

Software Engineering

Software :- It is a set of executable programming codes which serves some computational purpose.

Engineering :- It is all about developing products, using well-defined, scientific principles & methods.

→ Software, when made for a specific requirement is called software product.

"Software Engineering" is an engineering branch associated with development of software principles, methods & procedures.

Software + Engineering

Characteristics of good software:-

Operational

- Budget
- Usability
- Dependability
- Security
- Safety

Transitional

- Portability
- Reusability
- Adaptability

Maintenance

- Modularity
- Maintainability
- Flexibility
- Scalability

Types of software:-

(i) System software

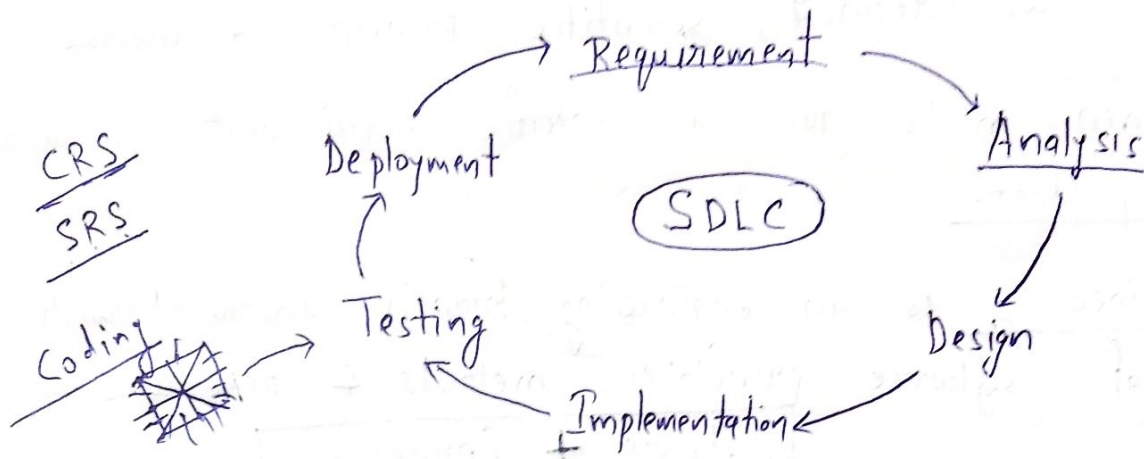
- OS
- compiler
- Assembler
- Linker etc.

(ii) Application software

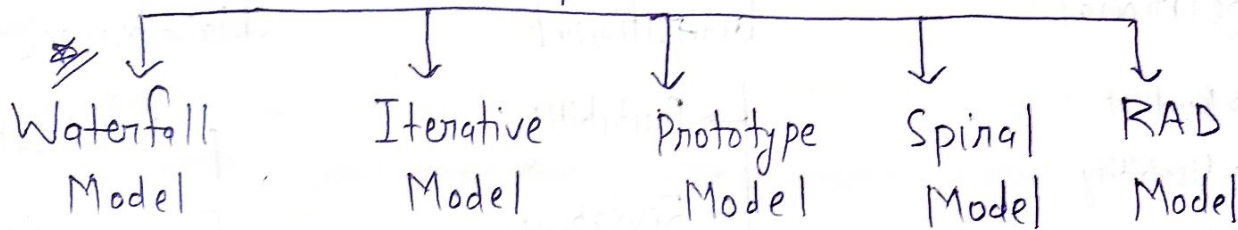
- email
- Games
- Apps
- Database

What is SDLC?

- SDLC is a process used to produce a high quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.
- It is acronym of Software development life Cycle.

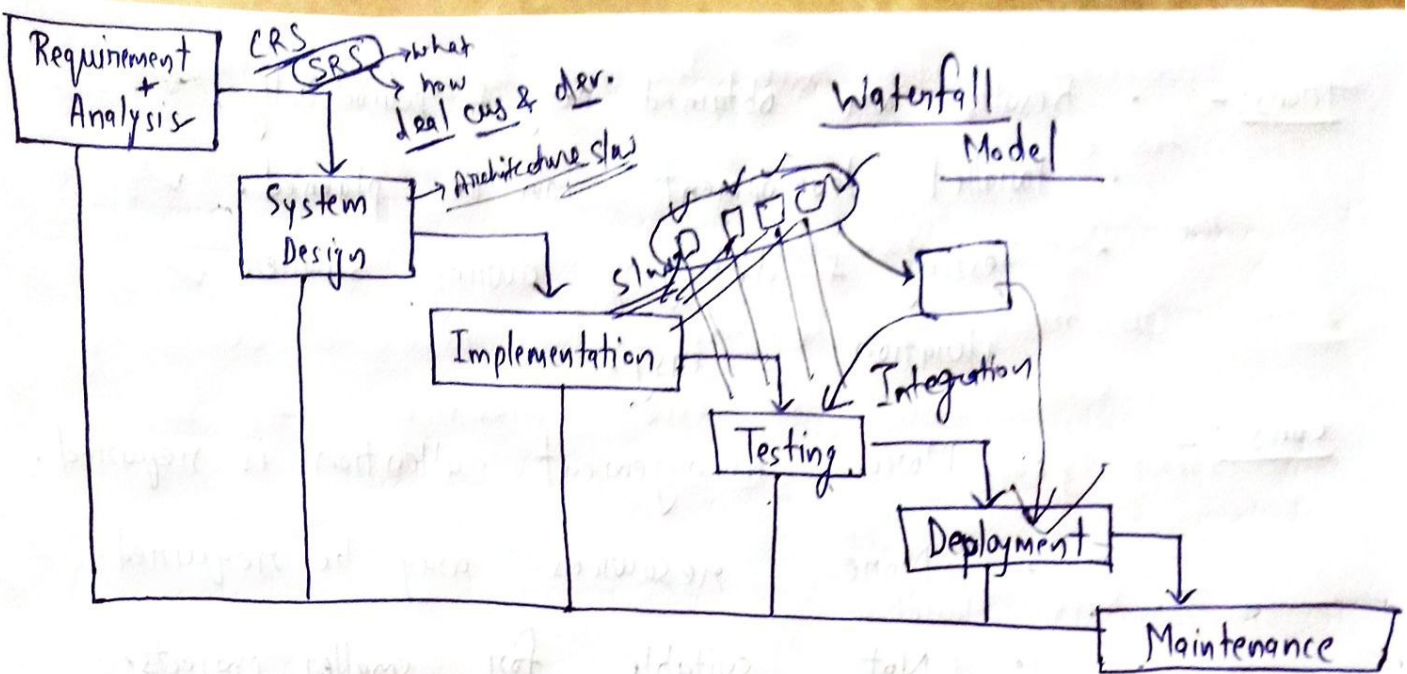


SDLC Models



Waterfall / Linear Sequential Model :-

- It was the first process model to be introduced.
- In this model, each phase must be completed before the next phase can begin & there is no overlapping in the phases.
- Typically, the outcome of one phase acts as the input for next phase sequentially.

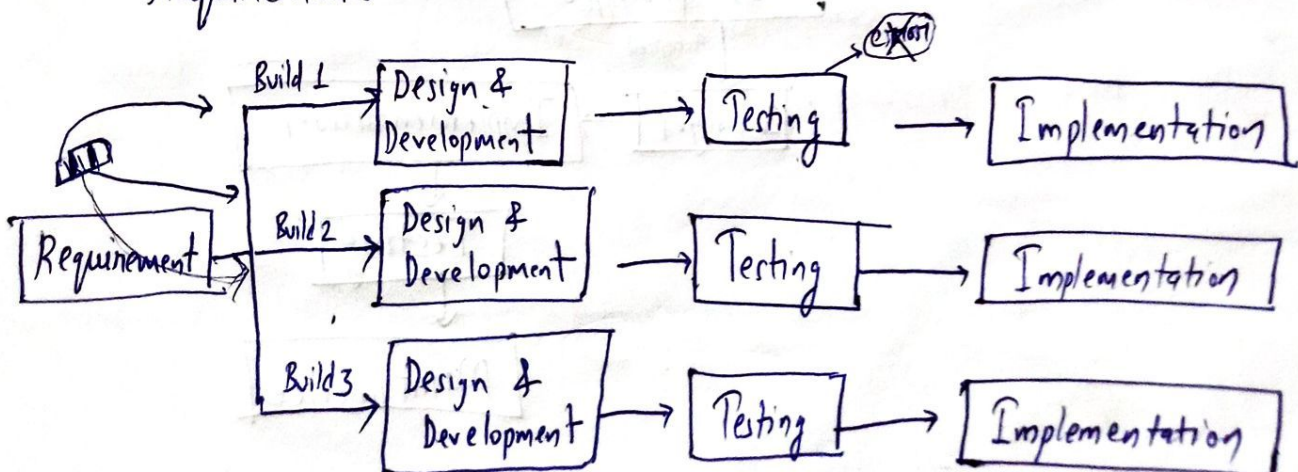


- Adv.:-
- Simple & easy to understand & use.
 - Phases are processed & completed one at a time.
 - It allows for departmentalization & control.

- Disadv.:-
- No working slw is produced until late.
 - High amounts of risk & uncertainty.
 - Poor model for long & ongoing projects.

Iterative Model :-

- Iterative process starts with a simple implementation of a small set of slw requirements & iteratively enhances the evolving versions until system is ready to be deployed.
- It does not attempt to start with full specification of requirements.



Pros:-

- Results are obtained early & periodically. ✓
- Parallel development can be planned. ✓
- Testing & debugging during smaller iteration is easy. ✓

cons:-

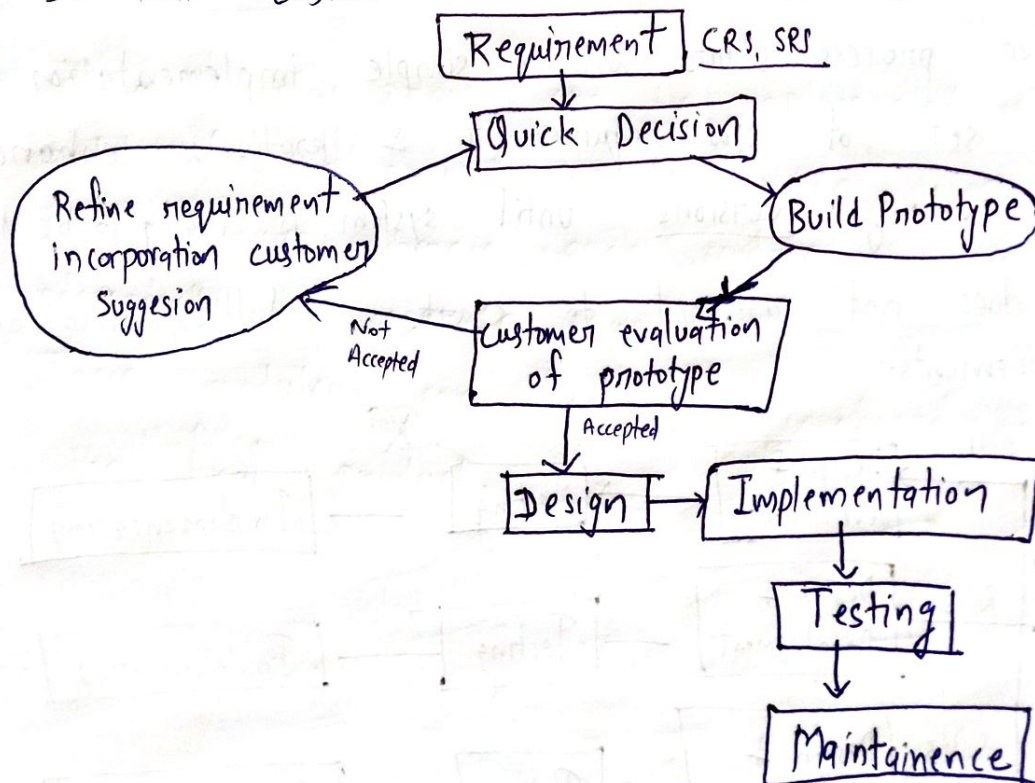
- More management attention is required.
- More resources may be required.
- Not suitable for smaller projects.

Prototype SDLC Model:-



The prototype model requires that before carrying out the development of actual sw, a working prototype of the system should be built.

→ Prototype is used to give an idea to the customer that how our system will look like & work. So that customer can take more requirements

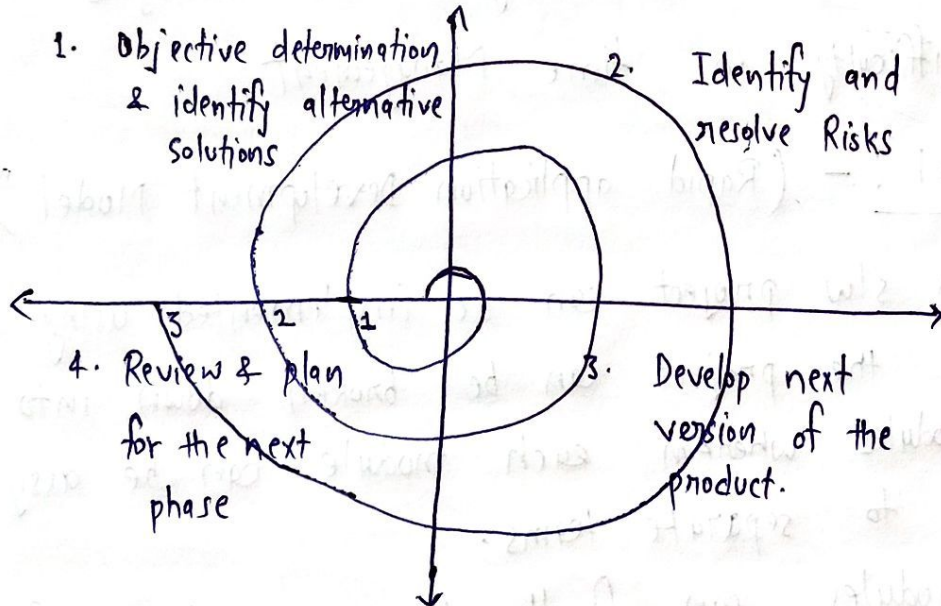


- Adv:-
- Reduce the risk of incorrect user requirement.
 - Good where requirements are changing.
 - Support early product marketing.

- Disad :-
- Costs customer money.
 - Needs committed customer.
 - Difficult to know how long project will last.
 - Prototyping Tools are expensive.

Spiral SDLC Model:- It is a combination of iterative development process and waterfall model.

- It allows incremental releases of the product.
- It provides support for Risk Handling.



1. → Requirements are gathered and objectives are identified, elaborated & analyzed.
2. → All possible solutions are evaluated to select best possible solution. Then risks associated with solutions are identified. and risks are resolved using best possible strategy.

3. → Identified features are developed & verified through testing. And in end, the next version of the sw is available.

4. → Customers evaluate the so far developed version of sw. In the end, planning for the next phase is started.

Adv. :-

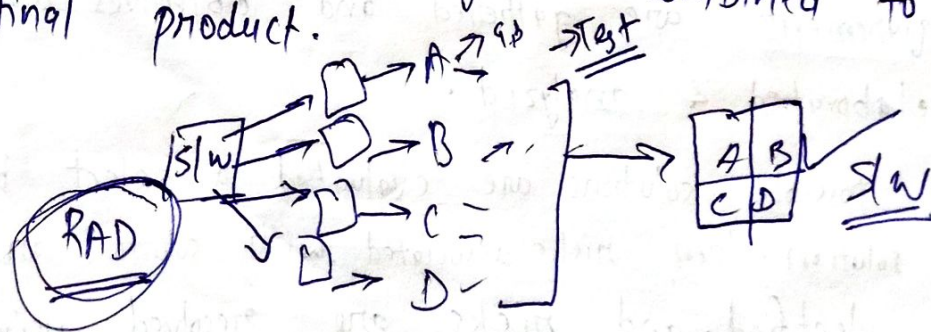
- Risk Handling
- Good for Large projects
- Flexibility in requirements
- Customer Satisfaction

Dis :-

- Complex
- Expensive
- Too much dependability on Risk Analysis
- Difficulty in time Management

RAD Model :- (Rapid application Development Model)

→ ~~A~~ is A sw project can be implemented using this model if the project can be broken down into small modules wherein each module can be assigned independently to separate teams. These modules can finally be combined to form the final product.



Requirements
Planning

User Description

Construction

Cutover

- It involves use of various tech used in req. elicitation like brainstorming, task analysis, FAST etc.
- It also consists of entire structured plan describing the critical data, methods.

- Taking user feedback & building prototype using developer tools.
- It includes re-examination & validation of data

- Refinement of prototype & delivery takes place.
- All the required modifications & enhancements are done.

- All interfaces b/w independent modules developed by separate teams have to be tested properly.

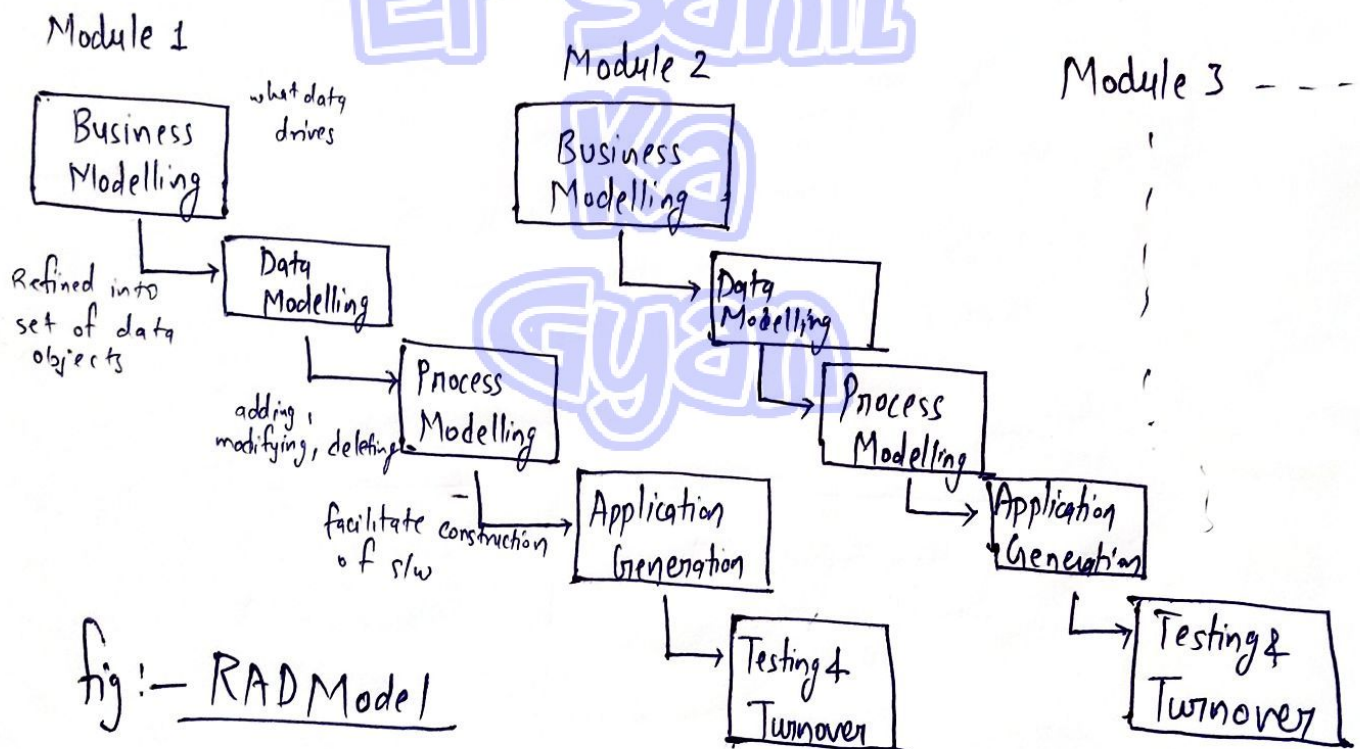


fig:- RAD Model

- Adv.:-
- flexibility
 - Adoptable
 - Reusability
 - Reduced development time

SDLC

Disad:-

- All application is not compatible
- Can't use in small projects
- High tech. risk

- Verification \Rightarrow It answers the question.
Are we building the system right?
- \rightarrow It is the process of evaluating a system or component to determine if the product of a development phase satisfies the conditions imposed at start of that phase.
 - \rightarrow It is applied during the development phases.
 - \rightarrow Verification includes manual testing.

Manual testing - look and review the documents.

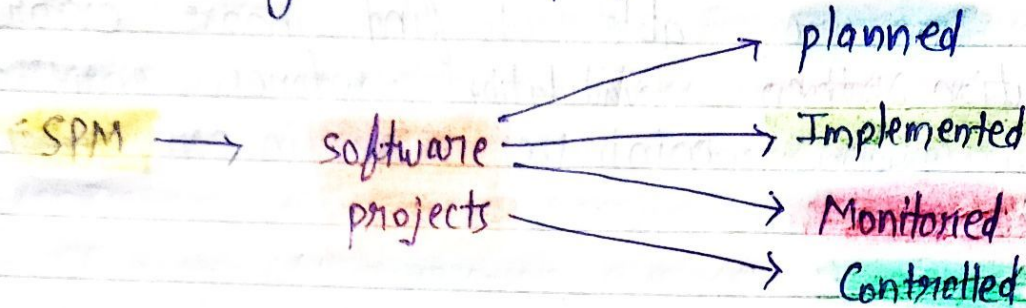
Activities involved reviews, meetings & inspection. Verification is carried out by internal quality assessment team. Verification is done prior to validation.

- Validation \Rightarrow It is the process of evaluating the system or component at the end of development process to determine whether it satisfies the specified req.
- \rightarrow Applied at the end of development process.
 - \rightarrow Program ~~exec~~ execution is done to validate the requirements.

- Activities involved black box testing & white box testing.
- Validation is done after verification
- Both validation & verification are complementary activities.
- If we are able to find more errors before execution then validation becomes easier.
- Verification minimizes error in early phases of development.

Software Project Management.

SPM is the art and science of planning & leading sw projects.

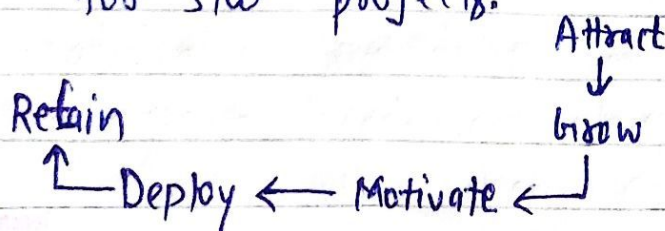


Effective software project management focuses on the 4 P's \Rightarrow



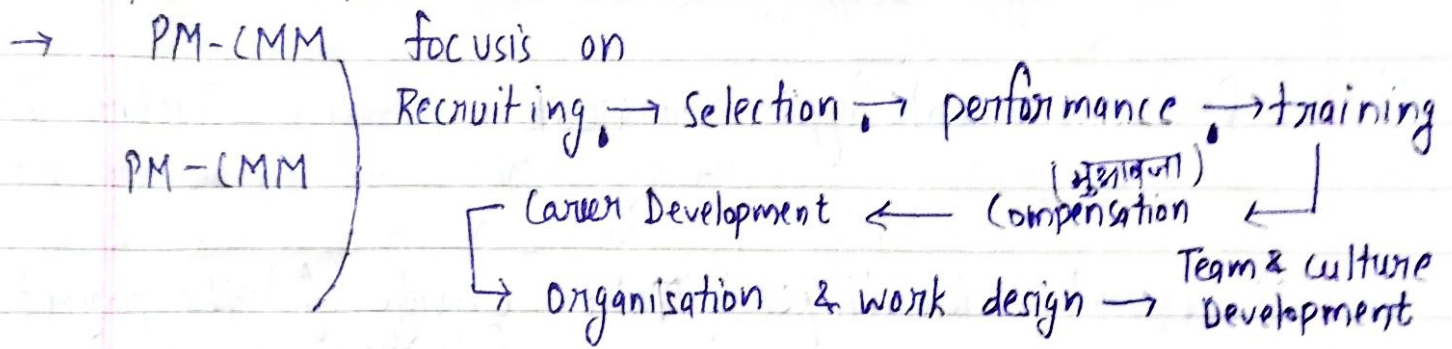
(A) **PEOPLE** \Rightarrow Include team members (Head, manager, team-leader, developer, team, testing-team, & owner/borrower). It deals with the cultivation of motivated and highly skilled people. S/w Engineering institute has developed of people management capability, ~~model~~ maturity model (PM-CMM) \Rightarrow

\rightarrow This model helps to build a right development team for sw projects.



Teacher's Signature.....

~~PM - CMM~~



People consists of (i) Stake Holder ~~and~~ (Senior managers, project manager, practitioner, customer and ~~end user~~ end user)

(ii) The team leaders :- Right skills & experience.

(iii) Software team :- Coder, tester, manager.

→ Five categories of stake holder :-

(a) Senior manager :- Define business issue that often has significant influence

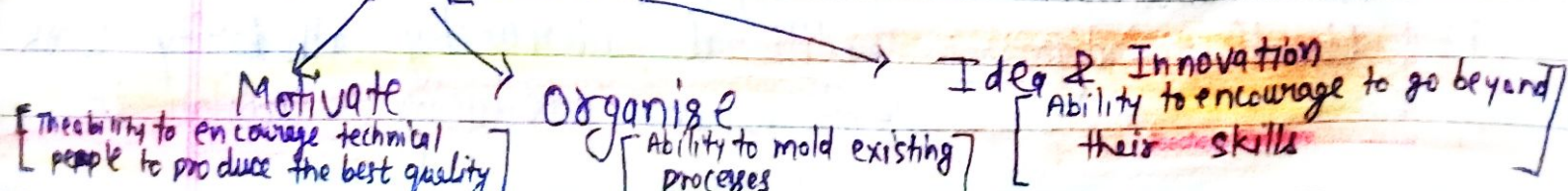
(b) Project manager :- They plan, motivate, organise and control the practitioner who do the work.

(c) Practitioner :- Deliver the technical skills that are necessary to engineer of product or application

(d) Customers :- Specify the requirements for the s/w to be engineered.

(e) End User :- Interact with the s/w once it is released for use.

→ team leaders :-



Subset of useful leadership ~~read~~ Traits ^(निशान) ⇒

(i) Problem solving :- Diagnose a problem, structure a solution, apply lessons learnt, remain flexible

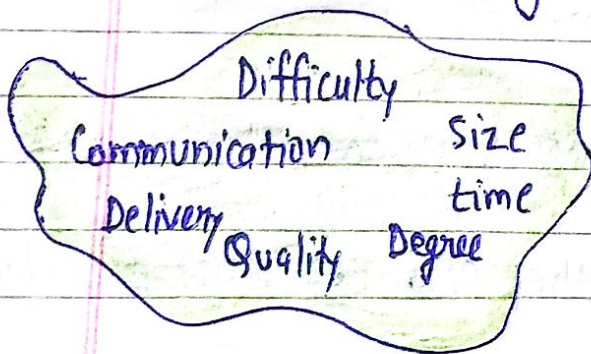
(ii) Managerial identity :- Make change of the project at confidence to assume control, have assurance to allow good people to do their job.

(iii) Achievement :- Reward initiating demonstrate that ~~this~~ ^{risk} ~~packing~~ ^{taking} will not be punished.

(iv) Influence and team building :- Be able to read people, understand verbal & non-verbal signals, able to react to signals, remain under control in high stress situation.

(v) Software team ⇒

7 project factors to be considered when structuring a software development team



4 organisational paradigm for software development Team

(i) Closed paradigm :- Traditional hierarchy authority work well when producing similar to

past ~~efforts~~ ^{efforts}, members are less likely to be innovating.

(ii) Random Paradigm:— Depends on individual initiating of team member, work well for project required in innovation on technological break.

(iii) Open Paradigm:— Hybrid of the closed and random paradigm. Works well for solving complex problems requiring ~~collaboration~~ and communication among team member. Collaboration

(iv) Synchronous paradigm:— Organises team members based on the natural ~~pea~~ pieces of the problem.

(B) Product \Rightarrow Thus the scope of the s/w development must be ~~ext~~ established and bounded:—

(i) Context:— How does the s/w to be built fit ^(संज्ञा) into a largest system, product or business context. And what ~~context~~ constraints are imposed as a result of a context. (संज्ञा)

(ii) Information objective:— ~~for~~ What customer visible data objects are produced as output from the s/w? and what data objects are required for input?

(iii) Function & performance:— What function does the s/w perform to transform i/p

data into performance output? Are there any special characteristics to be addressed?

→ slw project scope must be unambiguous and understandable at both the managerial & technical level.

(c) **The Process** ⇒ The project manager must decide which process model is most appropriate based on the customers who have requested ~~requested~~ the product and the people who will do the work.

(ii) The characteristic of product itself.

(iii) The project environment in which slw team works.

(D) **PROJECT** ⇒ Planning and controlling a slw project is done for one primary reason, it is the only way to manage the complexity.

W⁵HH principal ⇒ A series of questions they lead to a definition of key project characteristic and the result in project plan.

Q.1 Why is the system being developed?

Ans - Defines the validity of business reasons and justification.

Q.2 What will be done? Ans - establishes the task set require for the project.

Q.3 When will be done?

Ans- Establishes the project schedule.

Q.4 Where are they organisationally located?

Ans- Not the organisational location of team member, customers and other stock holder.

Q.5 Who is responsible for function?

Ans- Define the role and responsibility of each team member.

Q.6 How will the job be done technically & managerially

Ans- Establishes the management and technical strategy.

Q.7 How much of each resource is needed?

Ans- Establishes estimates base on the answer for the previous questions.

PROJECT PLANNING PROCESS: —

(A) Objective: — The objective of slw project planning is to provided framework that enable the manager to held make ~~regi~~ reasonable the estimate of resource, cost and schedule. The plan must be adopted and updated as the ~~project~~ proceeds.

(B) Task set for project planning: —

(i) Establish project scope.

(ii) Determine feasibility

- (iii) Analysis Risk
- (iv) Define require resources
 - (a) Determine human resources required
 - (b) Define reusable software resources
 - (c) Identify environmental resources
- (v) Estimate cost and efforts
 - (a) Decompose the problem
 - (b) Develop to more estimate using size, function point, process task or use cases.
- (vi) Develop a project schedule.
 - (a) Establish a meaningful task set.
 - (b) Define the task network
 - (c) Use scheduling tool to develop a time-line chart.

RESOURCES ⇒

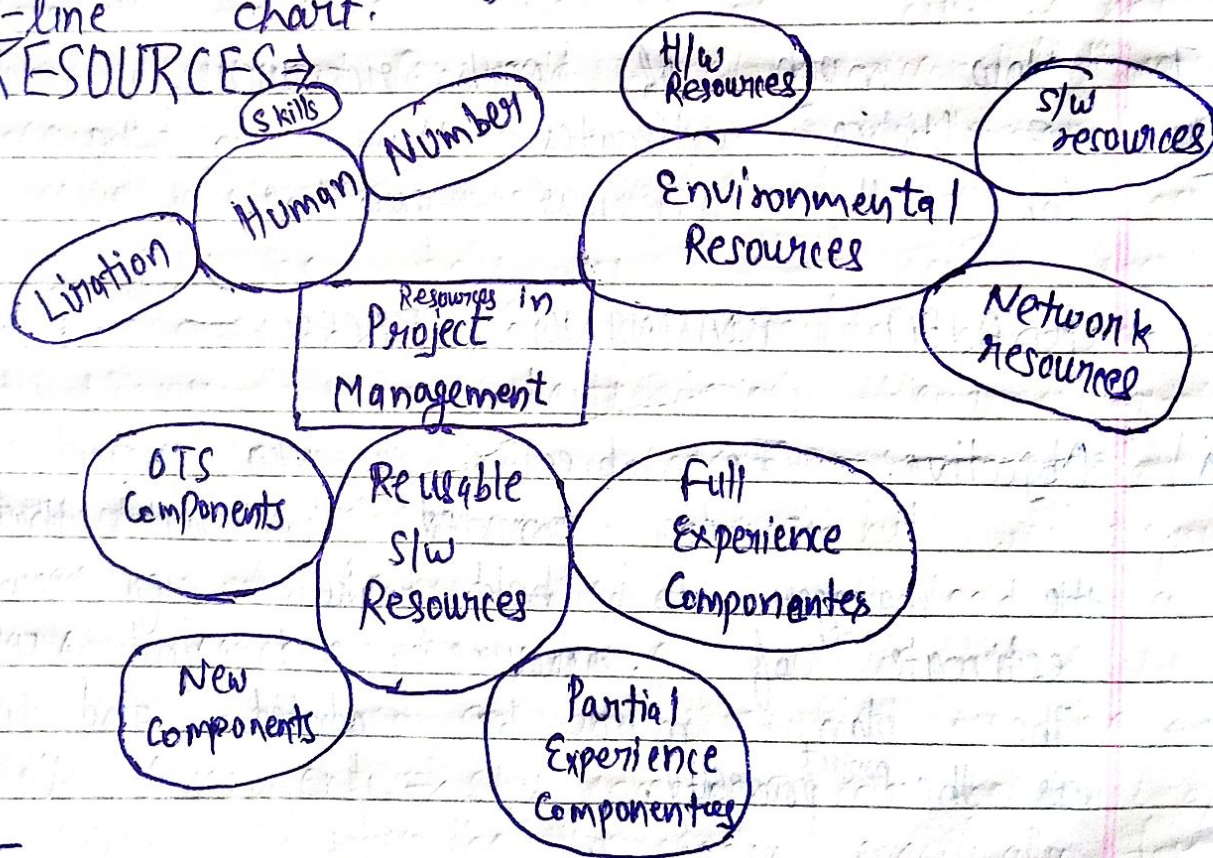


Fig: -

3 Major categories of slw engineering resources

The second planning task is estimation of the resources required to accomplish the SW development process.

Each resource has 4 characteristics :-

- (i) Description of the resource
- (ii) Statement of availability
- (iii) Time when the resource will be required (Time window)
- (iv) Duration of time that resources will be applied.

* Availability of the resource for a specify time window must be established at the earlier times.

- (i) Human Resources \Rightarrow For relatively small project a single individual may perform all SW engineering task, consulting with specialist require. For larger project the SW team may be geographically dispersed. ~~which Across Across~~ the number of diff. location, hence the location of each human resource is specified.

The no. of people required for a SW project can be determined only after an estimate of development effort. (eg- Person-months) is made.

- (ii) Reusable SW Resources \Rightarrow Component based SW engineering focus on reusability that is the creation & reuse of SW building blocks. Such building blocks ~~after~~ called often.

as component.

Types of Reusable S/W Components:—

(i) OFF-The-shelf (OTS) components \Rightarrow

Existing S/W can be acquired from a third party all has been developed internally for Past project. Commercial OTS components are purchased from a third party and are ready for use on the ~~current~~ current project. They have been fully validated.

(ii) Full Experience components \Rightarrow Existing specification, designs, code, or test data developed for past projects are similar to the S/W to be built for the current project. Therefore modification required for full exp. components will be relatively low risk.

(iii) ^{Experience} Partial Experience Components \Rightarrow Existing specification, design, code & test data developed for past projects are related to the S/W to be built for the current project but will require some modification. And it will lead to a fair degree of risk.

(iv) NEW components \Rightarrow S/W component must be built by the S/W team specifically for the needs for the current project.

3. Environmental Resources \Rightarrow The environmental that support a SW project is often called the SW engineering environment. In corporates HW & SW, HW provides a platform that supports the tool (SW tools) required to produce the work product. That are an output come of good SW engineering practice. A project planner must prescribe the time window required for HW & SW and verified that the resources will be available.

Software Metric \Rightarrow Software metric is the major of SW characteristics which are quantifiable and countable. SW metrics are important for measuring SW performance, planning work items, measuring productivity and estimating cost.

- \rightarrow Basic quality and productivity data are collected, this data are analysed, compare against past averages.
- \rightarrow The goal is to determined whether the quality & productivity improvement have occurred. The data can also be used to point out problem areas.
- \rightarrow Remedial Remedies can be ~~then~~ ~~also~~ developed and SW process can be improved.

Uses of measurement (SW) \Rightarrow

- (i) Can be applied to the SW process with the intent of improving it on the continuous bases.

- (ii) Can be used through out a slw project to assist in estimation, quality control, productivity assistance and project control.
- (iii) Can be used to help ^{assess} the quality of slw work products and to assist in ~~tech~~ tactical decision making as the project proceed.
- (iv) ~~Technical decision~~ → process making as the project proceed.

Type of slw metrics

(i) ~~Product~~ product metric (ii) process (iii) Project

(i) Product Metric ⇒ Describe the characteristic of the project such as size, complexity, design features, performance & quality level.

(ii) Process Metric ⇒ Used to improve slw development & maintenance. Process metric are collected across all project and over long period of time. The only way to ~~know~~ how & where to improve any process is to :

- (i) Major specific attribute of the project
- (ii) Development a set of meaning full metric based on these attributes.

The set of metrics are :-

- (i) Error uncovered before release of slw
- (ii) Defects delivered and reported by the end users.
- (iii) Work product delivered
- (iv) Human effort expended
- (v) Calendar time expended
- (vi) Time & effort to complete each activity.

(iii) Project Metric \Rightarrow Describe the project characteristic & execution. (i) project metric include the no. of s/w developers (ii) the staffing pattern over the SDLC

(iii) Cost (iv) Schedule (v) Productivity

Use of project metric:-

- (i) The first application of project metric occurs during estimation.
- (ii) Metric from past projects are used as basis of estimating time & efforts.
- (iii) Used to minimise the development schedule by making the adjustment necessary to avoid delay.
- (iv) Assess the product quality on ongoing basis and when necessary & modify the technical approach to improve the quality.

(s/w size estimation)

LOC (line of code) \Rightarrow It includes declaration & actual code including logic & computation.

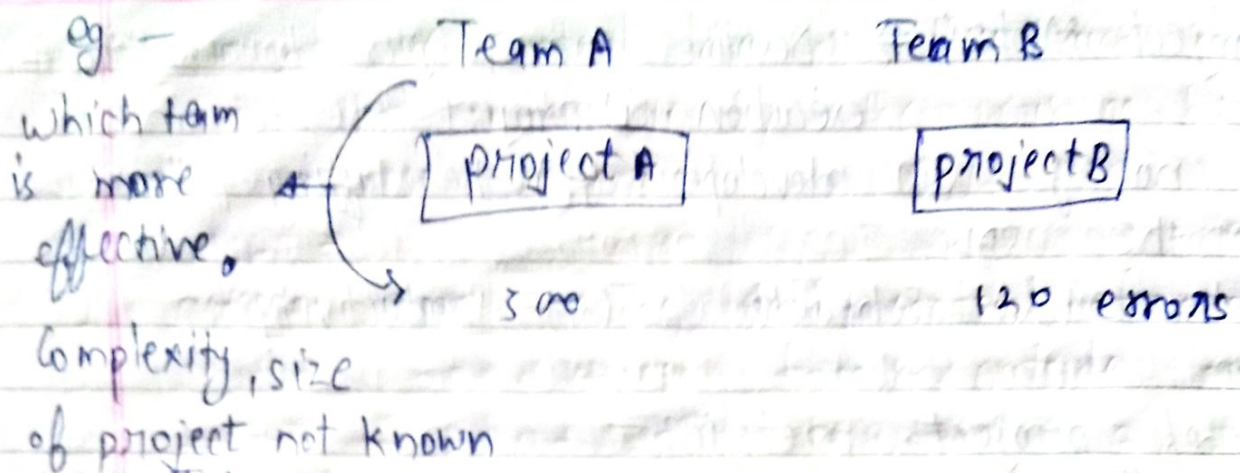
What is ~~known~~^{not} the LOC?

- (i) Blank lines:- Included to improve the readability.
- (ii) Comments:- Included to help in code understanding as well as during maintenance.

~~(iii)~~

This is not a LOC because (i) do not contribute to any kind of functionality.

- (ii) Can be misused by developers to give a false notion about productivity.



This question can not be answered because the size or complexity of project is not known

Advantage of LOC for size estimation : (Loc)

(i) It is very easy to count and calculate from the developed code.

Disadvantage of using LOC : —

eg -

```

for(i=0; i<10; i++)
    cout << i;
    LOC = 2
    
```

```

for(i=0; i<10; i++)
{
    cout << i;
}
    LOC = 4
    
```

- (i) LOC is language plus technology dependent.
- (ii) What constitutes an LOC varies from organisation to organisation.

Project name	EFFORT	Page per documentation	Errors/line		Defects	\$
Alpha	64	1120	350	8	60	120
Beta	82	950	120	5	20	200
Gamma	42	600	250	7	50	215

An organisation maintains table for size oriented metrics as shown in the figure.

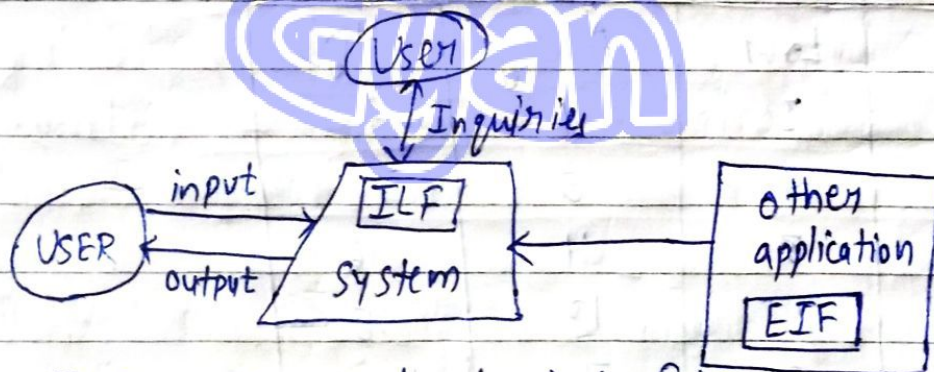
To develop metric that can be compared with similar metric from other projects we choose like of code our normalization value. Now from a data contain in the table a set of simple size oriented metric can be developed for each project.

KLOC \Rightarrow Thousand of LOC.

Errors per KLOC, defects per KLOC, dollar per KLOC, page of documentation per KLOC, in addition other interesting metric can be computed such as errors per person month, KLOC per person month, \$ per page of documentation.

Function Point (FP): — It measures functionality from users point of view.

What the user receive from the software and what the user requests from software.
It focus on what functionality is being delivered?



ILF: Internal logical files

EIF: External interface files

Fig: - Functional Units

A system has five type of system functional unit: —

- (i) Internal logical files: — The control information & logically related data that is presented in the system.

(ii) EIF :- The control data or other logical data such that reference by the system but present in another system.

(iii) External input :- The data / control information that comes from outside our system.

(iv) External o/p :- Data that goes out of the system after generation.

(v) External inquiry :- Combination of input and output resulting in data retrieval.

Counting function point :-

STEP-1 Each function point is related according to complexity. They exist predefined.

Functional Units	Weighting FACTORS		
	Low	Average	High
E-I	3	4	6
E-O	4	5	7
E-Q	3	4	6
ILF	7	10	15
EIF	5	7	10

STEP-2 Calculate Unadjusted function Point (UFP) by multiplying each function point by its corresponding weight factors.

$$UFP = \sum_{i=1}^5 \sum_{j=1}^3 Z_{ij} * W_{ij}$$

Z_{ij} = Count of no. of F.P. of category i classified as complexity j .

W_{ij} = Weight factor

eg-

5 EI \rightarrow Low $\rightarrow 3$

3 EO \rightarrow High $\rightarrow 7$

$$UFP = 5 \times 3 + 3 \times 7 = 15 + 21 = 36$$

\rightarrow How to assign weight of function points is depending on the organisation based on past projects?

Step-3 Calculate final function point (FFP)

$$\boxed{\text{Final FP} = UFP * CAF}$$

Complexity Adjustment
Factor

CAF \Rightarrow CAF is calculated using 14 experts of processing complexity. 14 questions are answered on the scale of 0 to 5.

Scale:-

0 \rightarrow No Influence

1 \rightarrow Incidental

2 \rightarrow Moderate

3 \rightarrow Average

4 \rightarrow Significant

5 \rightarrow Essential

$$\boxed{CAF = [0.65 + 0.01 \times \sum F_i]}$$

F_i varies from 0 to 14.

Q. Given the following values compute function point when all complexity adjustment function and weighting factors are average.

user I/p = 50, user o/p = 40, user enquiries = 35

User files = 6

External interfaces = 4

$$\sum f_i = 14 \times 3$$

Ans -

$$UFP = \sum_{i=1}^5 \sum_{j=1}^3 z_{ij} * w_{ij}$$

$$= 50 \times 4 + 40 \times 5 + 35 \times 4 + 6 \times 10 + 4 \times 7$$

$$= 200 + 200 + 140 + 60 + 28$$

$$UFP = 628$$

$$CAF = [0.65 + 0.01 \times 14 \times 3]$$

$$CAF = 0.65 + 0.42 = 1.07$$

$$\text{Final FP} = 628 * 1.07 = \cancel{672.16} \quad 671.96$$

Advantages of using function approach:-

- (i) Size of SW is measured independent of language & technology and tools.
- (ii) Function points are directly estimated from requirements before design & coding.
- (iii) Useful even for those user without technical expertise
- (iv) Function point is based on external function structure of the system or SW to be developed.
- (v) Any change in requirements can be easily reflected in function point ~~count~~ count.

Q.2.1 Compute the function point value for a project with the following information.

No. of External Inputs = 32, No. of EO = 60,
No. of EQ = 24, No. of internal logical files = 8

No. of external interface files = 2 assume that all complexity adjustment value and weighting factors are average.

$$\sum i f_i = 14 \times 3$$

$$UFP = \sum_{i=1}^5 \sum_{j=1}^3 Z_{ij} * W_{ij}$$

$$= 32 \times 4 + 60 \times 5 + 24 \times 4 + 8 \times 10 + 2 \times 7$$

$$= 128 + 300 + 96 + 80 + 14$$

$$= 380 + 128 + 110 = 490 + 128$$

$$UFP = 618$$

$$CAF = [0.65 + 0.01 \times 14 \times 3] = 1.07$$

$$\text{Final FP} = 618 \times 1.07 = 661.26$$

COCOMO Models \Rightarrow

(Constructive cost model): - It is the model to estimate the cost of slw development and maintenance developed by Barry Boehm. It is the hierarchy of slw cost estimation model. It is the set of 3 models.

(i) Basic Model (ii) Intermediate model (iii) Detailed model

(i) Basic Model: - ~~It~~ It estimates the cost of slw development & maintenance in very rough and quick manner. It is not very precise and detailed.

Mostly used for small and medium size slw.

Basic model is consist of 3 model of development

3 model of development

Organic

Semi Detached

Embedded

	Organic	Semi detached	Embedded
1. Size	2-50 KLOC	50-300 KLOC	300 & above KLOC
2. Team Size	Small	medium	large
3. Developer Experience	Experienced	Average experienced	Very less previous experienced
4. Development Environment	Familiar env.	less familiar env.	Significant environment changes
5. Innovation Req.	little	Medium	Major
6. Deadline	No tight type	Medium	Payroll system tight
7. Example	Payroll system	Utility system	Air traffic monitoring system

⇒ Basic Model Equation: —

(i) Effort = $a(KLOC)^b$ person-month

(ii) Development Time = $c(Effort)^d$ months

(iii) Average Staff size = $\frac{Effort}{Dev. time}$ persons

(iv) Productivity = $\frac{KLOC}{Effort}$ KLOC/person-month

Mode	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Q. Suppose that a project was estimated to be 400 KLOC. Calculate effort and time for each of 3 models of development.

Ans - For organic Model: —

$$\text{EFFORT} = 2.4 (400)^{1.05} = 2.4 \times 539.71 = 1295.31 \text{ p-m}$$

$$\text{Development time} = 2.5 (1295.31)^{0.38} = 2.5 \times 15.23 = 38.075 \text{ month}$$

For Semi-detached :-

$$(i) \text{ Effort} = 3.0 (400)^{1.12} = 3.0 \times 820.93 = 2462.79 \text{ p-m}$$

$$(ii) \text{ Development time} = 2.5 (2462.79)^{0.35} = 2.5 \times 15.381 = 38.453 \text{ month}$$

For Embedded :-

$$(i) \text{ EFFORT} = 3.6 \times (400)^{1.20} = 3.6 (1325.781) = 4772.81 \text{ p-m}$$

$$(ii) \text{ Development time} = 2.5 (4772.81)^{0.32} = 2.5 \times 15.038 = 37.596 \text{ month}$$

Conclusion :- To accomplished the task of different difficulty level in same time we have to put different efforts and diff. no. of people for development.

~~Metri~~ Merits of Basic Model :-

→ Basic COCOMO Model is good for quick, early & rough estimate of sw project.

Limitation of Basic Model :-

→ Accuracy of this model is limited because it doesn't consider ~~a~~ certain factors for cost estimation of sw.

→ These factors are hardware constants, personal quality & experience, ~~modern~~ modern & latest techniques and tools.

2.) ^{COCOMO} Intermediate Model ⇒ This is an extension of basic COCOMO model. This cost estimation model makes use of a set of (cost drivers) 15 additional predictors to compute the cost of software development. It also takes development environment into account during performance estimation.

USE of COST Drivers :-

Cost driver adjusts nominal cost of project to actual project environment. The 15 cost drivers are divided into 4 categories:

(i) Product attributes :-

- (a) Required software reliability (RELY)
- (b) Data base size (DATA)
- (c) Product complexity (CPLX)

(ii) Computer attributes / Hardware attributes :-

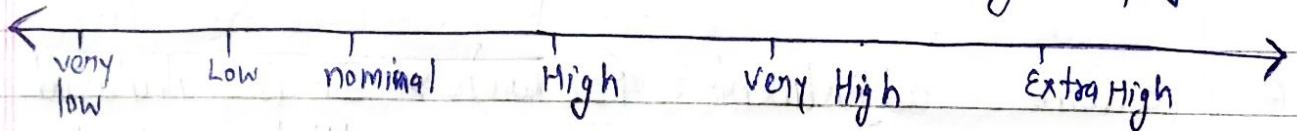
- (a) Execution time constant constraints (अवधि)
- (b) Main storage constant constraints
- (c) Virtual machine volatility
- (d) Computer turn-around time

(iii) Personnel attributes :-

- (a) Analyst capability
- (b) Application Experience
- (c) Programmer capability
- (d) Virtual machine experience
- (e) Programming language experience

- (iv) Project attributes :—
- (a) Modern programming practice
- (b) Use of s/w tools
- (c) Require development schedule.

Each cost driver is rated for the given project env.



This rating is done on the bases of, to what extent the cost driver applies to the project

Cost Drivers	Ratings					
	Very low	Low	Nominal	High	Very high	Extra High
Product attributes						
Required s/w reliability	0.75	0.88	1.00	1.15	1.40	
Size of application database		0.94	1.00	1.08	1.16	
Complexity of s/w	0.70	0.85	1.00	1.15	1.30	1.65
Hardware attributes						
Real-time performance constraints			1.00	1.11	1.30	1.66
Memory constraints			1.00	1.06	1.21	1.56
Virtual machine Volatility		0.87	1.00	1.15	1.30	
Computer turnaround time		0.87	1.00	1.07	1.15	
Personnel attributes						
Analyst capability	1.46	1.19	1.00	0.86	0.71	
s/w engineers capability	1.42	1.17	1.00	0.86	0.70	
Applications experience	1.29	1.13	1.00	0.91	0.82	
Virtual machine experience	1.21	1.10	1.00	0.90		
Programming language experience	1.14	1.07	1.00	0.95		

Project attributes					
Use of slw tools	1.24	1.10	1.00	0.91	0.82
Modern programming practice	1.24	1.10	1.00	0.91	0.83
Required development schedule	1.23	1.08	1.00	1.04	1.10

EAF (Effort adjustment Factor) \Rightarrow It is calculated by multiple all the values that have been obtained after ~~categorising~~ each of the cost drivers.
 ~~categorising~~

$$\text{EFFORT} = a(KLOC)^b \times \text{EAF}$$

$$\text{Development time} = c(\text{Effort})^d$$

Model	a	b	c	d
Organic	3.2	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.36
Embedded	2.8	1.20	2.5	0.32

Q. A new project ^{with} ~~with~~ estimated 400 KLOC embedded system has to be developed. Project manager has a choice of ^{highly} ~~highly~~ ^{heirng} from 2 pools of developers. (i) very highly capable (with application) ~~experience~~ ^{experience} in programming language choice in term of 2 pools. (ii) ~~With very little~~ ^{With very little} ~~experience~~ ^{experience} in programming language but ~~lots~~ ^{lots} of ~~exp~~ ^{exp} of programming language, which is better.

Ans- (i)
$$\text{EAF} = 0.701 \times 1.14 = 0.8094$$

$$\text{EFFORT} = 2.8 (400)^{1.20} \times 0.8094$$

$$= 2.8 \times 1325.78 \times 0.8094 = 3004.64 \text{ p-m}$$

$$\text{Development time} = \frac{3004.64}{2.5} = 1201.856$$

$$2.5 (3004.64)^{0.32} = 2.5 \times 12.96 = 32.421$$

(ii) EAF = low quality of developer's experience X lot of programming language

$$\text{EAF} = 1.17 \times 0.95$$

$$= 1.1115$$

$$\text{EFFORT} = 2.8 (400)^{1.20} \times 1.1115$$

$$= 2.8 \times 1325.78 \times 1.1115 = 4126.092 \text{ p-m}$$

$$\text{Development time} = \frac{4126.092}{2.5} = 1650.4368$$

$$= 35.884 \text{ months}$$

(i) EAF = very highly capable X very little experience in pro. lang

$$= 0.82 \times 1.14 = 0.9348$$

$$\text{EFFORT} = 2.8 (400)^{1.20} \times 0.9348$$

$$= 2.8 \times 1325.78 \times 0.9348 = 3470.1496 \text{ p-m}$$

Development time =

$$\frac{3470.1496}{2.5} = 1388.05984$$

$$2.5 (3470.1496)^{0.32} = 2.5 \times 13.58 = 33.95 \text{ months}$$

(I) is better

Merits of intermediate model \Rightarrow

- (i) This model can be applied to almost entire s/w product for easy & rough cost estimation during earlier stage.
- (ii) It can be also applied at the s/w product component model for obtaining more accurate cost estimation.

Limitation: -

- (i) The effort multipliers are not depended on Phases.
- (ii) A product with many components is difficult to estimate.

③ Detail COCOMO model: -

	Mode and Code size	Plan & Requirement	System Design	Detail Design	Code Test	Integration & Testing
	Life Cycle	Phase	Value of Σp			
size = 2	Organic Small size	0.06	0.16	0.26	0.42	0.16
Size = 32	Org. Medium size	0.06	0.16	0.24	0.38	0.22
size = 32	S-detach medium	0.07	0.17	0.25	0.33	0.25
size = 128	S-detach large	0.07	0.17	0.24	0.31	0.28
size = 128	Embedded large	0.08	0.18	0.25	0.26	0.31
Size = 320	Embedded XL	0.08	0.18	0.24	0.24	0.34

Life cycle phase Value of Σp

Organic small $s=2$	0.10	0.19	0.24	0.39	0.18
Org. medium size = 32	0.12	0.19	0.21	0.34	0.26
semi-detached medium $s=32$	0.20	0.26	0.21	0.27	0.26
semi-detached large $s=128$	0.22	0.27	0.19	0.25	0.29
Embedded large $s=128$	0.36	0.36	0.18	0.18	0.28
Embedded XL $s=320$	0.40	0.38	0.16	0.18	0.30

- Detailed COCOMO model is phase sensitive.
- It calculates the effect of cost driver on each phase of SDLC. It uses phase sensitive efforts multiplier for each cost driver. To determine the amount of effort to complete each phase of SDLC.
- It establishes module sub-system, system hierarchy

Estimation for each phase

$$\text{Effort} = \sum p E$$

$$\text{Development time} = \sum p D$$

6 phases of detailed COCOMO

- ① planning & req.
- ② system design
- ③ detailed design
- ④ module code & test
- ⑤ integration & test
- ⑥ cost constructive model

- Q. Consider the project with the following components
- | | |
|------------------------|---|
| (i) Speed screen edit | (4) Cursor movement |
| (ii) CLI | (5) Screen movement the size |
| (iii) file - i/p & o/p | for these are estimated to be 4k, 2k, 1k, 2k, 3k LOC using COCOMO determine overall cost & schedule estimation. |

(I) Overall cost & schedule estimates (assume values for cost drivers with atleast 3-being different from 1.0).

(II) Cost & schedule estimates for different phases

slw reliability: High = 1.15

Language Experience: Low = 1.07

Product complexity: High = 0.86

Analyst capability: High = 1.15

$$EAF = 1.15 \times 1.15 \times 1.07 \times 0.86 = 1.216$$

$$\text{Effort} = 3.2 (12)^{1.05} \times 1.216 = 52.9 \text{ P-M}$$

$$\text{dev. Time} = 2.5 (52.7)^{0.36} = 11.29 \text{ months}$$

(II) Cost & schedule estimates for different phases:
 $\text{Effort} = \sum p \times E$
 $\text{dev. time} = \sum p \times D$

effort:—

Plan & Req: $0.06 \times 52.9 =$

Design: $0.16 \times 52.9 =$

Detail design: $0.26 \times 52.9 =$

Code & test: $0.42 \times 52.9 =$

Integⁿ & test: $0.16 \times 52.9 =$

Dew time:—

Plan & Req: $0.10 \times 11.29 =$

Design: $0.19 \times 11.29 =$

Detail Design: $0.24 \times 11.29 =$

Integration & Test: $0.18 \times 11.29 =$

Code & test: $0.39 \times 11.29 =$

Software Risk Management:-

During development of any product or when we are doing any project we come across some risk.

This is a uncertain event they may have +ve or -ve effect on project.

→ What is Risk management?

Risk analysis & management are a series of steps that help a slw team interest and manage uncertainty.

→ Who does this management?

Everyone involve in slw process - managers, slw engineers and stocke holders participate in this analysis & management.

Risk management strategy

Strategic

Reactive

Productive

(1) Reactive Strategy:- In this strategy more commonly the slw team does nothing about risk until something goes wrong. Then the team flies into action in attempt to correct the problem rapidly. This is called as fire fighting Mode.

Strategic

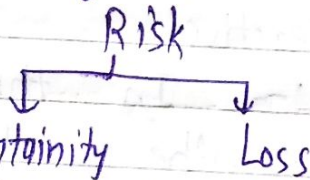
(2.) Productive strategy:- A considerably more intelligent strategy for risk management is to be productive.

Productive strategy begins long before technical work is initiated. Potential risk are identify there probability and import are recognised and they are ranked by importance. When the slw

Team establishes a plan for managing risk. The primary objective is to avoid risk but because not all risk can be avoided so that team works to develop a plan it will enable it to respond in a controlled and effective manner. There is a general agreement the risk always involved to characteristics:—

(i) Incertantity \Rightarrow The risk may or may not happen. There are no 100% possible risk.

(ii) Loss \Rightarrow If the risk becomes reality unwanted consequences and does ~~not~~ may occur, when risk is analysed it is important to qualify the level of uncertainty and the degree of loss associated with each risk.



to accomplished this different categories of risk are considered.

Degree of Risk:— (i) Project risk threatens the project plan that is if the project risk become real it is lightly that project schedule will slip and cost will increase. Project risk identifies potential budgetry, schedule, personal (staffing & Organisation) Resource, Stake holders & requirements problems and their impact on slw project.

(ii) Technical Risk: —

Technical risk threatens the quality and the timeliness of the sw to be produced. If a technical risk becomes a reality implementation may become difficult or impossible. Technical risk identifies potential design, implementation, interface, verification and maintenance problems.

In addition specification ambiguity, technical uncertainty & leading edge technology are also risk factors. Technical risk occurs, because the problem is harder to solve than we thought it could be.

(iii) Business Risk: — Threatens the validity of the sw to be built.

Top 5 business risks are

- (i) Building an excellence product or system that no one really wants (market risk)
- (ii) Building a product that no longer fits into the overall business strategy for the company.
(Strategy Risk)
- (iii) Building the product that the sales force does not understand how to sell. (Sales Risk)
- (iv) ~~thing~~ Losing the support of senior management due to a change in focus or change in people
(Management Risk)
- (v) Lose in budget rate or personal commitment
(Budgetary Risk)

② Risk identification \Rightarrow R.I. is a systematic attempt to specify ~~trades~~ threats to a project plan. (Estimates, schedule, resource loading). One method for identifying risk is to create a Risk item checklinks. The checklinks is used to for R.I. & focus on some subset of no one aim predictable risk in the following generic sub-category :

(i) Product size :- Risk associated with the all size of sw to be built or modify.

(ii) Business impact :- Risk associated with constant imposed. ~~imposed~~ by management or the market place.

(iii) Customer characteristic :- Risk associated with the sophistication of customer and the developer ability the communication with the customer in the timely manner.

(iv) Process Definition :- Risk associated with the sw process has been defined and ~~the~~ is followed by the development organisation.

(v) Development Environment :- This associated with the availability and quality of the tools to be used to built the product.

(vi) Technology to be built \Rightarrow Risk associated with the complexity of the system to be built ~~at the~~ enter newness of the technology that is a package by the system.

(vii) Staff size & experience:— This associated with the overall technical and project experience of the slw engineers who will do the work.

After this step, finally a set of risk components & risk drivers are listed along with their probability of occurrence.

~~1~~ (1) ^{आकॅन} Assessing overall project risk \Rightarrow Some questions have been derived from risk data obtained by ~~serv~~ surveying experience slw project managers in different parts of the work.

These questions are ordered by their relative importance to the success of the project:—

- (A) Have top slw & customer formally committed to support the project?
- (B) Are on user enthusiastically committed to the project & the system to be built?
- (C) Are requirements fully understood by the slw ~~And~~ engineering team & its customer?
- (D) Have customers being involved fully in the definition requirement?
- (E) Do end users have realistic expectation?
- (F) Is ~~the~~ project scope stable?

Have

- (G) ~~Do~~ Do slw developer team have right mix to skills?
 - (H) ~~Are~~ projects requirement stable?
 - (I) ~~Does~~ the project team have experience with the technology to be implemented?
 - (J) Is the no. of people on the project team adequate to the job?
 - (K) Do all customer agree on the ~~information~~ importance of the project and on the requirements for the system / product to be built?
- If any one of these question is answered negatively when RMMM plan (Risk mitigation monitor management plan)

(2) Risk components & drivers ⇒

The US Air-Force has written a pamphlet that contains excellent guidelines for slw risk identification and removal of risk. These approach requires that the project manager identify the risk drivers that effects slw risk components. Risk components are defined as:-

- (i) Performance Risk:- The degree of uncertainty that the project will ~~need~~ its requirement and will fit into intended use.
- (ii) Cost Risk:- The degree of uncertainty that the project budget will be maintained.

(iii) Support Risk:— The degree of uncertainty that the resultant ~~slw~~ will be easy to correct and enhance.

(iv) Schedule Risk:— The degree of uncertainty that the project schedule will be maintained and the project will be delivered on time.

The impact on each risk driver ~~on~~ the risk component is divided into one of the four ~~and~~ impact categories:—

(i) Negligible

(iii) Critical

(ii) Marginal

(iv) Catastrophic

Risk PROJECTION:—

→ It is also called as risk estimation. It attempt to rate each risk into 2 ways

(i) The likely hood / probability that risk is real

(ii) The consequence of problem associated with risk.

→ The project planners along with managers & technical staff perform risk projection ~~stage~~ Step:—

(A) Establish the scale that reflects a perceived likelihood of risk.

(B) Describe the consequence of risk

(C) Estimate the impact on the risk ~~though~~ on the project and the proj product.

→ The intent of these step is to considered risk in a manner that leads to a prioritization

Risk Table :-

Name of Risk	Category	Probability	Impact	RMMM

Developers a risk table: provides project manager with a simple technique for risk estimation.

- (i) Team list all risk in first column.
- (ii) Each case risk into second column.
- (iii) Probability of occurrence of list in 3rd column.
- (iv) Impact of risk is mentioned in 4th column.

Software project scheduling:-

(1) Basic concept \Rightarrow

- Although there are many reasons why slw is delivered late? Most can be traced ~~precised~~ to the following root causes :-
- (i) Unrealistic deadline established by someone outside the slw engineering group and forced on managers, an practitioners within the group.
 - (ii) Changing customer requirements ~~data~~ that are not reflected in schedule changes
 - (iii) Underestimate of the amount of effort and the no. of resources that will be required to do the job.
 - (iv) Predictable and unpredictable risk that are not reflected in schedule changes

- (v) Technical difficulties that could not have ^{been} forcing in ~~advant~~ advance.
- (vi) Miscommunication among project staff that result in delays.

(A) Basic Principle of scheduling : —

- (i) **Compart-mentization** \Rightarrow The project must be divided into a no. of manageable activities, action & task. To accomplished compart mentization both the product & process are decomposed.
- (ii) **Interdependency** \Rightarrow The interdependency of each divided activity, action & task must be determined. Some task must occurs in sequence while others can occur in parallel. Some activity / action can not comments until the work product produce by other is available. Other action can occur ~~in~~ independently.
- (iii) **Time allocation** \Rightarrow Each task to be scheduled must be allocated some no. of work unit (person days of effort). In addition, each task must be assigned a start date and the completion date. That are a function of the interdependency & whether work will be conducted on the full time or part time.
- (iv) **Effort validation** \Rightarrow Every project has a define no. of people on the slw team.

Is time allocation occurs the project manager must ensure that no more than allocated no. of people having been scheduled at any given time.

(5.) Define Responsibility \Rightarrow Every task that is scheduled should be assigned to a specific team member.

(6.) Define outcome \Rightarrow Every task ^{that is} get a scheduled outcome should have defined outcome. For slw project the outcome is normally a work product.

(7.) Define Milestone \Rightarrow Every task or group of task should be associated with the project milestone. A milestone is accomplished when one & more project have been reviewed for quality and has been ~~approved~~ approved.

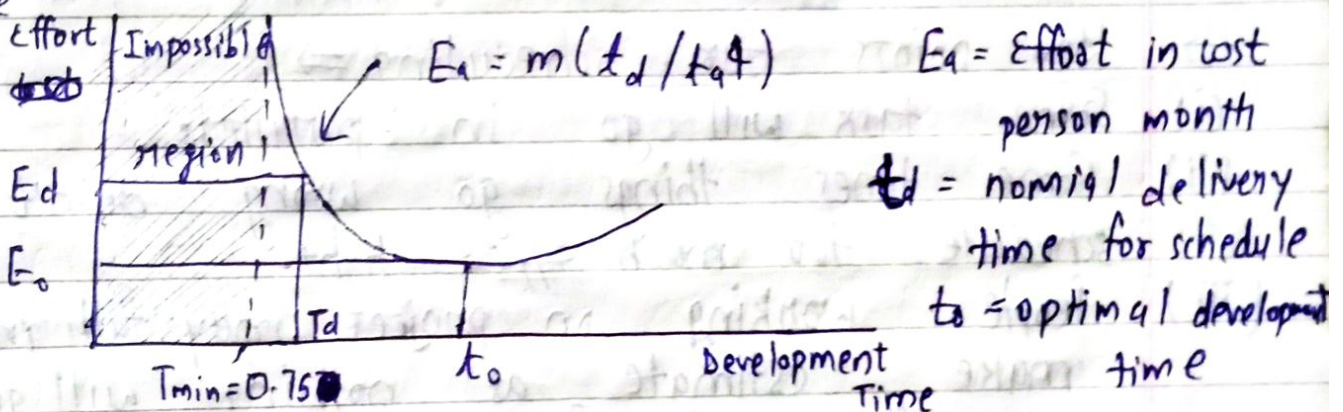
(B) The Relationship b/w people & Effort \Rightarrow

In the small slw project a single person can analyse requirements, perform design, generate code & conduct test. As the size of project increases more people must involve. There is a common myth that is still believed by many managers who are responsible for slw development efforts. If we fall behind schedule we can always

need more programmers and catch up later in project. Unfortunately adding people later in the project has a disruptive effect causing schedule slip even further. The people who are added must learn the system & the people who teach them are the same people who are doing the work. During teaching no work is done and Project falls further behind.

The Putnam - Norden - Rayleigh (PNR)

Curve



(c) Effort distribution \Rightarrow A recommended distribution of effort across the software project process these often referred as the 40-20-40 Rule. 40% of all efforts there is allocated to front end analysis and design. A similar % is applied to back end and testing. And the remaining 20% effort is applied for coding. You can correctly infer that coding is deemphasized.

Project scheduling:- This is an activities in which decide How we will decide project into the Smaller task? How this task will be executed and how much time take to task complete?

- Step-1 Set milestone
Step-2 Estimate Resources

Project Schedule Representation

- Key points for scheduling ⇒
- (i) Some task will go in parallel.
 - (ii) Some times things go wrong do-n't make schedule too much type tight.
 - (iii) People working on project may fall in so make estimate as nothing will go wrong and then add few more days through the final estimate as if go something wrong so we have at least some extra good.

Project Schedule Representation:-

Task	Duration	Dependencies
T1	14	
T2	7	→ T(1) } F1
T3	20	→ (M1) } F1
T4	6	→ T3 } F2
T5	27	→ (M2) } F2
T6	7	
T7	10	→ M3 } F3

Unit -3

Requirement Analysis & Specification

Need of RAS ⇒

- Many sw projects fail -
- (i) If developers start implementing the sw without understanding the reqⁿ of the customer & product.
- So requirement Analysis & Specification is needed to understand the system reqⁿ in documented form.

Goal of RAS ⇒

To fully understand reqⁿ, remove in consistency, anomalies from reqⁿ and prepare document

Requirement analysis Task ⇒

- (i) Elicitation of Requirement :- It means gathering of reqⁿ by communicating customer, users and others and decide what are the objective for the system and product are. What is to be accomplished?
- How the system or product fit into the need of the business and how the product is to be used on the day to day base.
- Kang identify the no. of problem that help to understand by reqⁿ elicitation is difficult.

(A) Problem of scope:- The boundary of system is ill defined or the customer specify unnecessary technical details that ~~get~~ may confuse rather than clarify over all system objective.

(B) Problem of understanding:- The customer is not completely sure what is needed, don't have full understanding of the problem domain, have trouble communicating means needs to the system engineer, omit information that is believed to be obvious and specify requirements that are unstable.

(C) Problem of volatility:- The reqⁿ change over time

To help overcome these problems reqⁿ engineer must approach the reqⁿ gathering activity in a organised manner.

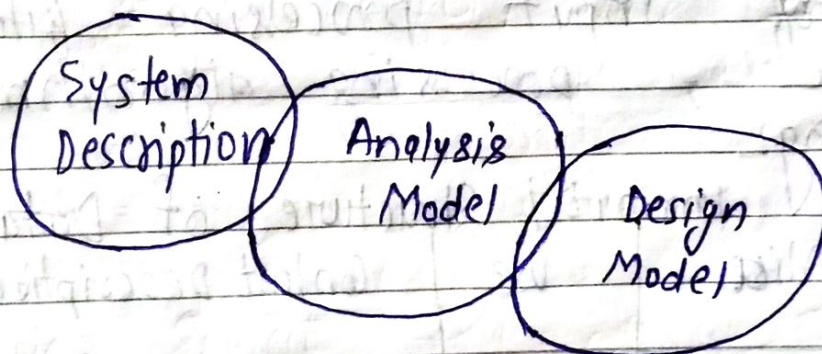
(2.) Analysing the requirements \Rightarrow

This step help to determine the quality of reqⁿ. It involves identifying whether the reqⁿ are unclear, incomplete & contradictory these issues are resolved before moving to the next step.

(3.) Requirements modelling \Rightarrow In reqⁿ modelling the reqⁿ are usually documented in different format such as use case, user stories, natural language documents or process specification.

(4.) Review \Rightarrow Review is conducted to make improvement in the previous step.

Requirement Analysis Principle \Rightarrow



Throughout analysis model the software engineer primary focus is on what. What the objects does the system manipulate?, what function must the system perform?, what behavior does the system exhibit?, what interfaces are defined? and what constraint applied?

Overall objective:-

The analysis model must achieve

3 primary objective-

- (i) To describe what the customer required.
- (ii) To establish the basis for the creation of s/w design.
- (iii) To define a set of reqⁿ that can be validated once the s/w is made. built.

Data dictionary \Rightarrow

Data dictionary is the collection of descriptions of the data objects or item in a data model. For the benefit to the programmers & others who need to refer them.

OR

(data about data)

Data dictionary is known as Metadata \approx repository. A data dictionary becomes a source document for specification & design of input processing file & data structure, processing algorithm & output processing.

Format / Structure of Data structure

Name	Aliases	Use	Content Description	Other Information

Name \Rightarrow A formal or primary name of the data or control item, data storage & external entity.

Aliases : — It is the other name or ~~acronym~~ ~~alias~~ of data item

Use : — A listing of process which use the data on control item and when it is used? and how

Content Description : — Standard format of representing the content of the data on control item.

Additional information : — other information such as limitations, restriction, default or initial values etc.

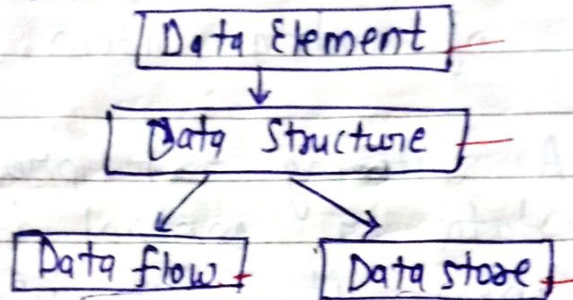
Symbols used in data - Dictionary : —

Symbols	Definition
(i) $=$	is composed of [Equivalent to]
(ii) $+$	and (exclusive)
(iii) $[]$	either / or (Exclusive)
(iv) $\{ \}^n$	'n' repetition of
(v) $()$	optional data entry
(vi) $* \cdots *$	comments

★ Example : Project data Element sheet of a telephone number

Name	Alias	Content	Additional Information
① Telephone no.	—	Telephone no. = [Local no. / long distance no.]	—

Classes of Data Item :-



Data Element \Rightarrow The smallest unit of data that cannot be further decomposed.

Data structure \Rightarrow It is the group of data element refer as a single unit.

(A) Group item - combination of data item for some logical purpose.

(B) Record - collection of data item or group item which are gathered to define an entity for a particular ^{processing} purpose.

Ex: User telephone No: data item

User name & address: Group item

(C) Files :- collection of record

(d) data base \Rightarrow database is collection of file.

Data Flow :- Group of data structure in & data store motion is known as data flow & group of data structure in rest is known as data store.

Advantages of data-Dictionary \Rightarrow

(i) - It provides documentation.

(ii) - It improves the communicate b/w system analyst & user by establishing consisting definition of various item, term & procedures.

(iii) It is used to remove redundancy in data definition.

Finite State Machine

FSM is a mathematical model of computation.

→ It is an abstract machine that can be

in exactly one of a finite no.

of state at a given no. of time.

→ The FSM can change from one state

to another in response to some

external inputs.

→ The change from one state to another is

called transition.

State :— A state is the description of the status

of system i.e. ~~waiting~~ to execute transition

Transition :— A transition is a ~~state~~ of action to

be executed when a condition is

fulfilled or when an event is received

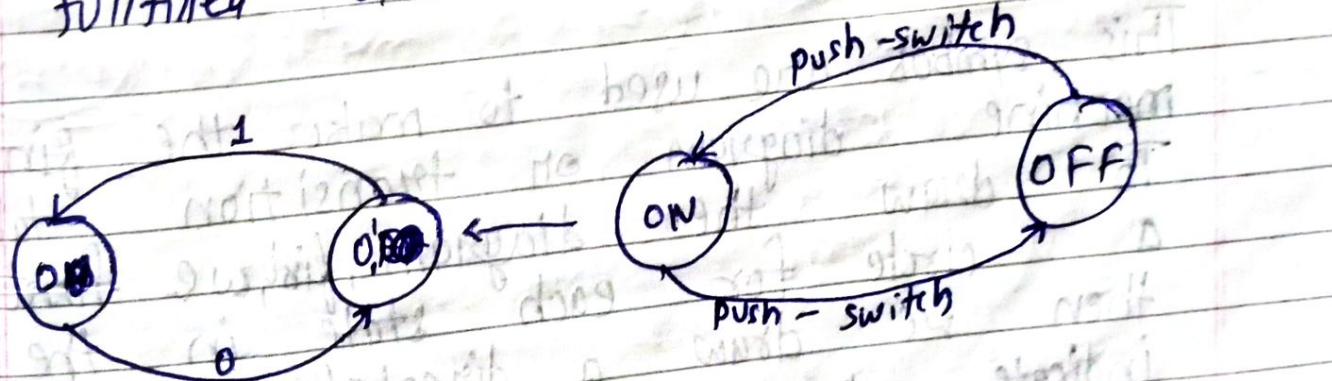


fig: FSM of a tubelight

2. State of FSM:-

- (i) An entry action \Rightarrow performed when entering a state.
- (ii) An ~~exit~~ exit action \Rightarrow performed when exiting a state.

\rightarrow Such machine are basically appropriate tools to express the requirement & design specification of sw product.

symbols used

$\Sigma =$ Finite set of inputs

$\Gamma =$ Finite set of states

$s =$ Starting state

$\delta =$ Transition function

$\delta:-$ Input \times State \rightarrow State
 $\delta: \Sigma \times \Gamma \rightarrow \Gamma$

This symbols are used to make the finite state machine diagram or transition table.
To draw the diagram, first, we first draw a circle for each state in the diagram, then we draw a directed arrow to indicate the transition from one state to another.

Q. The following is the description of an FSM

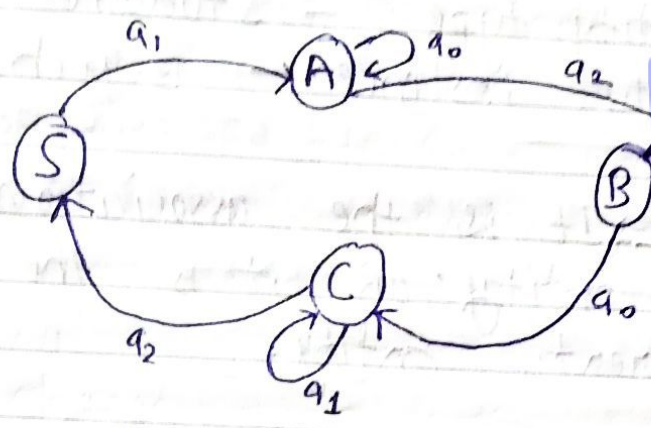
$$\Sigma = \{a_0, a_1, a_2\}$$

$$\Gamma = \{S, A, B, C\}$$

S = starting state

$$\begin{aligned} \delta(S, a_1) &= A, & \delta(A, a_0) &= A, & \delta(A, a_2) &= B \\ \delta(B, a_0) &= C, & \delta(C, a_1) &= C, & \delta(C, a_2) &= S \end{aligned}$$

Ans -



Er Sahil
Ka
Gyan

Structured Analysis \Rightarrow Structured analysis (Software Modelling) is the model building activities. It is the job of the sw engineer to build a structure model by using the requirements as specify by the customer.

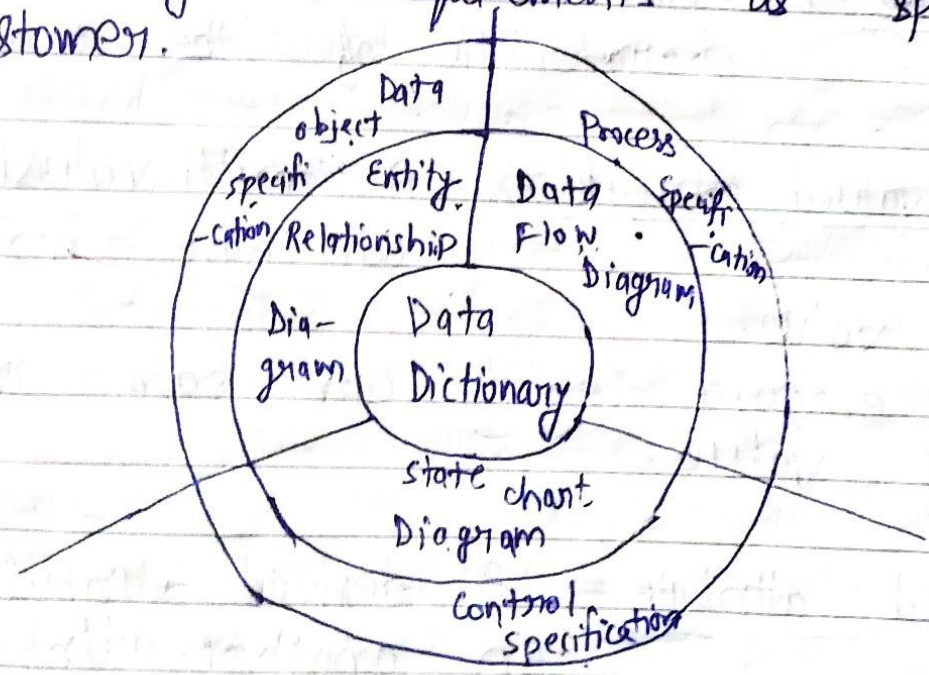


fig: Elements of structural analysis model

Entity Relationship Diagram \Rightarrow

Entity :- It is an object or theme that is different from another object.

Entity type :- Entity type belongs to particular time kind of entity.

Entity instance :- Entity instance is a group of entity type.

eg - Ram is a student in a B.Tech class.

Entity :- Ram

Entity type :- Student

Entity instance :- B.Tech class

Relationship \Rightarrow It is the association among different entity. OR it is the connectivity among different entity.

Weak entity \Rightarrow A weak entity is an entity that have no primary relationship with another entity.

Attribute \Rightarrow Attribute is unique or different characteristic of the entity.

Multi valued attribute \Rightarrow A multi valued attribute can have more than one value.

eg - An employee entity can have multiple skilled value.

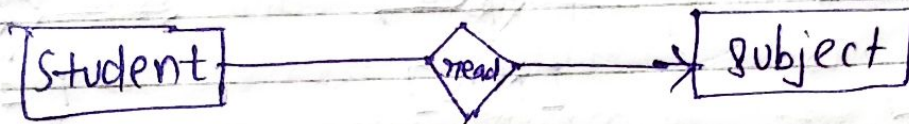
Derived attribute \Rightarrow A derived attribute is based on another attribute. for ex - An employee monthly salary is based on the employees annual salary.

Cardinality \Rightarrow Cardinality specify how many ^{instances} of the entity relate to one instance of another entity

- ① $[1:1]$: — In this mapping, one entity relates to one another entity.



- ② $[1:M]$: — In this mapping, one entity related to many other entity.





- ③ $[M:1]$: — In this mapping, many entity related to one entity.

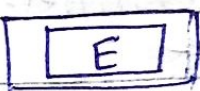
- (iv) $[M:M]$: — In this mapping, many entities related to many another entities.




Symbol used in ER diagram \Rightarrow


- 1.]  $\xrightarrow{\text{rectangle}}$: — represent entity


- 2.]  $\xrightarrow{\text{circle}}$: — Attribute of an entity

- 3.]  $\xrightarrow{\text{Double rectangle}}$: — Weak entity


- 4.]  $\xrightarrow{\text{Double circle}}$: — Multivalued attribute




- 5.]  $\xrightarrow{\text{Dotted circle}}$: — Derived attribute

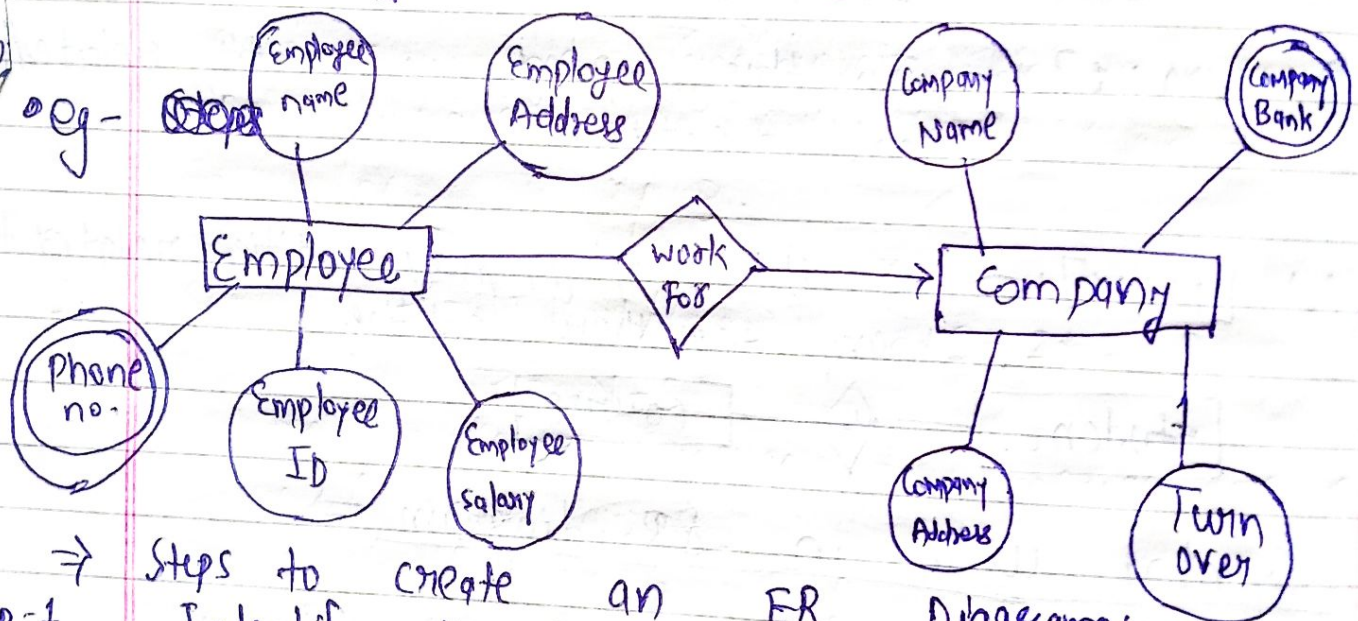
6.1  Diamond → Represent Relationship among entities

7.1  Double Diamond → Relationship set for weak entities

8.1  → — primary key attributes

9.1  → — attribute of weak entity set

10.1  [M:M]  [M:1]  [1:1]



- ⇒ Steps to create an ER Diagrams! —
- Step-1 Identify the entities
 - Step-2 Identify the entity attributes
 - Step-3 Identify the primary keys
 - Step-4 Identify the relation b/w different entities.
 - Step-5 Map the cardinality constraint
 - Step-6 Draw the ERD and check the ERD.


(2) Data Flow Diagram \Rightarrow A DFD is one means of representing the functional model of a software product. DFD looks like a flowchart but both are different in a very important base.


(A) DFD don't supply detailed description of internal functions sub-routine. But show the flow of data on information and the interrelationship among several transformation process.

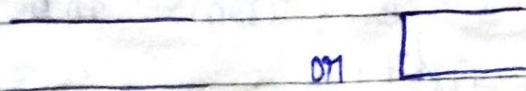
(B) Flowchart describes about the flow of control but DFD describes about the flow of data.

Data Flow diagram is also known as couple bubble chart. As it consists of a series of bubbles joint by lines. Where the bubble represent the data, transformation and the line represents data flow in the system. The basic purpose of DFD is to classify the system requirements & identify major transformation that will become programmable in system design.


DFD elements \Rightarrow


(i)  \rightarrow It represents external entity.

(ii)  \rightarrow It represents process that transform data flow.

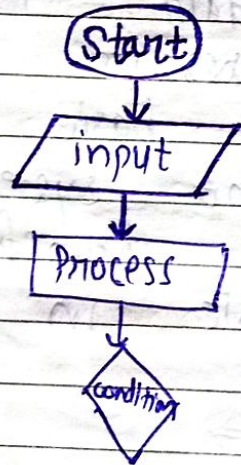
(iii)  \rightarrow It represents data store
It may represent data base

file, general documents, slips.

(iv)  \Rightarrow It represents data flow.

(v)  \Rightarrow It represents the decision condition and looks to implement decision.

(3) **FLOW CHART** \Rightarrow It is the diagrammatic or pictorial view of diagram program. It represents
 (i) logical model of program
 (ii) flow of control within the program
 (iii) Flow of control instruction within the program
 (iv) The way in which instructions are executing.



Rules to draw DFD \Rightarrow

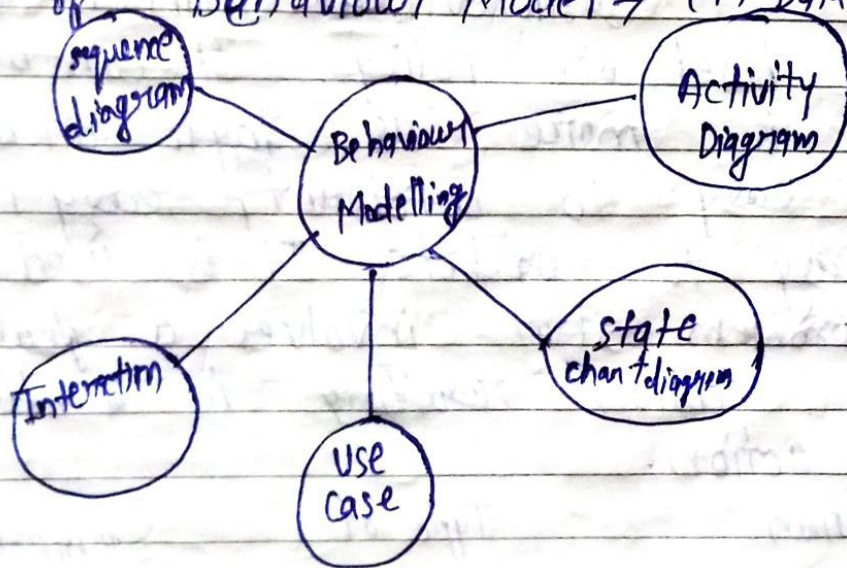
- (i) Arrow should not cross each other.
- (ii) Not 2 element of DFD can have same name.
- (iii) All elements must have meaningful names.
- (iv) The direction of flow is from top to bottom and from left to right.
- (v) Data dictionary flow from the source to the destination.

Page No. :
Date :
They may flow back to a source.
The name of a data store, sources and destination should be written in capital letter.
Process & data flow name should have a first letter of each word capitalised (title case)

Behavioural Modelling: - It is an operational principle for all requirement analysis method. It graphically represent how system is work stage switch to another stage. OR overall

Behaviour modelling represent ^{overall} behaviour of the system. The behaviour modelling shows interaction b/w objects to produce ^{some} a particular system behaviour. i.e. specify as a use-case.

Type of Behaviour Model \Rightarrow (i) Data model (ii) State machine model



Er Sahil
Ka
Gyan

(i) Data model: - Data model shows how data is processed as the it moves through the system.
eg - use case diagram

(ii) State machine model: - This model shows how the system will response to event.
eg - state chart diagram.

Both of these models are required for the description of system behaviour.

Advantages of Behaviour modelling:—

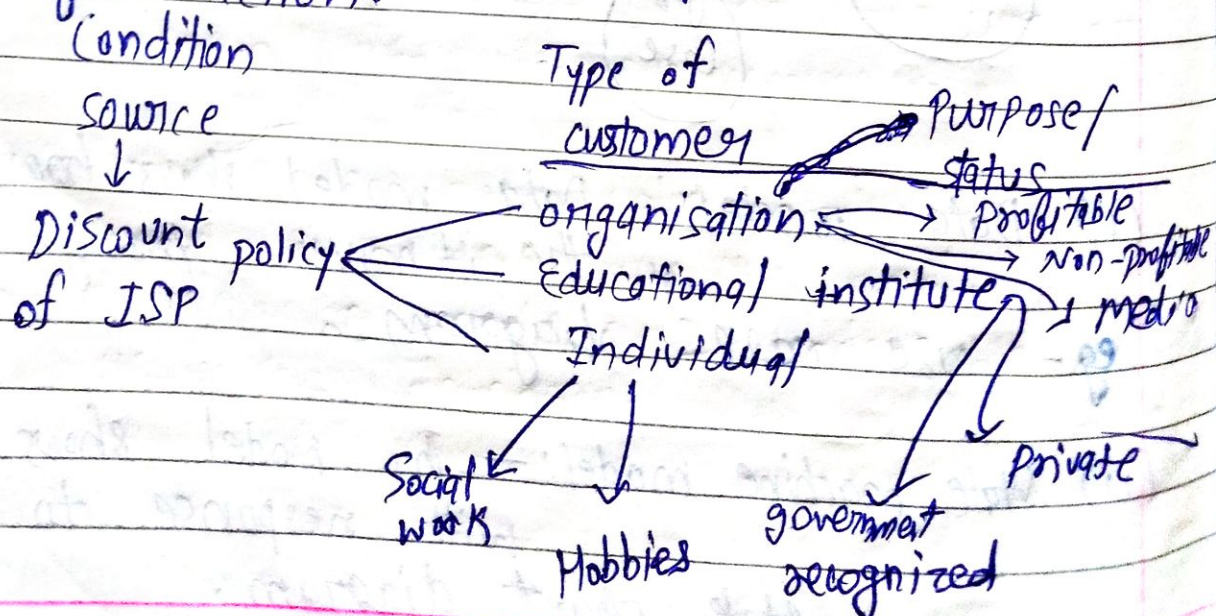
- (i) Reliability:— It refers to how easily & its logic can be read.
- (ii) Understandability:— It refers to how easily program can be understood.
- (iii) Comprehensibility:— It refers to how easily program can be comprehended.



Other structural Analysis Model:—

- (i) Decision Tree ⇒ Decision Tree is also like a structured manner. Its structure based on policy. It can be branch is more as logic or alternative. It is easy to construct, easy to read and easy to update. It is used to verify problem that involves a few complaint decision resulting in a limited no. of action.

eg—



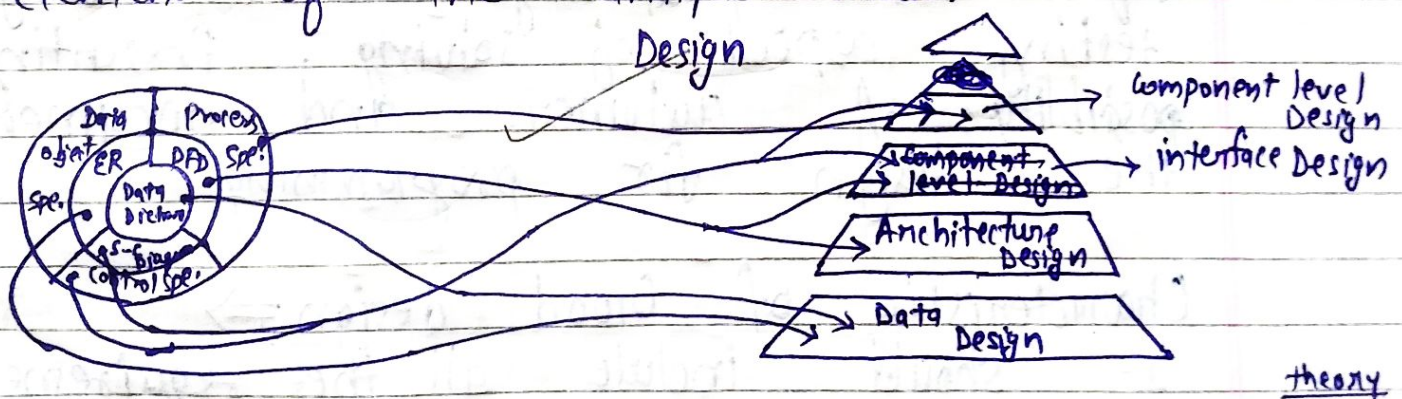
Unit - 4 Software Design

requirement $\xrightarrow[\text{sequential steps}]{\text{transform}}$ representation of system

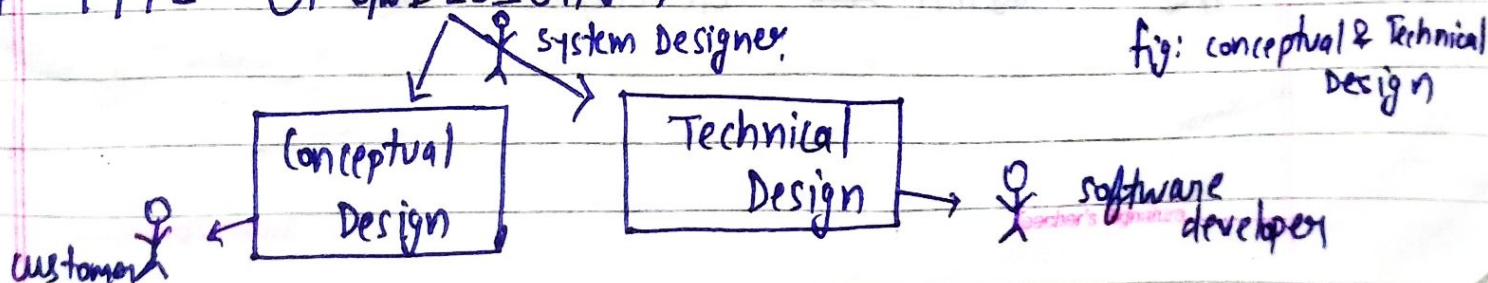
sw Design:- It transforms the software reqⁿ through a no. of state into a final product. A sw decomposition into module, description of both each module is intended to do. And the relationship among the module. Such type of description is called as sw architecture or sw structure.

- Development of sw design model \Rightarrow The design model is developed by the sw designer only. His main goal is to develop a model based on experience, design principle, its guidelines and fundamental concept.

Types of All design models are developed by using the element of the analysis model



TYPE OF sw DESIGN \Rightarrow



Designer must satisfy both customer & system builder (s/w engineer). The customer should understand what the system will do? at the same time the system builder understand how to do? To accomplished these design is divided in two part.

- (i) Conceptual or preliminary design.
- (ii) Technical or detail design.

(i) Conceptual design produces that tell the customer exactly what system will do? It belongs to what ~~both~~ part of solution. One the customer approved it is transferred into technical design which allow s/w engineer to understand the actual hardware & the s/w needed to solve customer problem. It belongs to how part of solution. This process is iterative. The designer move ~~that~~ ~~can~~ back and forth among the activity involving understanding the requirement ^{proposing} ~~proposing~~ the solving, testing aspect of solving. Presenting possibility of customer and document the design for programming.

Characteristic of Good design \Rightarrow
 It should include all the requirement.
 Design should be readable & understandable by the customer-developer.

Design principle:—

- (i) The design process should not suffer from tunnel vision.
- (ii) The design should be possible through the analysis model.
- (iii) The design should not reinvent the wheel.
- (iv) The design should exhibit uniformity & integration
- (v) The design should be structured to accommodate changes.
- (vi) Design is not coding & coding is not design.

Fundamental Design concept \Rightarrow

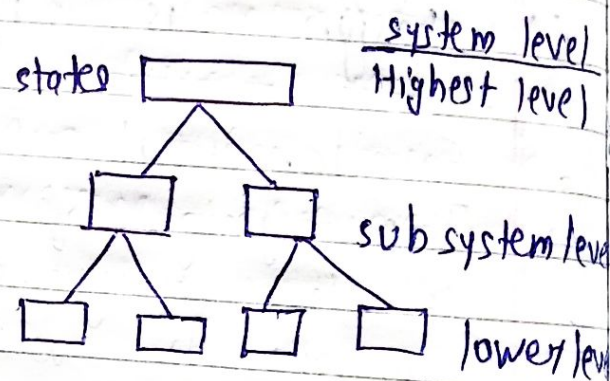
- \rightarrow The beginning of wisdom for the computer programmer is to recognise the difference b/w getting a program to work and getting it right.
- \rightarrow Fundamental software design concepts provide the necessary framework for
 - (i) Decomposition and modularity
 - (ii) Abstraction
 - (iii) Step wise refinement
 - (iv) software architecture
 - (v) Control hierarchy
 - (vi) Data structure
 - (vii) software procedure
 - (viii) Structural partitioning
 - (ix) Information hiding.

(i) Decomposition & Modularity:—

There are many methods to create a design, however every design method involve some kind of starting with the high level description of the systems key elements & creating lower level loop at how the system feature & function will filled together.

The modular design can be created into 5-ways -

- (i) Functional
- (ii) Event - oriented
- (iii) Object - oriented
- (iv) Outside - In
- (v) data - oriented



Need of Modularity:—

Problem : P_1 & P_2

complexity	function	$C(X)$
effort	function	$E(X)$
	$C(P_1) > C(P_2)$	
	$E(P_1) > E(P_2)$	
	$C(P_1 + P_2) > C(P_1) + C(P_2)$	
	$E(P_1 + P_2) > E(P_1) + E(P_2)$	

→ It results that combined effort will be much so if we required that the complexity & effort should not be much to solve a problem.

(iii) Stepwise Refinement \Rightarrow It is a top down approach. If we have a big problem then it breaks it down into smaller part so it is easier to manage the problem. So that the parts are small enough to be programmed as a single subroutine. It is all known as divide and conquer subroutine.

(A) Subroutine (B) function
(C) Procedure (D) Parameter

Set Steps of Refinement :-

- (i) Start with the main method
- (ii) Identify or write any required function
- (iii) Look over complex section.
- (iv) Identify helper function
- (v) Repeat

Advantage :-

- (i) Each module on focus sub task.
- (ii) Problem is easier to solve, easier to test & easier to maintain.
- (iii) Easier to update.
- (iv) Development can be shared b/w the programmer.
- (v) Different module can be programmed in different language.
- (vi) Module can be reused.

- DATE / /
- Disadvantages:-
- (i) It can be difficult to link module together.

④ Software Architecture \Rightarrow

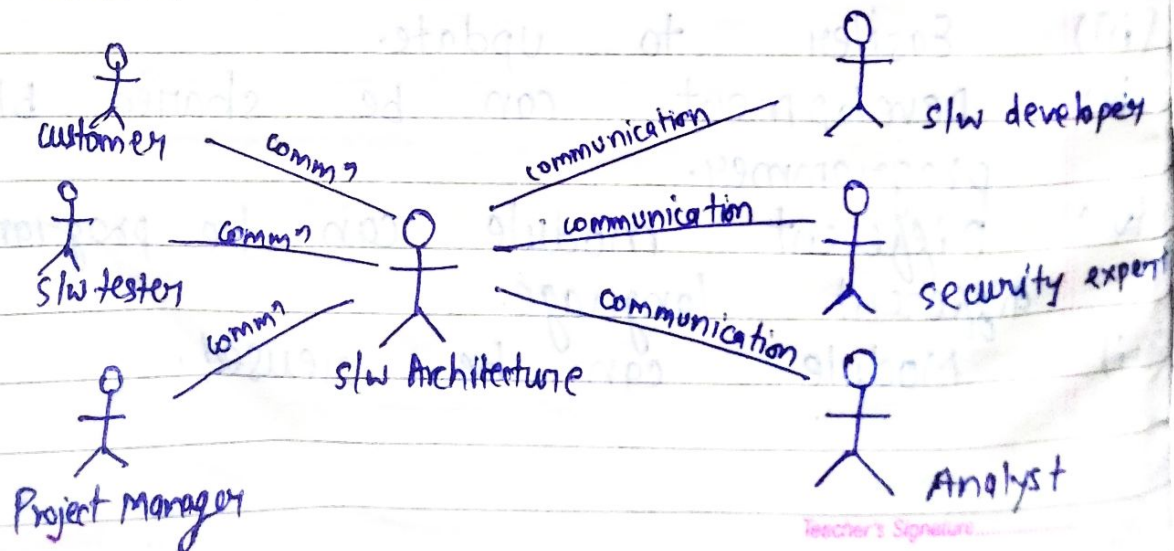
It is the structure of sw system like the blueprint in the building architecture.
sw architecture consists of :-

- (i) sw components
- (ii) Details regarding data structure & algorithm
- (iii) Relationship among the components.
- (iv) Data flow, control flow & dependency.

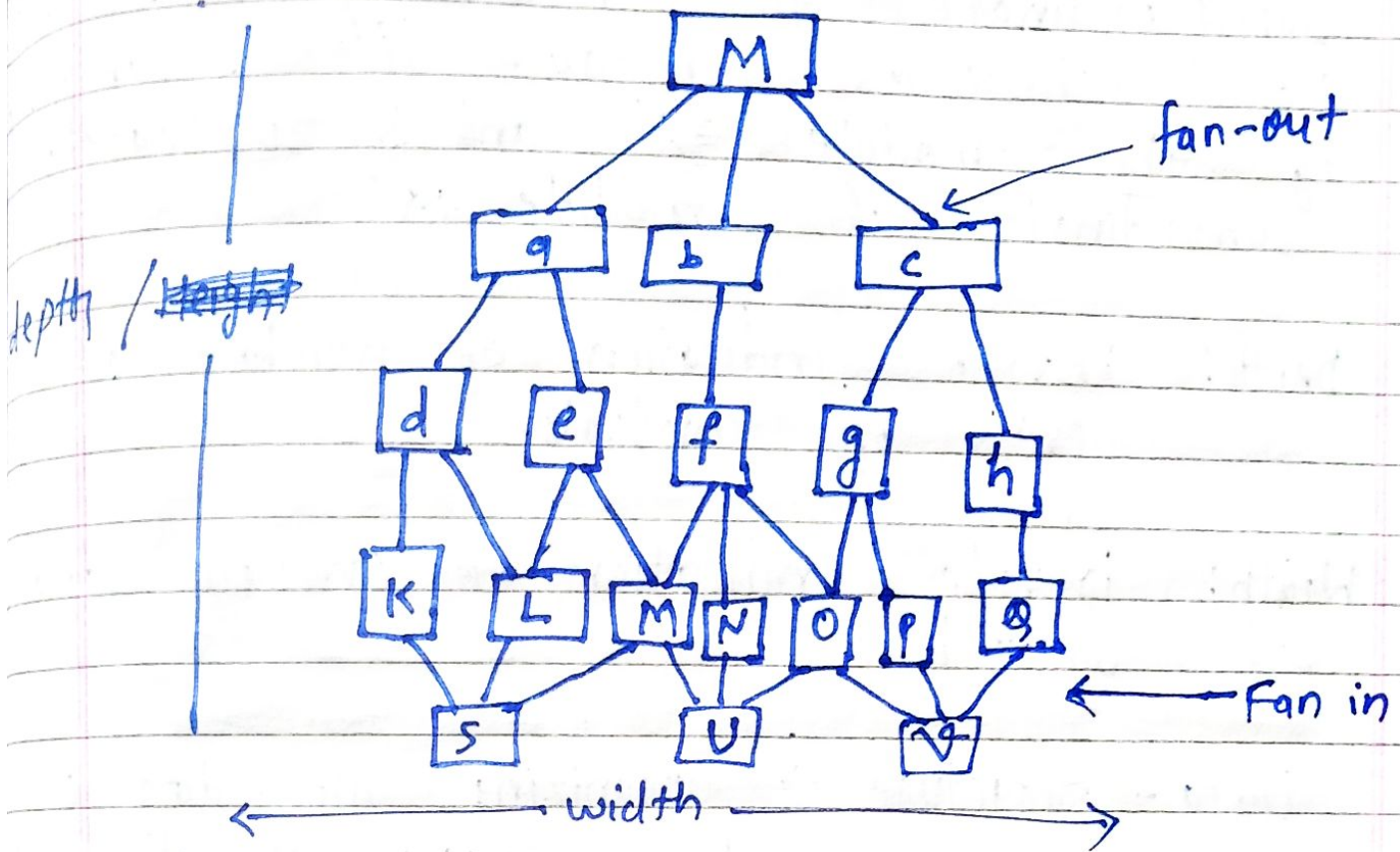
If we know all these details then we can develop the sw architecture.

When to develop sw architecture?

- (1) If sw is consisted of 500 LOC.
- (2) If sw is consisted of 5000 LOC then use class diagram.
- (3) For more than 5000 LOC go for sw architecture.



⑤ Control Hierarchy \Rightarrow



It is also called as program structure it represents the hierarchy of control. It describes hierarchy of control by omitting the procedural aspect that is sequence of process, occurrence or order of decision or repetition of operation.

The tree-like structure notation is used to represent the control hierarchy. Program structure is usually a simple hierarchy showing super ordinate (module that control another module) and sub-ordinate (i.e. that is controlled by another module) relationship of modules.

In fig: Controlling module 'M' is super ordinate to module 'a', 'b' and 'c' where as module 'f' is subordinate of the module 'd' and is ultimately subordinate of module 'M'.

Depth: Provide indication of number of levels of controls.

Width: Provide indication of overall span of control.

- ⑥ Data-structure \Rightarrow Organizing the data in the memory is called the data-structure. There are many ways to organize data in the memory. Data-structure represents the organization of data, methods of access, degree of associativity and processing alternatives for information.

The classical data structures that form the building blocks for more sophisticated structures are:

Scalar item, sequential vector, linked list, an n -dimensional space & a hierarchical tree.

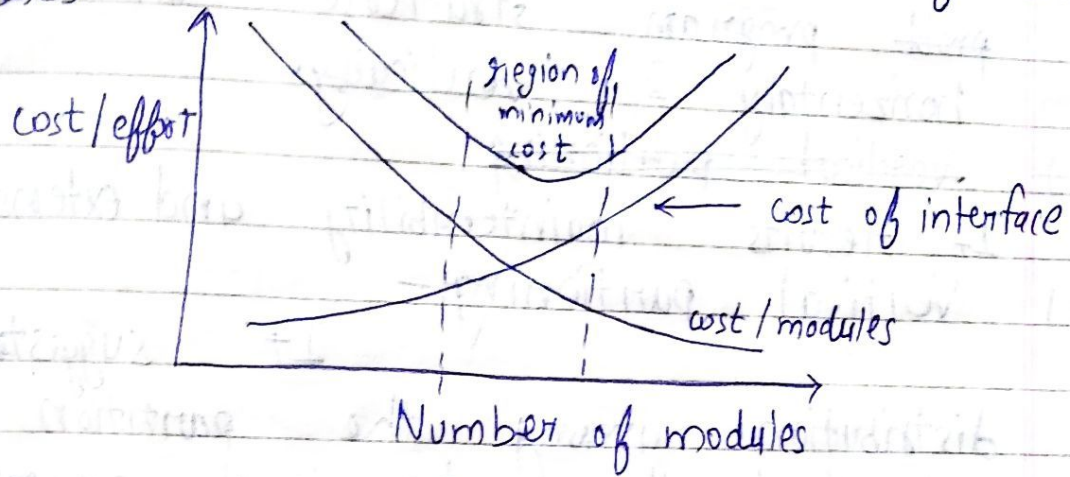


A scalar item

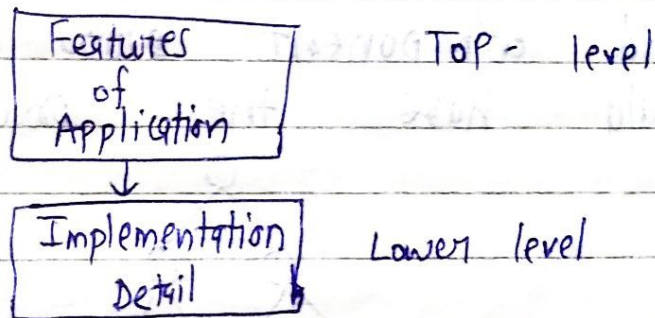


sequential vector

must be modularized into manageable pieces.
If the problem is modularized the cost will also decrease as the total no. of module increases.



2) Abstraction \Rightarrow



It is laying of application in a way that abstracting the complexity.

e.g.

We just mention the data we want to retrieve. We are not really ~~was~~ worried about how it will be retrieved.

e.g. Printing the documents on printer

Abstraction is all about hiding complexity and inner level details.

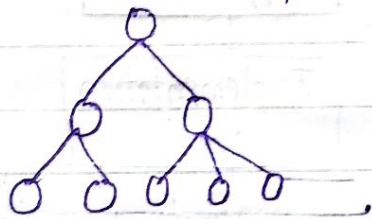
⑦ Structural Partitioning :— In the large scale problem, if architecture is a tree like structure then that ~~prob~~ program structure can be partitioned horizontally & vertically.

(i) ~~Vertical~~ partitioning

It results maintainability and extensionability.

(ii) Vertical partitioning:—

It suggests top down distribution among the partition component. such that the top level component should be assigned a control ~~functi~~ ~~respon~~ function and component low in the structure should have the processing responsibility.



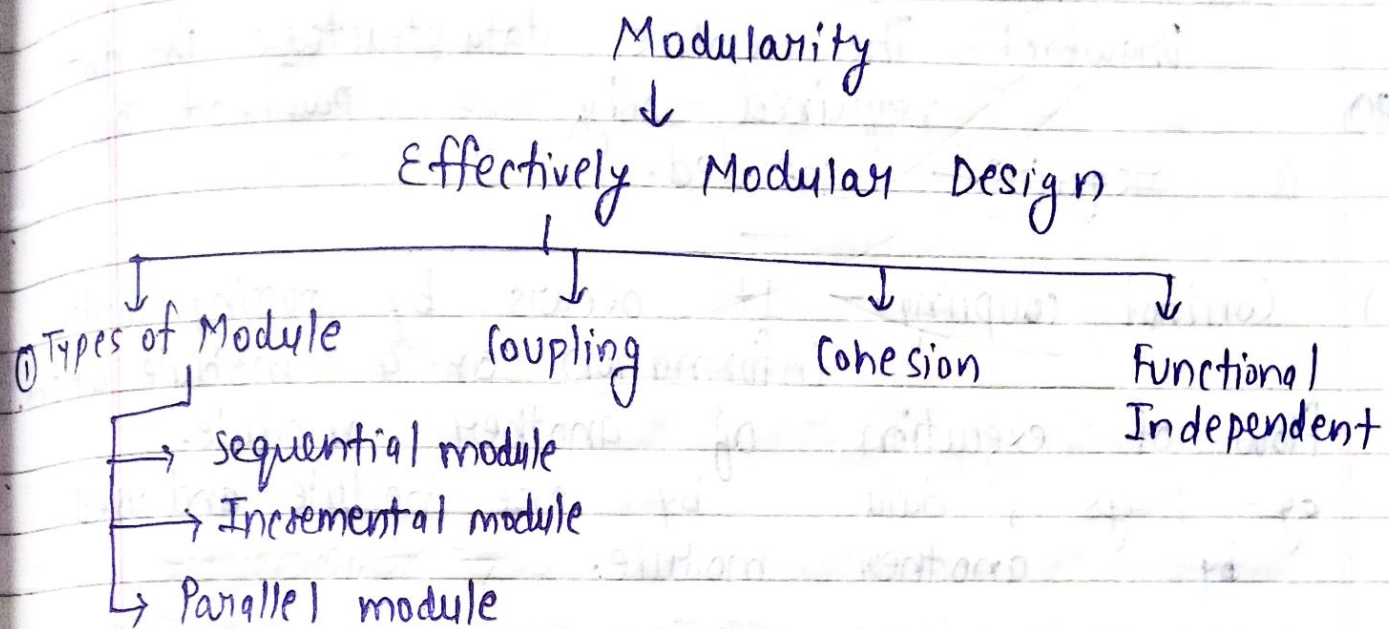
(ii) Horizontal partitioning:— sibling

⑧ Information hiding :— Each component hide a design decision from the others. Means component hide the internal details and processing from one another.

⑨ slw procedure :— It focus on the processing detail of each module individually. slw procedure provide a precisig specification of slw procedure including ~~sequence~~ except sequence of event

, exact decision

except ~~procedure~~ point, repetitive operations & even data organization / structure.



② Coupling: — Degree of measure of dependencies b/w two modules.

Trick → Did she conduct EXAM for Communication & Calculus

- | | | |
|-----------------------|--------------|-------------|
| (A) Data coupling | ↓ coupling ↑ | (desirable) |
| (B) Stamp coupling | | |
| (C) Control coupling | | |
| (D) External coupling | | |
| (E) Common coupling | | |
| (F) Content coupling | | |

(A) Data coupling: — Communication b/w modules occur by only passing the necessary data (no control information) —

(B) Stamp coupling: — This coupling occurs when a complete data structure is passed from one module to another.

Drawback — The entire data structure is not required only a part of DS is ~~required~~ needed.

(c) Control coupling: — It occurs by passing control information or a module control flow of execution of another module.
eg - Flags, data by one module and used by another module.

(d) External coupling: — Dependency of a module on another module which is present in a sw or h/w external to the system.

(e) Common coupling: — When two modules have common shared data or global data accessed by both.
Shared data makes error isolation difficult.

(f) Content coupling: — When one module changes data of another module


Module cohesion: — Degree to which elements of a module are functionally related.

high cohesion $\xRightarrow{\text{implies}}$ Low coupling

Cohesion is the glue that hold the module together.

* Strongly cohesive module implement functionality focusing on a single feature of a s/w
high cohesion is designed.

[Trick! — First semester of college provide tough & long curriculum]

- (i) Functional Cohesion
 - (ii) Sequential Cohesion
 - (iii) Communicational cohesion
 - (iv) Procedural Cohesion
 - (v) Temporal "
 - (vi) Logical "
 - (vii) Coincidental "
- 
- cohesion ↑

(i) Functional Cohesion: — If two operation present within the module perform the same functionality or they are part of the same purpose.

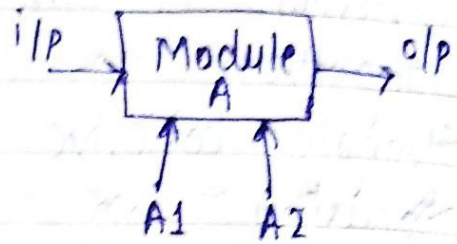
(ii) Sequential Cohesion: — In a Module, if 2 operation are such that

X output \rightarrow Y's input

X, Y \in same module

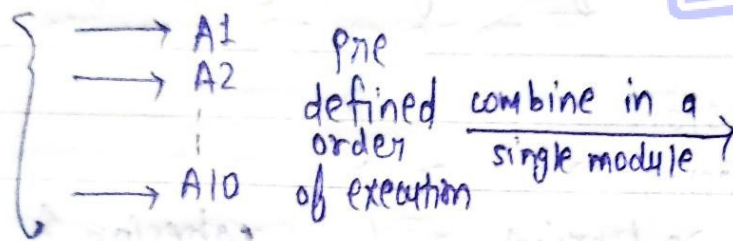
(iii) Communicational Cohesion: — Element inside the module that operate on same input data or contribute to same data.

A1: a set of instructions
A2: a set of instructions



Each element affects each other.

4. Procedural Cohesion:—

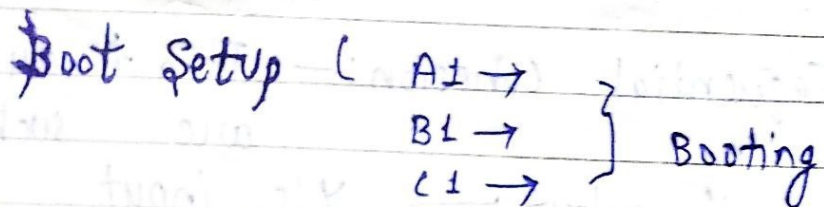


Module instruction task but to ensure particular order in which task are performed they are put in same module.

(v) Temporal Cohesion:—

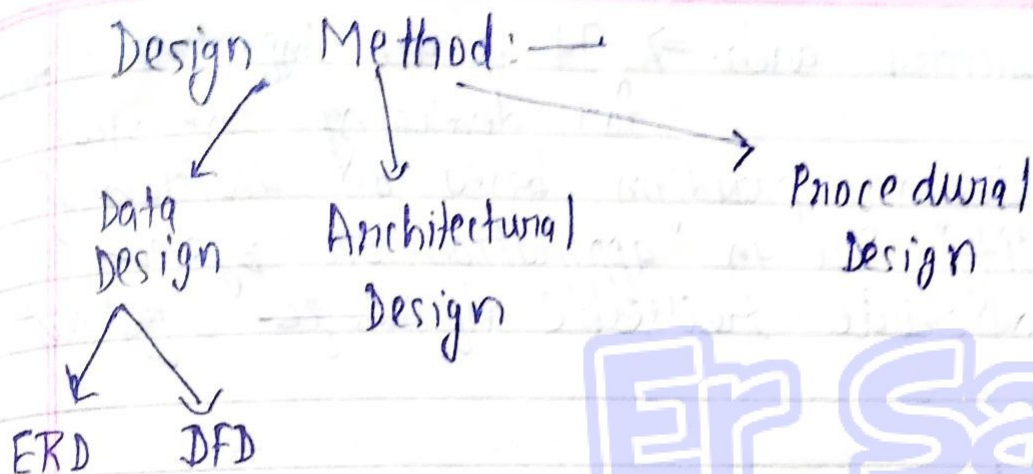
Instructions that must be executed during same time span are put together in a single module.
(Activities related in time)

eg -



(vi) logical cohesion:— When modules have logically similar instructions or elements.

7.1 Coincidental cohesion:— When instructions are not related or are very less related to each other and we group them in a single module.



Architectural design :- It represents the structure of data & program components. Data required to build a computer base system.

There are various style of software architectural :-

1. Pipes & filter
2. Object oriented arch.
3. Implicit invocation
4. Layering
5. Repositories
6. Interpreters
7. Process control

1. Pipes & filter \Rightarrow It is a simple arch. design that connects a no. of component that processes the stream of data. Each connected to the ~~no~~ next component in the processing pipeline ~~by~~ pipe.

It consists of one or more data sources.

The data sources are connected to the filter by a pipe.

The filter process the data they receive passing them to other filters in the pipeline.

The final data is received at a data sink.

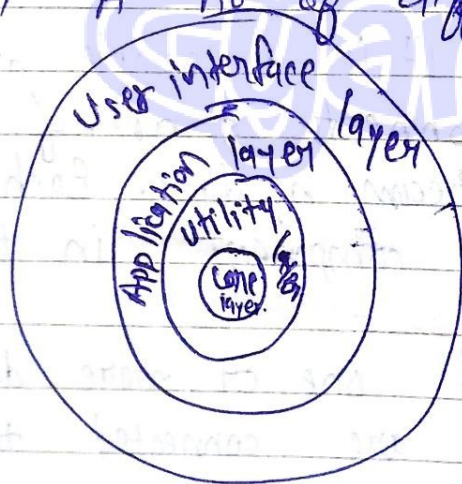
2. Object-oriented arch. \Rightarrow It is an important concept for developing the s/w.

It is a design paradigm based on division of responsibility. For an application or system in 2 individual reusable self ~~enough~~ sufficient object.

3. Implicit Invocation \Rightarrow

The design model for implicit invocation is event-driven based on the notion of broadcasting. To invoke a procedure, a component announces that one or more event has taken place. Then other components can associate a procedure with those events & the system invoke all such registered procedure.

4. Layered \Rightarrow A no. of different layer architecture are defined with each layer



performing a well defined set of operations.

At the outer layer, components will receive the user interface operation and at the inner layer, components will perform the operating system interface.

Intermediate layers do utility services and application s/w functions.

5.1

Repository \Rightarrow A repository arch: is a system that will allow several interfacing component to share the same data. Each component interfaces the same data set that is utilises system ~~and~~ write. Data manipulation taking place in one component will reflect in an "identical" ~~represen~~ of data in another component.
eg - Data Base Management system.

6. Interpreter \Rightarrow This design style is used for design a component that interpret program return in dedicated language. It mainly specifies how to evaluate line of program known as sentence or expression return in particular language.

7. Process control \Rightarrow The purpose of process control system is to maintain specific property of process outputs on near specify ^{process} reference value ~~res~~ called set point.
for ex - home heating & cooling system.
Uses a thermostat to monitor a air temp and then controls the furnace and air conditioner to keep air temperature within an acceptable range.

Procedural Design \Rightarrow

Architectural
design



Structure
program