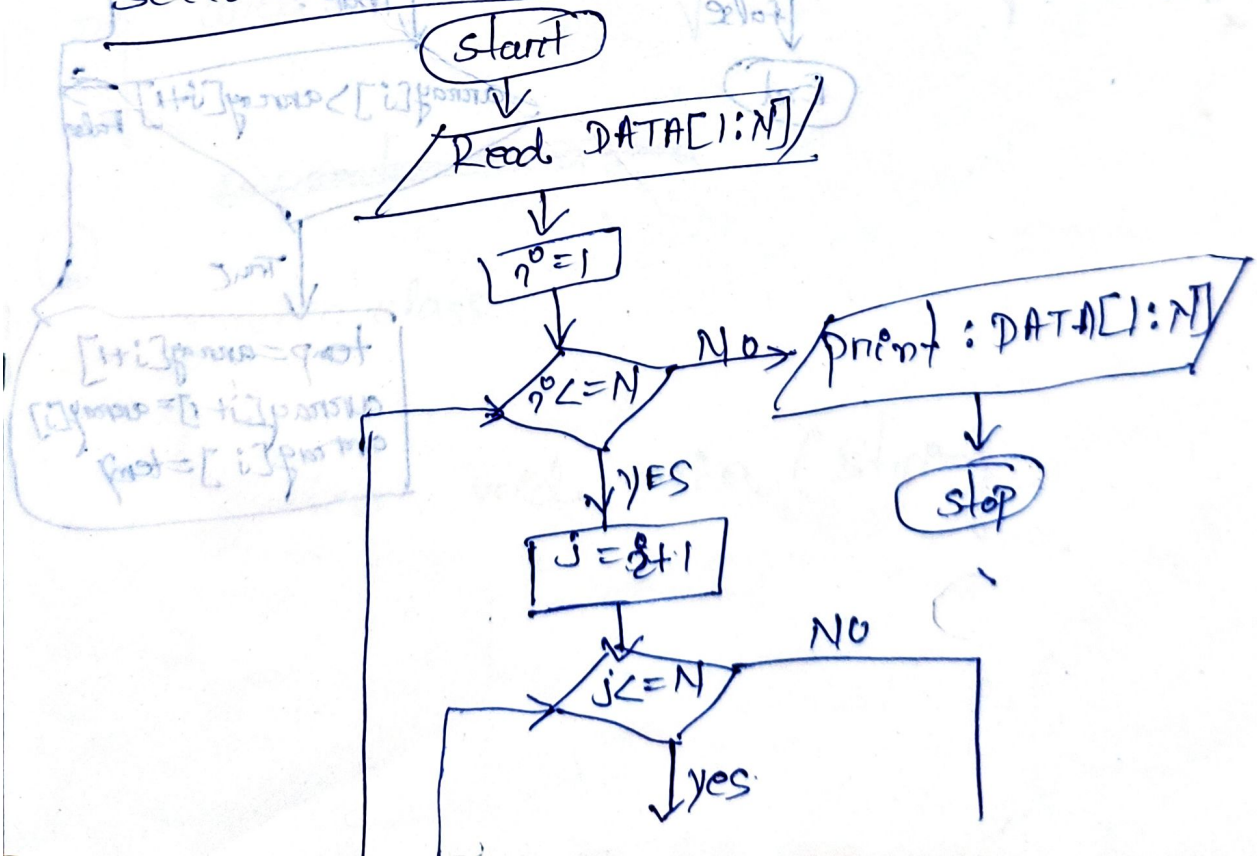
(2) WAP to find out average of 3 Numbers.

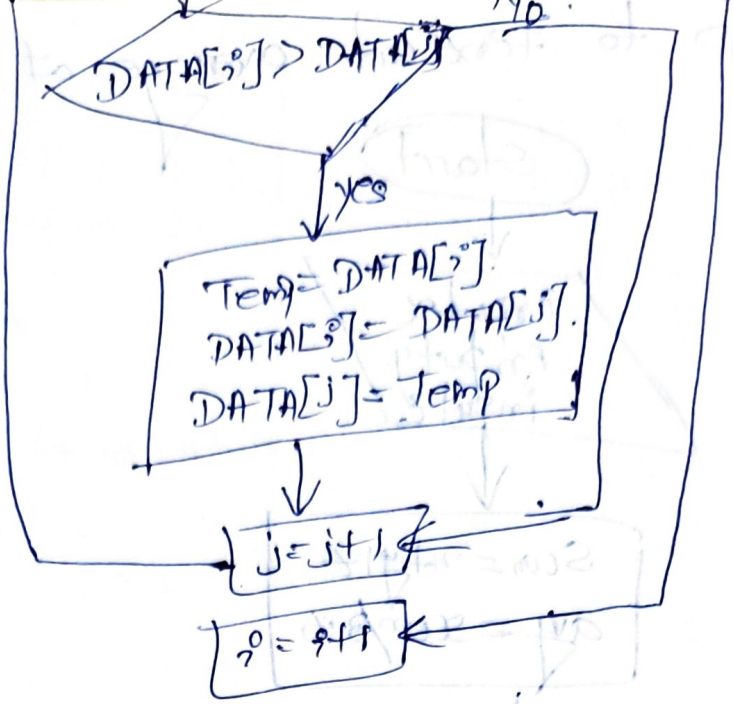


(3) ~~using recursion find factorial~~

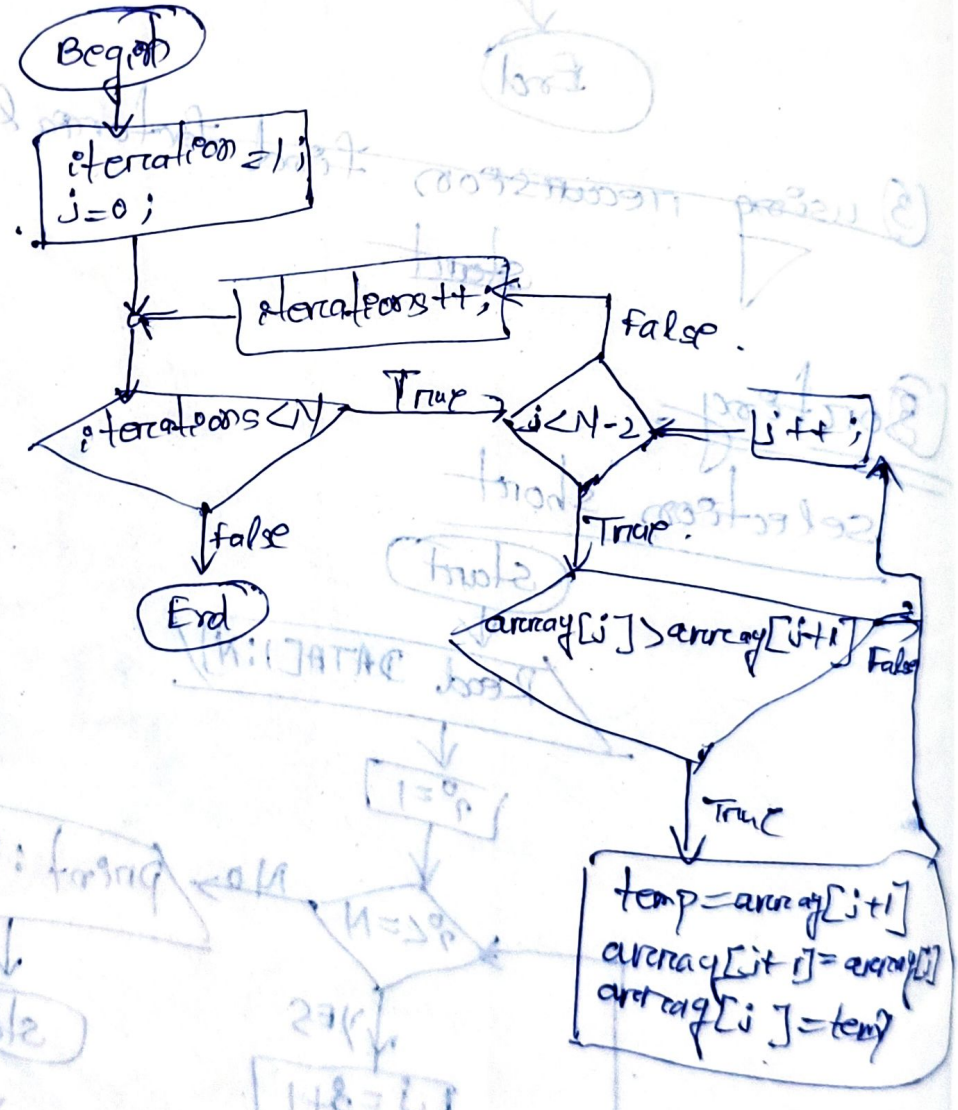
~~Sorting~~

selection short

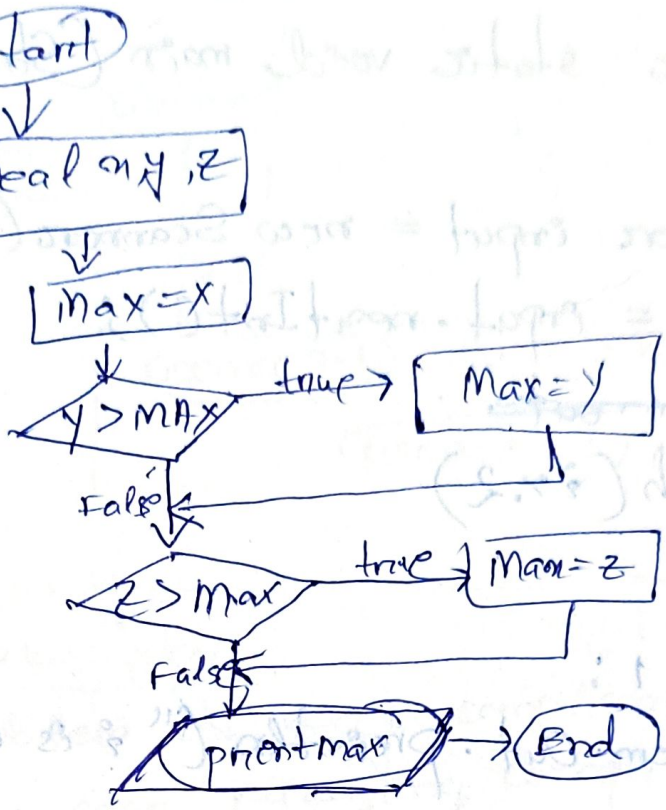



DATA[i] > DATA[j]
yes
Temp = DATA[i]
DATA[i] = DATA[j]
DATA[j] = Temp
j = j+1
i = i+1

Bubbel sort


Begin
iteration = 1;
j = 0;
iterations++;
false
iterations < N
True
i < N-2
j++;
false
End
True
array[j] > array[j+1]
False
True
temp = array[j+1]
array[j+1] = array[j]
array[j] = temp

Insertion sort



(1) WAP to cheack whether a no. is even or odd using switch case.

(2) WAP to dlostrate overloading, for calculating area of rectangle, triangle and circle using constructer

~~import java.u~~

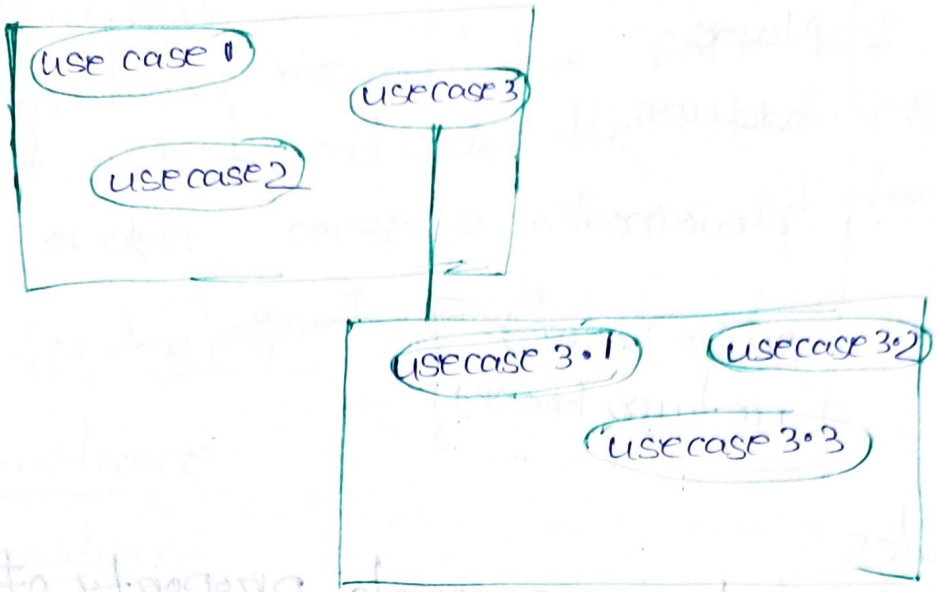
(1)

```
public class
{
static void main (String
```

```
import java.util.Scanner;
Public class SimpleSwitch {
public static void main (String args
{
Scanner input = new Scanner (system.in
int i = input.nextInt();
System.out.
Switch (i%.2)
{
case 1:
System.Out.println (" i is odd.");
break;
case 0:
System.Out.println (" i is even");
```

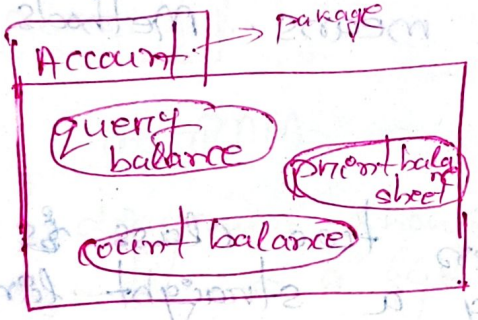
DT:- 18.10.16

organisation of use cases:-



Use case pakageing

If a problem is very complicated then we can use use case pakageing concept.



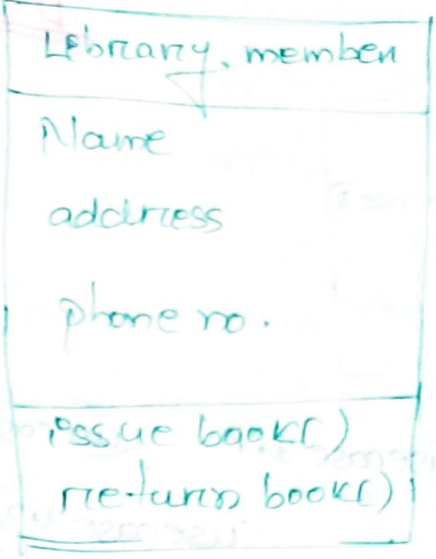
class diagram

A class diagram describes static structure of a system.

It shows how a system is structured rather than how it behaves.

classes

A class consists of no. of attributes and operations.



Attributes

An attribute represents property of a class or represents the kind of data that an object might be contain.

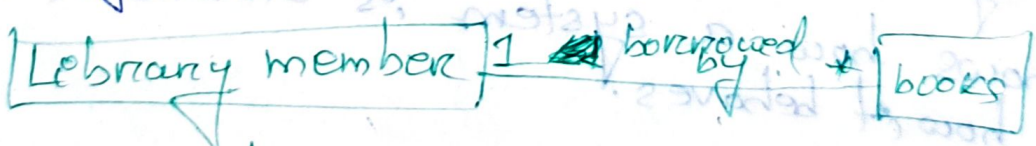
operation

Usually operation means Methods of a class

Association

Association between two class is represented by drawing a straight line in between the concerned classes.

Ex:- Many books borrowed by a library member.

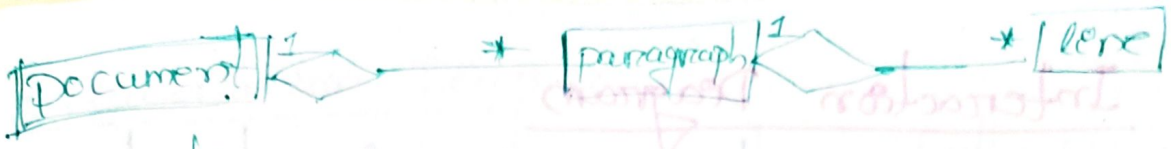


Aggregation

Aggregation means "whole-part" relationship between the classes.

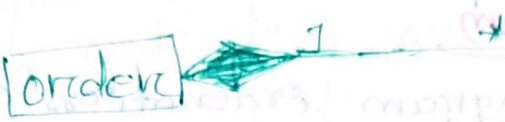
Ex

A document consist of many paragraphs a paragraph consists of many no. of lines.



composition:

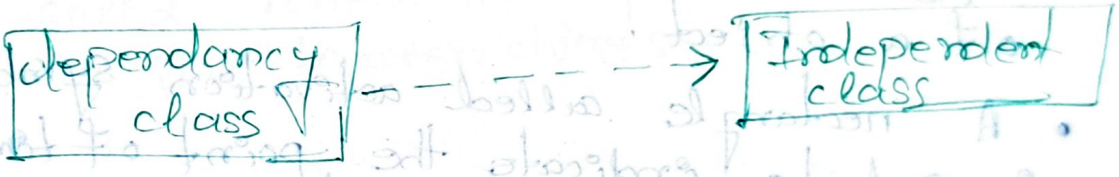
- composition means the life of the parts can't exist outside the whole class.
- A order consists of no. of items.



Inheritance

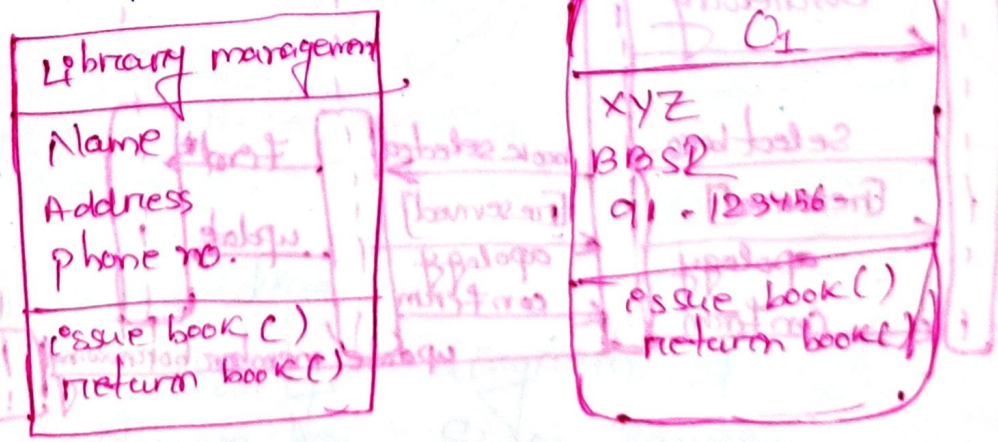
Dependancy

In class diagram a dependancy relation-ship is shown by dotted arrow.



OBJECT DIAGRAM

- object diagrams shows the snap shot of the object in a system at a point in time.
- Since object represents instance of a class it is also known as instance diagram.
- The objects are represented using rounded rectangles.



Interaction Diagram

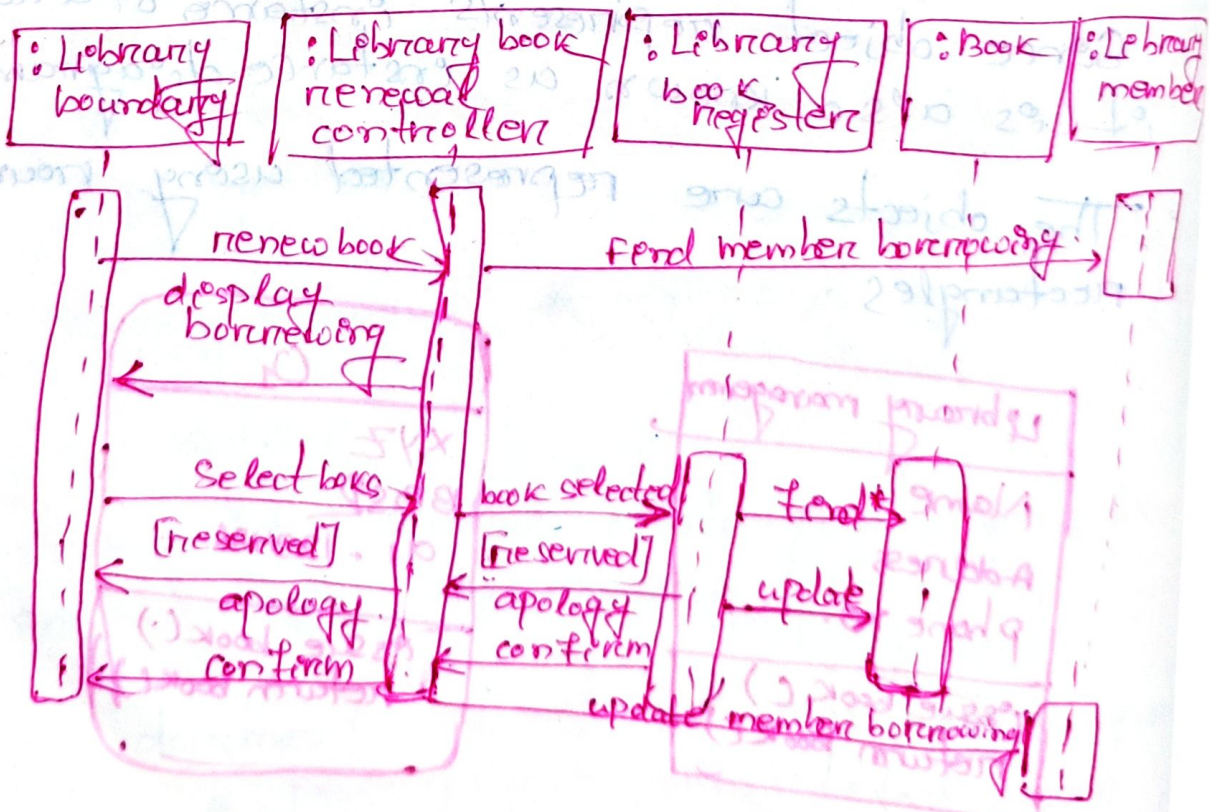
There are two kinds of interaction diagram

(i) sequence diagram
(ii) collaboration diagram.

(i) sequence diagram

In sequence diagram interactions among the objects are shown.

- class name and object name written within a box and both are separated by colon and underline.
- The vertical dashed line shows lifeline of a object.
- A rectangle called activation symbol is used to indicate the point of time at which the object is active.
- Each message is indicated as an arrow between the lifelines of objects.



Collaboration Diagram

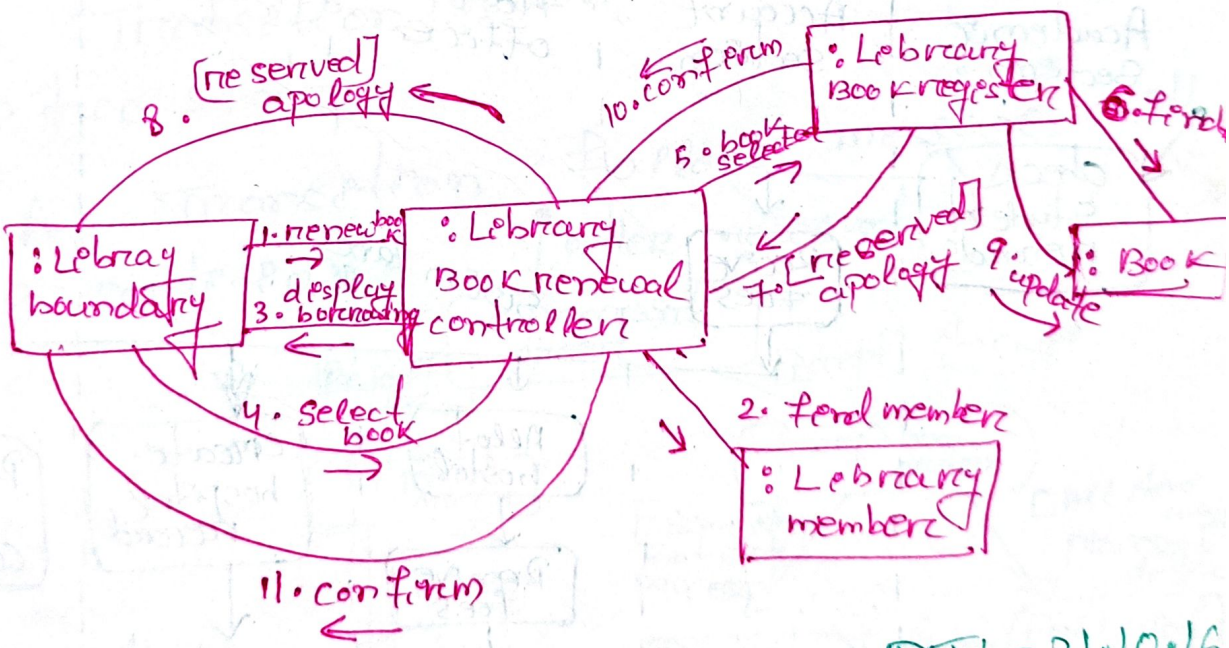
A collaboration diagram shows both structural and behavioural aspect.

The structural aspect of a collaboration diagram consists of objects and links among them indicating association.

Each object is called collaborator.

The behavioural aspect is described by set of messages exchanged among different collaborators.

The link between objects is shown as a solid line. and can be used to send messages between objects.



DT:- 21.10.16

Activity diagram

- In activity diagram parallel activities are shown by swim lanes.

- Swim lanes enable to group activities based on who is performing them.
- The basic elements of activity diagram.

(i) initial state

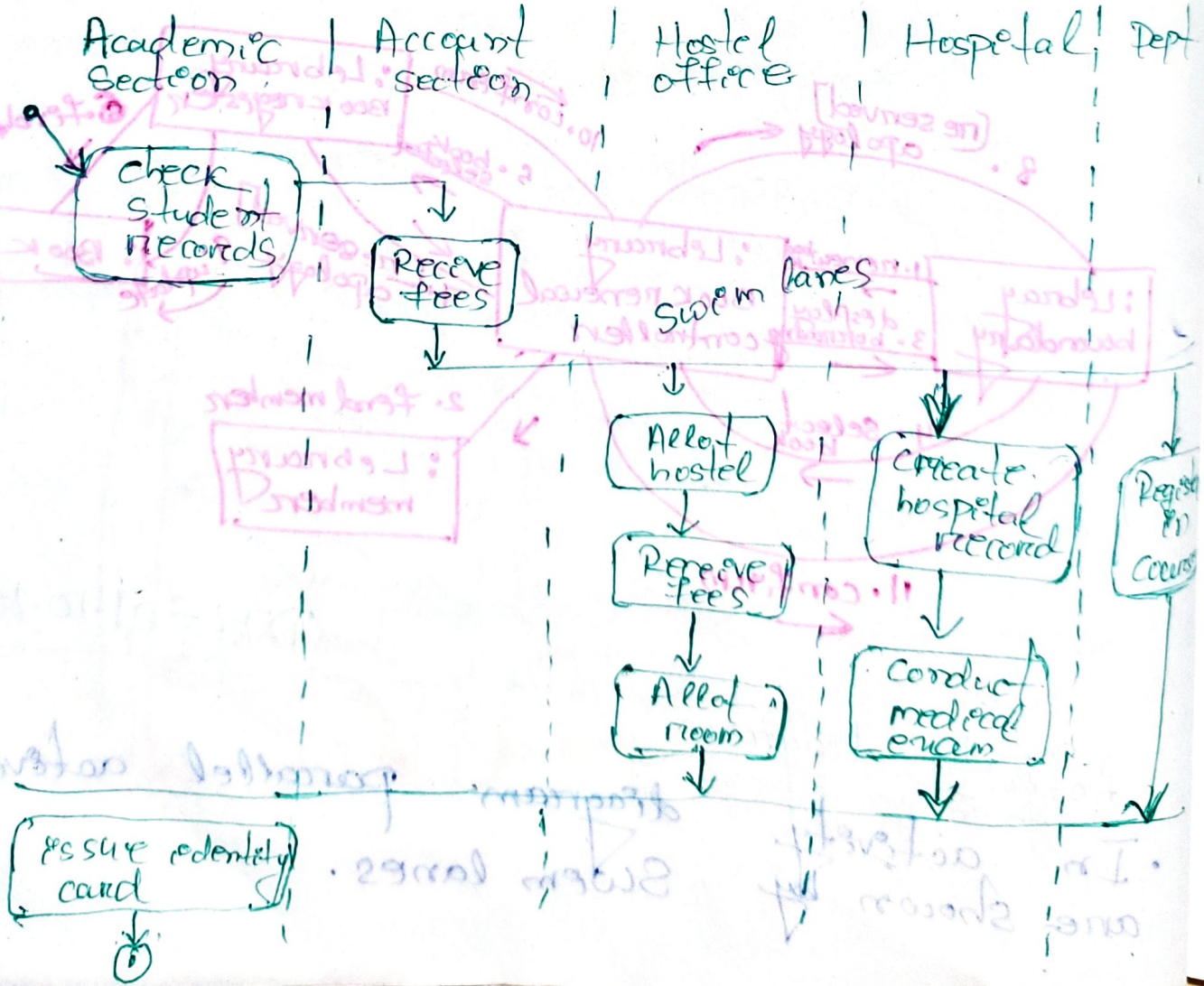
This is represented by filled circle.

(ii) Final state

Represented by filled circle inside a large circle.

(iii) Activities

These are represented by rectangle with rounded corners.



State chart diagram

- State chart diagram is used to state how object is changes in its life time.
- Basic components of state chart diagram

(i) initial state.

It is represented by filled circle.

(ii) final state

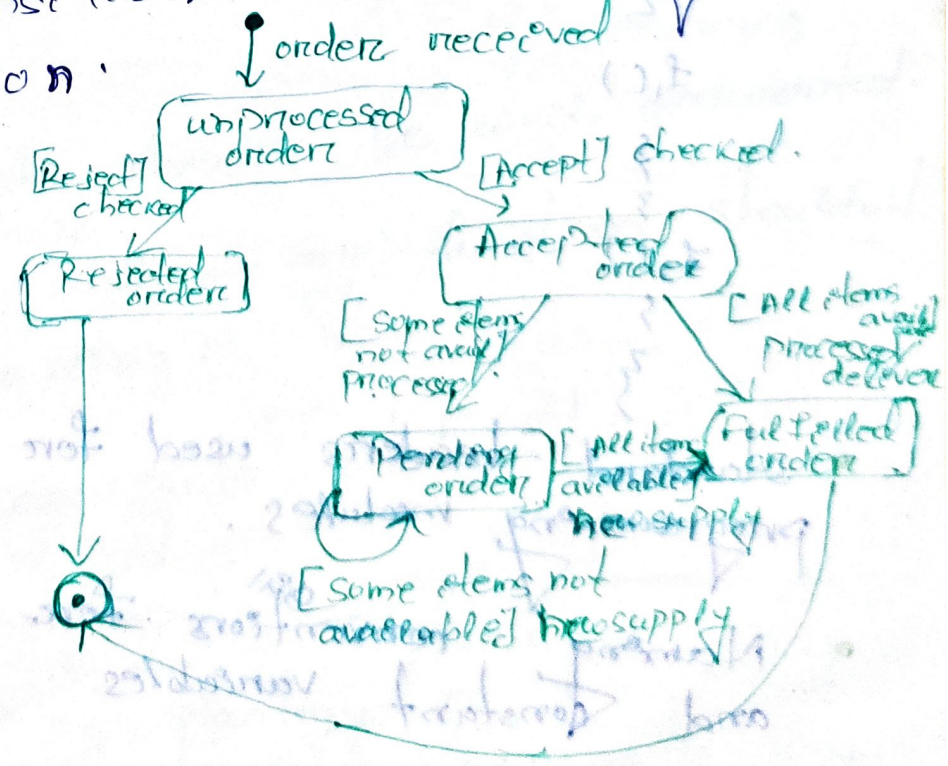
This is represented by filled circle in a large circle.

(iii) state

Represented by rectangle with rounded corners.

(iv) Transition

- A Transition shown as an arrow between two states
- A Transition follow message with a condition.



Module

coding chapter

- coding is undertaken once the design phase is complete and the design documents have been successfully reviewed.
- objective of coding phase is to transform the design of a system into code in a high level language and then do unit test this code.

coding standards and guide lines

- The rules for limiting the use of globals.

ex

```
Global
main()
{
}
f1()
{
}
f2()
{
}
```

- standard headers used for different programming modules.
- Naming conventions for global, local and constant variables

```
Main
main()
{
 int temp;
 const ABC= 10;
 }.
```

Conventions requarding errors return values:—

All functions while encountering errors must returns zero or one respectively.

Different ways to represent coding guide lines:—

- Don't use a coding style too diffi cult to understand.
- Don't use an identifier for multiple purposes.
- The code should be well documented.
- The length of any function shouldn't exceed 10 lines.
- Don't use goto statements.

DT:- 24.10.16

Code review

Code review startegy is more cost effective startegy, As compare to any

Code :

other strategy.

- There are two types of code review

(i) code inspection.
(ii) Code walk-through.

- The main objective of code walkthrough is to detect algorithm and logical errors on the program. There are some guidelines on the following.

(a) The team performing code walkthrough is neither too small or too big.

(b) Discussion should be focous on discovery of errors on the program.

- code inspection

The principal aim of code inspection is to check for the presence of some common types of errors on the programming. Followings are list of classical errors on the program.

(i) Use of uninitialised variable.
(ii) Jumping into loop (infinite loop)
(iii) Non terminating loop.
(iv) Incompatable assignments.
(v) Array index out of bound.
(vi) Improper storage allocation and deallocation
(vi) Mismatch between actual and formal

parameter.

(iv) Improper modification of loop variables

Software documentation

- Software documentation means SRS document, design document, Test document, system manuals. These are parts of software engineering process.
- A good document consists of following properties.
  (i) It helps to understand software product easily.
  (ii) It helps to reduce effort and time requirement for maintenance.
  (iii) It helps user to understand effective use of the system.
  (iv) It helps to track progress of the project.
- There are two types of software document.
  (a) Internal documentation.
  (b) External documentation.

(a) Internal documentation provided by the source code itself. The important type of internal documentation are

(i) use of comment lines (ii) use of meaningful variable names (iii) Include function header.

(b) External

- External documentation is provided through various types of supporting documents such as users manual, SRS, design document, Testing document.
- A systematic software development style says that all the documents should be arranged in proper manner.
- All the documents should be upto date and any change made in code that can be reflected for all documents
- External documentation should be properly made so that all categories of users can be easily understood the software development process

TESTING

- The aim of program testing is to identify all defects in a program
- There are some common terms associated with testing.

(i) An error is a mistake committed by development team during developing any phase.

(ii) A failure is symptom of an error. (hardware, memory, printer etc are failure)

(iii) A test case is the triplet [I, S, O] where 'I' is data input to the system, 'S' is state of the system and 'O' is output of the system.

(iv) A test suite is the set of all test cases with which a software product is tested.

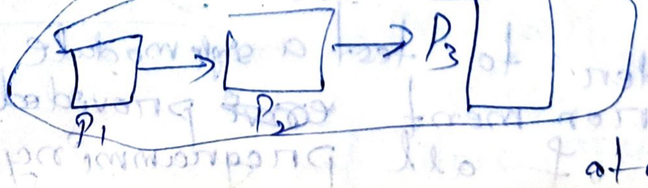
```
if (x>y)
    x++;
else
    y++;
```

(X, Y) → T.C.
(1, 2) → Test case
(2, 1) → T.C.

Test suite → no. of test case are there to check this code

Differentiate verification vs validation

Verification is the process of determining whether the output of one phase of software development conforms to that of its previous phase whereas validation is the process of determining whether a fully developed system conforms to its requirement specification.



validating giving right output acc. to SRS or not.

at atomic check whole

Q) Explain briefly testing activities.

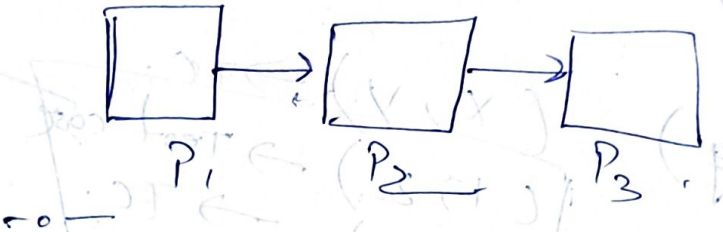
→ Test suite design

→ Running test cases and checking the result to detect failures.

→ Error correction.

After the error is located in previous activity, the code is appropriately changed to correct the error.

Ex



There are two approaches to design test cases.

(a) Black box approach.

(b) White box approach (glass box)

• UNIT TESTING

- Unit testing is referred to as testing in the small.
- Unit testing is undertaken after a module has been coded and reviewed.

DT:- 25.10.16

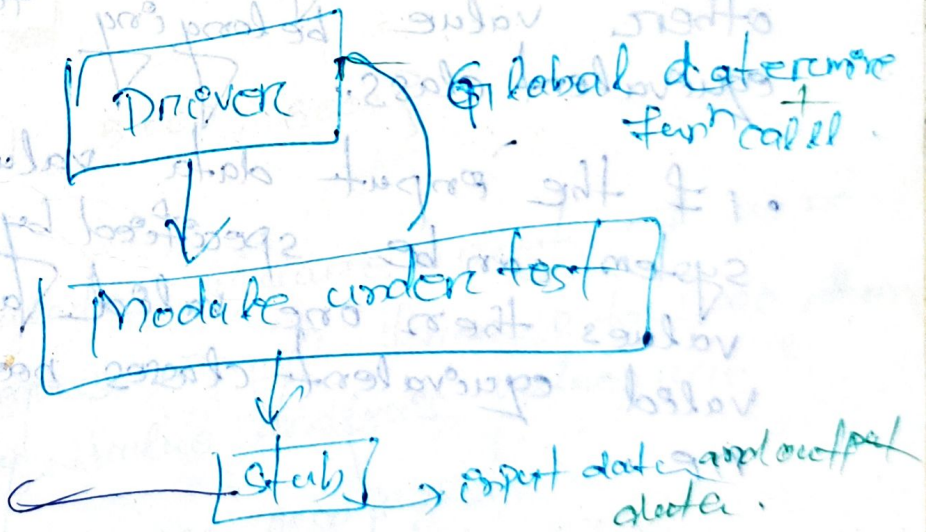
- In order to test a single module a complete environment is provided for execution of all programming code.

DT: 23.10.16

To test the module, the procedure belonging to other modules that the module under test call, The non local data structures that the module accesses, A procedure to call the function of the module under test with appropriate parameters.

A stub procedure is a dummy procedure that has I/O parameter as the given procedure.

A Driver module should contain non local data structures accessable by the module under test. It also contain code to call different functions of the module. Under test with appropriate parameters



Black box testing

- Test cases are designed only for functional specification. functional beha
- It is only for, internal and not for any

structure of the program so it is also known as functional testing.
(here system design should not need only input output) internal str. are

- To design black box test cases two approaches are used.
  - → Equivalence class partitioning.
  - → Boundary value analysis.

→ Equivalence class partitioning

The main idea behind defining equivalent classes of input data is that testing the code with any one value belonging to an equivalent class is as good as testing the code with any other value belonging to the same equivalent class.

- If the input data values to a system can be specified by range of values then one valid and two invalid equivalent classes need to be identified.

EX

Suppose range of value is from 1 to 10 then equivalent class is

[1, 10] → 3 testcases.

[-∞, 0] [11, +∞].

Ex
To calculate square root of an integer the input value ranges from 0 to 5000 determine equivalent class test suite.

{-500, 1000, 5010}.

→ Boundary value analysis

- It is based upon to design test suite that involves designing test cases using the values at the boundaries of different equivalent classes.
- In this analysis the boundary value with too non equivalent values are considered.

Ex
{0, -1, 5000, 6000}.

To calculate square root of a no. of integers range from 0 to 5000, then boundary value analysis values are

{0, -1, 5000, 6000}.

White box testing

- White box test cases require knowledge of internal structure of a problem so it is also known as structural testing.

- This testing strategy bases upon coverage based and fault based testing.

(a) fault based testing

It targets to detect certain types of faults.

ex:- Mutation (change) testing.

(b) coverage based testing

This strategy attempts to execute certain elements of a program.

ex:- statement coverage, Branch and path coverage etc.

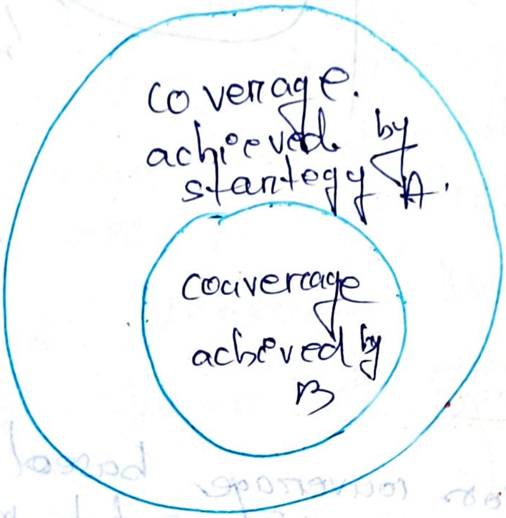
Testing criteria for coverage based testing:-

The set of specific program elements that a testing strategy requires to be executed is called the testing criteria of the strategy.

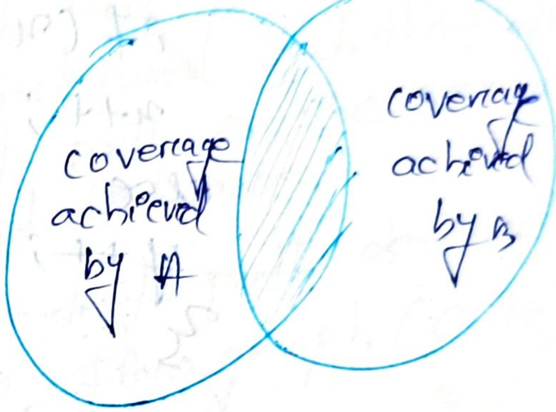
Stronger Vs weaker testing

A white box testing is said to be stronger than another strategy, if all types of program elements covered by second testing strategy

are also couvered by 1st testing startegy, and additionally 1st startegy couvers some more types of elements that are not couvered by second startegy. When non of two testing startegy couver the full program elements then they are called complementary testing startegy.



'A' is stronger than 'B'



A and B are complementary.

DT:- 26.10.16.

statement couverage

The principal idea of statement couverage startegy is that unless a stmt. is executed, there is no way to determine wheather an error exist in that statement:

EX

while (x != y)

{ if (x>y)

x++;

else

y++;

}

to couverage each stmt (3,2).

if testcase is (2,2)

(2,3) → for if statement.

Branch coverage

Branch coverage testing requires test cases to be designed so as to make each branch cond^n in the program to assume true and false values in turn.

Ex

```
while(x != y)
{
if (x > y)
x++;
else
y++;
}
```

Two branch here if and else

```
if (x>y)
x=x+1;
```

Condition coverage

- In the condition coverage based testing test cases are designed to make each component of a composite conditional expression to assume both true and false value.
- In a program if large no. of cond^{n}s are there then it is difficult to cover all the cond^{n}s. Therefore condition coverage is practically useful when 'n' is small where 'n' is no. of conditions.

```
if (a>b)
    if (a>c)
print a;
```

→ more composite cond^n

```
else
  print b;

else
  if (b>c)
    print b;
  else
    print c;
```

Path coverage

- Path coverage based testing strategy requires designing test cases such that all linearly Independent path in the program are executed at least once.
- A linear Independent path defined in terms of control flow graph (CFG)

CFG (control flow graph)

- A CFG describes the sequence in which different instruction of a program get executed. The different no. of statements are represented by nodes.
- Transfer from one node to another node represented by path line or edge.

Ex (Sequence)

1. a = 5.
2. b = a + 2.

1st stmt. then 2nd sequencially.

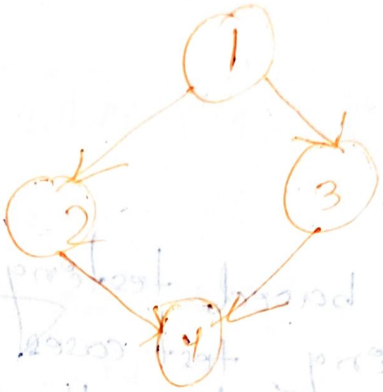
Selection:

each node represent each statement

if = T, F

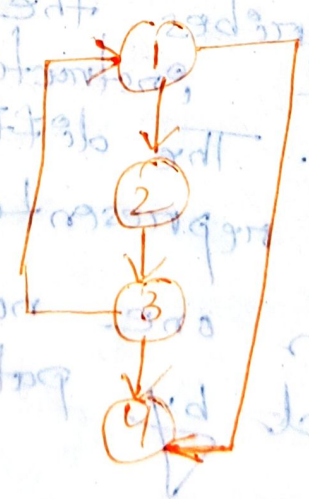
2 executed 3 exec 4

1. if (a > b)
2. c = 3;
3. else c = 5;
4. c = c * c;



Iteration:

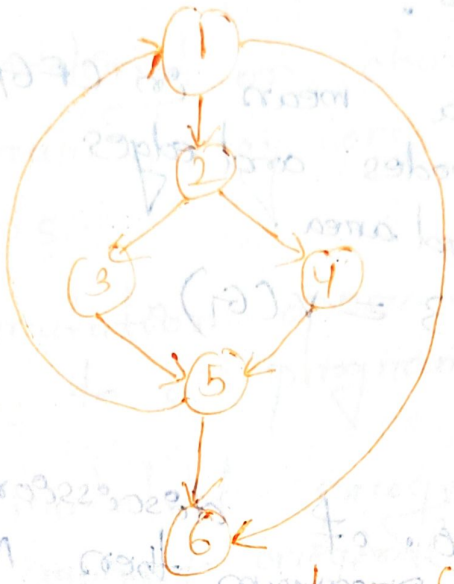
1. while (a > b) {
2. b = b - 1;
3. b = b * a; }
4. c = a + b;



int compute gcd (int x, int y) {

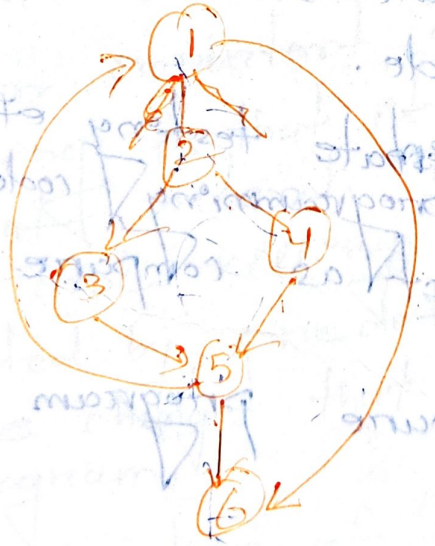
1. while (x != y) {
2. if (x > y) then
3. x = x - y;
4. else y = y - x;

5. }
6. return n;
}.



Mccabe's cyclomatic complexity metric

In a program it is very difficult to search linearly independent set of path. Independent path means on the set contains at least one new path i.e not defined in any other path on the set



linear independent path
1 - 2 - 3 - 5
1 - 2 - 4 - 5
1 - 6
5 - 1

Method:

M1: cyclomatic complexity computed by the following formula $V(G) = E - N + 2$

$V(G) = 7 - 6 + 2 = 3$

Method-2

Total no. of non overlaping bounded areas plus one.

Bounded area mean in CFG the area enclosed by nodes and edges

closed area.

$2+1=3 = V(G)$

Method-3

if 'N' is no. of discission and loop statements of a program then MC Cabe matrics is equal to N+1:

N = 2 = 1 loop and 1 conditional stmt.

$V(G) = 2+1 = 3$

Other uses of MC Cabe Matrics

- It is used to estimate structural complexity of programming code.
- It is used to estimate testing effort because testing much more effect as compare to any other strategy.
- It helps to measure program reliability.

Mutation Testing

- The idea behind mutation testing is to make certain random changes to a program at a time.

- Each time program is changed called mutated program and the change effected is called mutant.
- Depending on changes a mutated program may or may not introduce errors.
- A mutation operators makes specific changes to a program.
- A mutated program is tested ~~orig~~ ~~final~~ against original test suite of the program.
- If there exists at least one test case ~~suite~~ in a test ~~case~~ suite for which a mutated program gives incorrect result then the mutant is said to be dead.
- An important advantage of mutation testing is that it can be automated to a great extent. The process of generation and killing of mutants can be automated by predefining a set of primitive changes that can be applied to the program.

Disadvantage of mutation based testing

- Disadvantage of mutation based testing is that it is very expensive if no. of mutants are large.

- Manual work of mutation testing is not possible because using test cases each mutant need to test, so several tools are available that automatically generate no. of mutents for a given program.

Debugging

There are different types of debugging approach.

(i) Brute force method.

This is the most common method but is least efficient method in this approach print statements are considered through out the program to print intermediate values with the hope that some printed values will help to identify some errors in the statement.

```
main()
{
int x,y;
x = 10;
y = 20;  → print(x,y);
if (x>y)
  do
else
  do.
```

back tracking see from the

(ii) Back tracking

In this approach beginning from the statement at which an error symtom has been observe. The source code is traced back wards until the error is discovered.

(iii) Cause elemination Method

In this approach once a failure is observed, the symptoms of failure are noted. Based on failure symptoms Test are conducted to eliminate errors.

(iv) Program slicing. (is a research topic)

This is similar to back tracking. A slice of a program made to debug and correct errors.

Integration Testing

The objective of integration testing is to check different modules of a program interface with each other properly.

There are four approaches of integration testing.

(i) Big-Bang integration testing

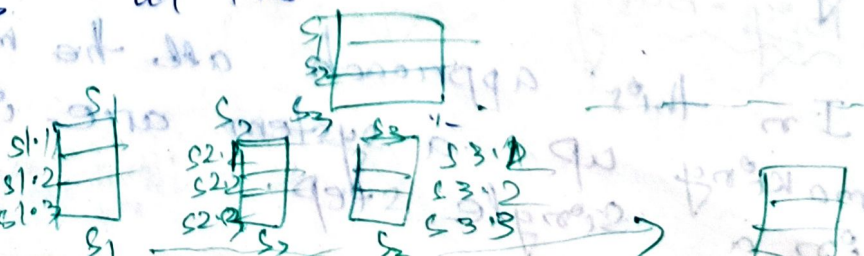
In this approach all the modules making up a system are integrated in a single step.

- The main problem with this approach to identify an error if detected.
- So it is very expensive approach and not useful for large program

m1 – m10 → together at a time.

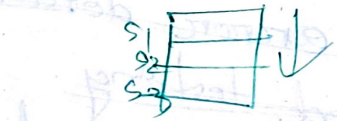
(i) Bottom-up Integration testing

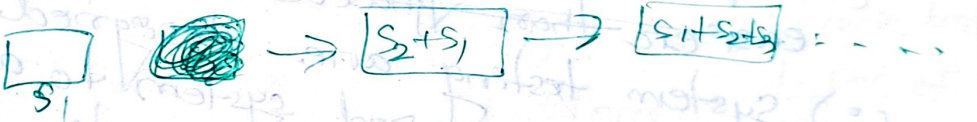
- Generally large softwares are divided into no. of sub systems.
- A sub system might consists of several modules which are communicated with each other.
- In this testing 1st the modules each subsystem are integrated.
- Advantage of this testing is that different sub systems can be tested simultaneously.
- Another advantage is that low level modules get tested throughly
- Disadvantage is that if a system consist of large no. of sub systems at the same level.



(iii) Top-Down Integration testing

- This testing starts with root module and one or more submodules in the system
- After root module is tested the immediate low layer are combined with root module and again tested.
- Advantage is that it requires comfortably only one stubs and stubs are simpler to create as compared to drivers.
- Disadvantage is that in the absence of low level functions the desired input, output operations can't be performed.





(iv) Mixed Integration testing

- It follows combination of top-down and bottom up approach.
- In top-down approach testing can start only after the top level modules have been coded and unit tested
- In bottom-up approach testing can start only after bottom level modules are ready
- In mixed approach testing can start as and when modules become available after unit testing.

• In this testing both stubs and drivers are reqd to design.

DT:-31.10.16

phased ~~vers~~ vs Incremental integration testing

• In incremental testing only two module is added to the partially integrated system each time.

• In phase testing a group of related module are added to the partial system each time.

• In incremental testing error detection is easier than phased testing.

System testing

~~(i) System testing error detection is easier than phased testing~~

(i) System testing are designed to valid a fully developed system to asure that it meets its requirements.

(ii) The system test cases are based upon SRS documents.

(iii) There are 3 types of system testing

* Alpha testing (α)

It is based upon a testing procedure where test 'T' within the developing organisation test the final system.

• The system test cases are classified into functional and performance test cases;

DT:- 31.10.16

- The system test cases are classified into functional and performance test cases.

(a) Performance (black box) testing

Performance testing is carried out to check whether the system meets the non functional requirements identified in the SRS document.

All performance test cases are black box test cases. There are different types of performance testing.

(i) Stress testing

- Stress testing is also known as endurance testing.
- This testing evaluates system performance when it is stressed for short periods of time.
- Ex:- Suppose an operating system at a time support 15 operations simultaneously then stress testing at a time tests all the 15 operations.

(ii) volume testing

- Volume testing cheaks the data structures "stack, queue, array, buffer etc" have been designed successfully, or not.
- It is used to cheack system is performing in extraordinary situationsor not.

(iii) configuration testing

configuration testing is used to test system behaviour in various hardware and software configurations specifying the requirements.

(iv) compactability testing

This type of testing is reqd when the system interfaces with external system.

(v) Regression testing

This type of testing is require when the system being tested is an upgradation of an already existing system, to enhance functionality, performance.

(vi) Recovery testing

Recovery testing tests the response of the system to the presence

of faults or loss of power services.

(vii) Maintenance testing

This testing is used to test system is performing correctly or not.

(ix) Documentation testing

It is used to check the required user manual, technical manual and maintainance manual exists or not

(x) Useability testing

This testing is used to test user interface requirements like display screens, message box, reports, etc

(xi) Security testing

Security testing is essential for software to process confidential data.

(b) Error seeding

- Error seeding has the name implies seeds the codes with some known errors or other words some artificial errors introduced.
- It is used to predict the following aspects of the system.
  (i) The no. of errors remaining in the product.

- The effective ness of testing strategies
- Mathematically it is expressed as follows
  let 'N' be total no. of defects in the system and let 'n' of these defects to be found by testing.
- Let 'S' be total no. of seeded efforts and let 's' of these defects be found during testing. Therefore $n/N = s/S$

$$\Rightarrow N = S * \frac{n}{s}$$

$$\Rightarrow N - n = n \times \left(\frac{(S-1)}{S} \right)$$

Regression testing

DT:- 01.11.16

(i) It doesn't belong to either unit, system or integration testing.

(ii) It is a separate testing method where an old test suite used to test the system after the system introducing new type of errors.

```
even.c.
char x;
```

(1.0)
(2.0)

chapter

Software maintainance

Different causes of maintainance

(i) corrective maintainance.
(ii) Adaptive maintainance.
(iii) perfective maintainance.

characteristic of software evolution

The scientist Lehman introduced three

different laws to characterised software evolution.

(a) Lahman's 1st law

Good products are maintained and bad products are thrown away.

b) Lahman's 2nd law

If a product is structured well then less maintainance reqd. Otherwise more maintainance reqd.

ex:- bike.

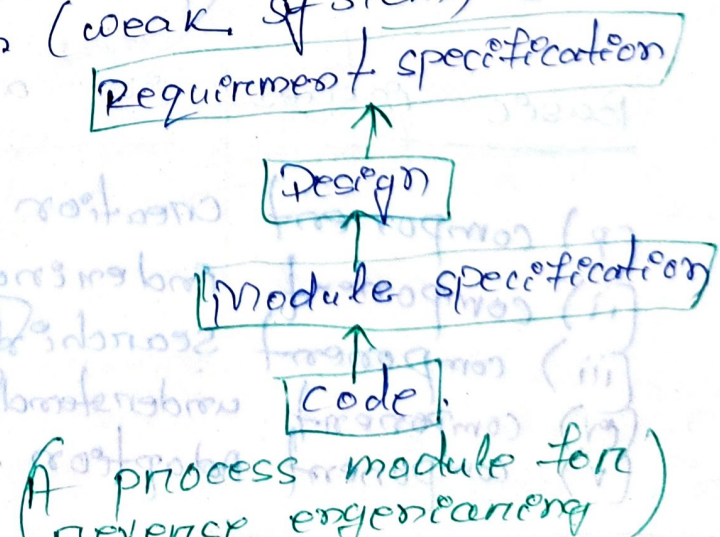
(c) Lahman's 3rd law

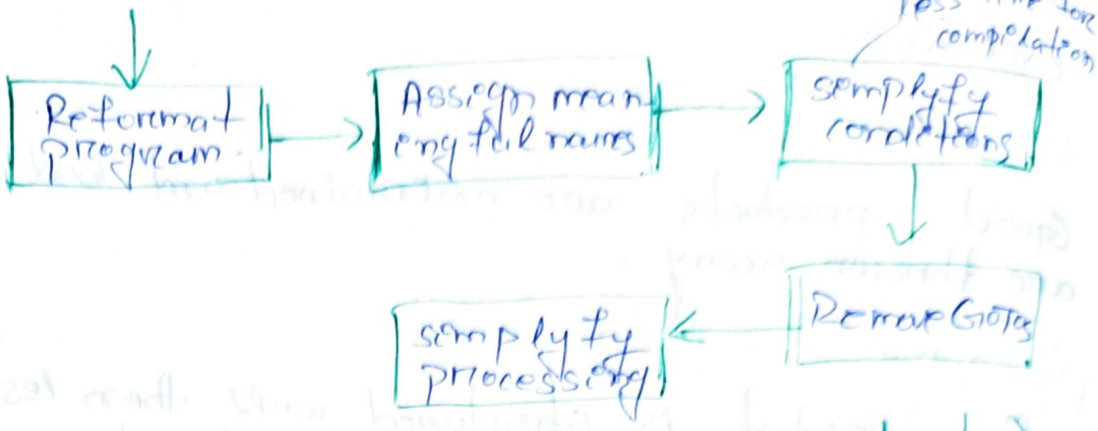
The rate at which code is written and modified is approximately same during product development and maintainance.

VIV sem IMP

Reverse Engeneering

- Software reverse engeneering is the process of recovery the design and the requirment specification of a product from an analysis of its code.
- The purpose of reverse engeneering is to facilate maintainance work by improving the under standability of a system and to produce necessary documents for a system (weak system)





(Cosmetic changes carried out before reverse engineering)

chapter

Software Reuse

- Software reuse technique is used to reduce development cost and time.
- Different parts of software can be reused.
  - (i) SRS.
  - (ii) Design.
  - (iii) Programming code.
  - (iv) Test cases.
  - (v) Knowledge.

Basic issues in any reuse program

(i) component creation.
(ii) component indexing and storing.
(iii) component searching.
(iv) component understanding.
(v) component Adaptation.

DT: 02.11.16

Repository maintainance

Repository maintainance means continuous maintainance.

Reuse approach

Domain analysis

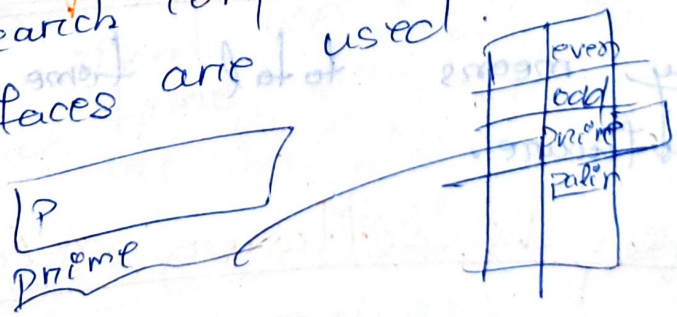
Domain analysis is used to reuse components of software products.
banking, Railway, hospital.

component classification

- It is very vital to classify components and give effective indexing and storing
- Relational Data base management system (RDBMS), object oriented (OODBMS) are used for storing components
- To classify components Prieto-Diaz's classification scheme used where component data structure, system that they are part of Discussed.

Searching

To search component different weblink interfaces are used.



Repository maintainance

Reuse without modification

Reuse at organisation level

- Acess an item depending upon its potential.
- Refigm the item for greater reusibility
- Enter the product into repository.
- check functionality of the product.

Refiguring product for better reusability

(a) product name should be generalised
eventhough name already---

(b) Product operation should be generalised.

(c) exception should be generalised.

(d) Product portability problems.

→ This involve to handel several types of exception in a program.

Chapter
SOFTWARE RELIABILITY

Reliability means total time period to run a software.

Hardware Vs software reliability.

- Hardware component failure due to software components whereas software components fails due to various bugs (failure / error)

Reliability metrics or Reliability measures

(i) ROCOF (Rate of occurrence of failure)

- It measures total no. of failures observe and duration of failure

(ii) Mean time to failure (MTTF)

- It measures between two successive failures the time gap.
- Let n = Total no. of failures.
  $t_1, t_2 \ldots t_n$ = failure at the time gap

$$MTTF = \sum_{i=1}^{n} \frac{t_{i+1} - t_i}{n-1}$$

(iii) Mean time to Repair

- Total time requen to repair the error

(iv) Mean time between Failure (MTBF)

$$MTBF = MTTF + MTTR$$

(v) Availability

- This says how long a system ava lable for use, it also account repair time of any failure is endica d.

classification of failure.

(i) Transient Failure (log10, logaXfail

certain input values when calling a function.

(ii) Permanent

All input values

(iii) Recoverable

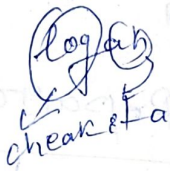
When failure occurres data can recover without shutdown and start up.

(iv) Un Recoverable

Need shut down and start up the system.

(v) Cosmetic changes

Need reformatting of a program



DT:- 04.11.16

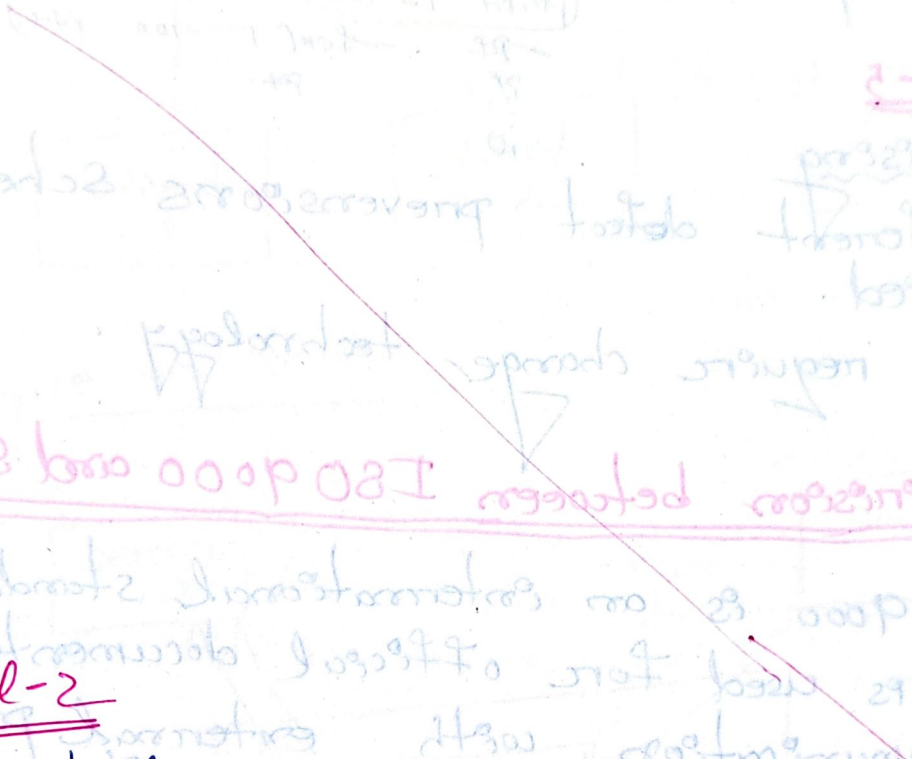
Reliability growth modeling

- It is used to determine when to stop testing to attemt reliability level.
- There are 2 models used to test reliability growth these models are.
  (i) Jelinski and Moranda Model.
  (ii) Zettle wood and Verall's Model.
- Software engineering integrated SEI CMM (capability maturity model).

- It is used to produce a good quality software from scratch level to maturity level.

Level-1:-

Initially



Level-2

Repeatable

- Here different software project planning made but Software configuration management there is no documentation work.
- All the cost estimations are made.

Level-3

Defined

- In this level documentation of management and development activities are all work introduced.
- Training for the developers and work reviews are made.

Level-4

Managed

- Differrent matricse are used forr process and product.
- It is use to cheak quality of the software product.

(print 1-100)

pf — for(1 100 i++)

pf pf

i0

Level-5

optimising

- Different defect prevensions scheme applied.
- If require change technology.

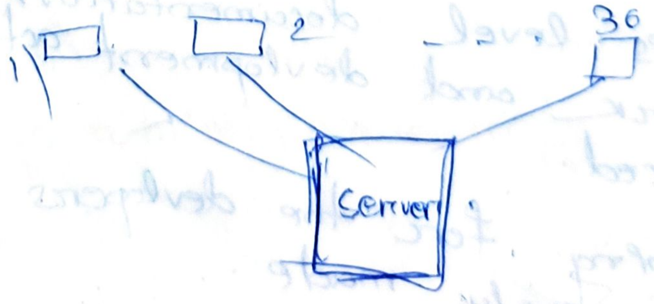
comparrission between ISO 9000 and SEICmm

- ISO9000 is an international standard.
- It is used forr officcal documents.
- communication with external parties where SEICmm is totally softwarre based it is only used by softwarre industry.

chapter

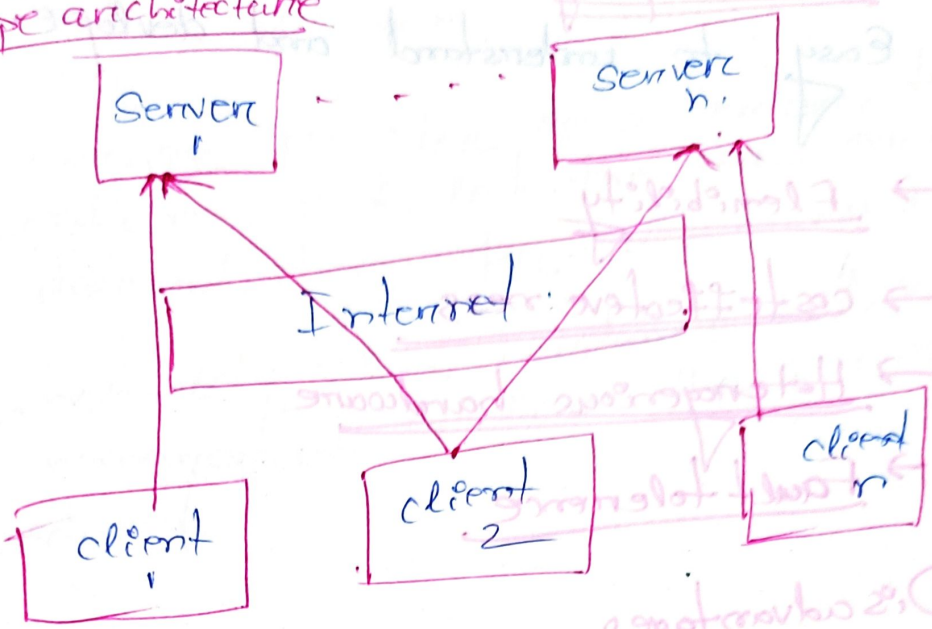
client server Softwarre Engeneering

client server architectures

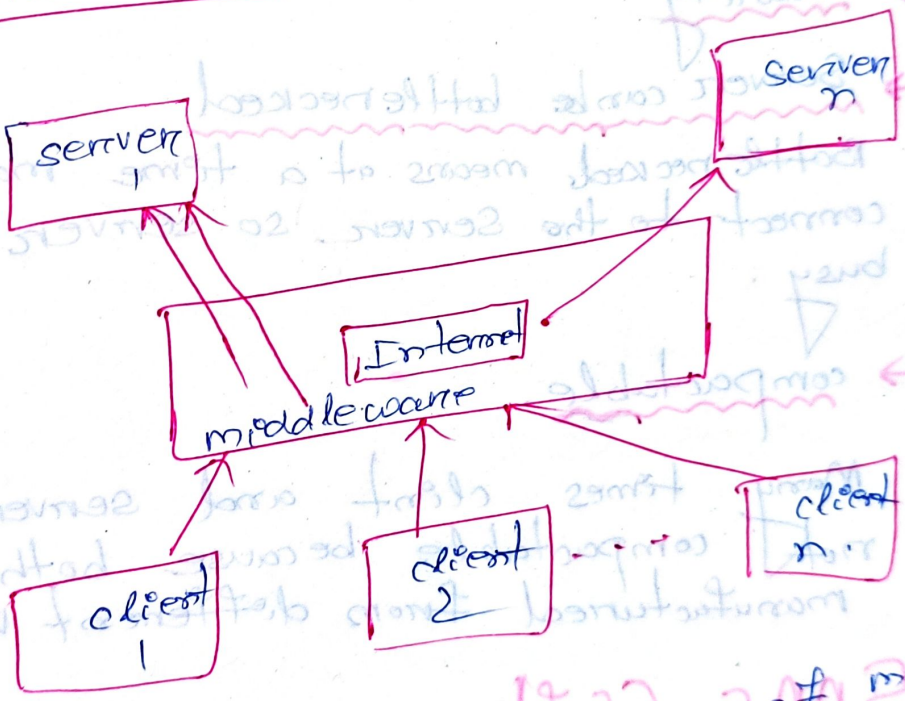


• client and server are two type architecture.

2 type architecture



3 type architecture



There are different types of middle ware the standard middle wares is CORBA (common object request broker Architecture.

Advantages of client server software

→ concurrency processing is faster.

→ Loose coupling
Easy to understand and devlope.

→ Flexibility

→ cost effectiveness

→ Heterogenious hardware

→ Fault tolerence

Disadvantages

→ Security

→ Server can be bottlenecked
Bottlenecked means at a time many client connect to the server. so server become busy.

→ compactable.
Many times client and server are not compactable because both are manufactured from different venders.

SAAS (software as a service)

- SAAS is a software delevery model and involves customers to pay for any software per unit time of use. with the price reflecting market place supply and demand.
- SAAS is a software or service provider

- Software vendor provides maintainance daily technical maintainance and support for the software.

The service provider is a vendors to host the software and let the users activate on demand charges for use to need.

Responsibility for hardware and software management is from the service provider

- RMI corba

Software maintenance

software maintenance denotes any changes made to a software product after it has been delivered to customer.

* characteristics of software maintenance

- products need to run on newer platform, newer environment and enhanced features.
- when support environment changes.

→ Types of software maintenance

- corrective: rectify the bugs observed while the system is in use.
- Adaptive: when customer need the product to run on new OS, platforms or interface with new hardware or software.
- perfective: change functionality, add new features as per customer demand enhance performance.

→ characteristics of software Evolution (Generalized) by Lehman and Belady

Lehman's first law

A software product must change continually or become progressive less useful.

Lehman's 2nd law

The structure of program tends to degrade as more and more maintenance is carried out.

3rd law

over a program's lifetime its rate of development is approximately constant.

→ problem associated with maintenance

Book

- Necessary to understand other's work
- majority are legacy systems

Software reverse Engineering