Lecture No - 20

UNIT - 3

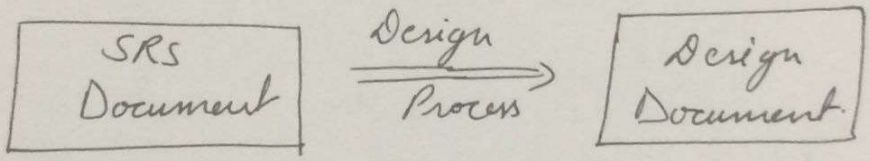
Software Design.

Content :- Basic concepts of Software design, Architectural Design.

→ **Software Design** -

→ It is the process of defining software methods functions, objects and overall structure and interaction of our code so that the resulting functionality will satisfy our customers / users requirements.

→ Activities carried out during design phase (called the design process) transform the SRS document into the design document.

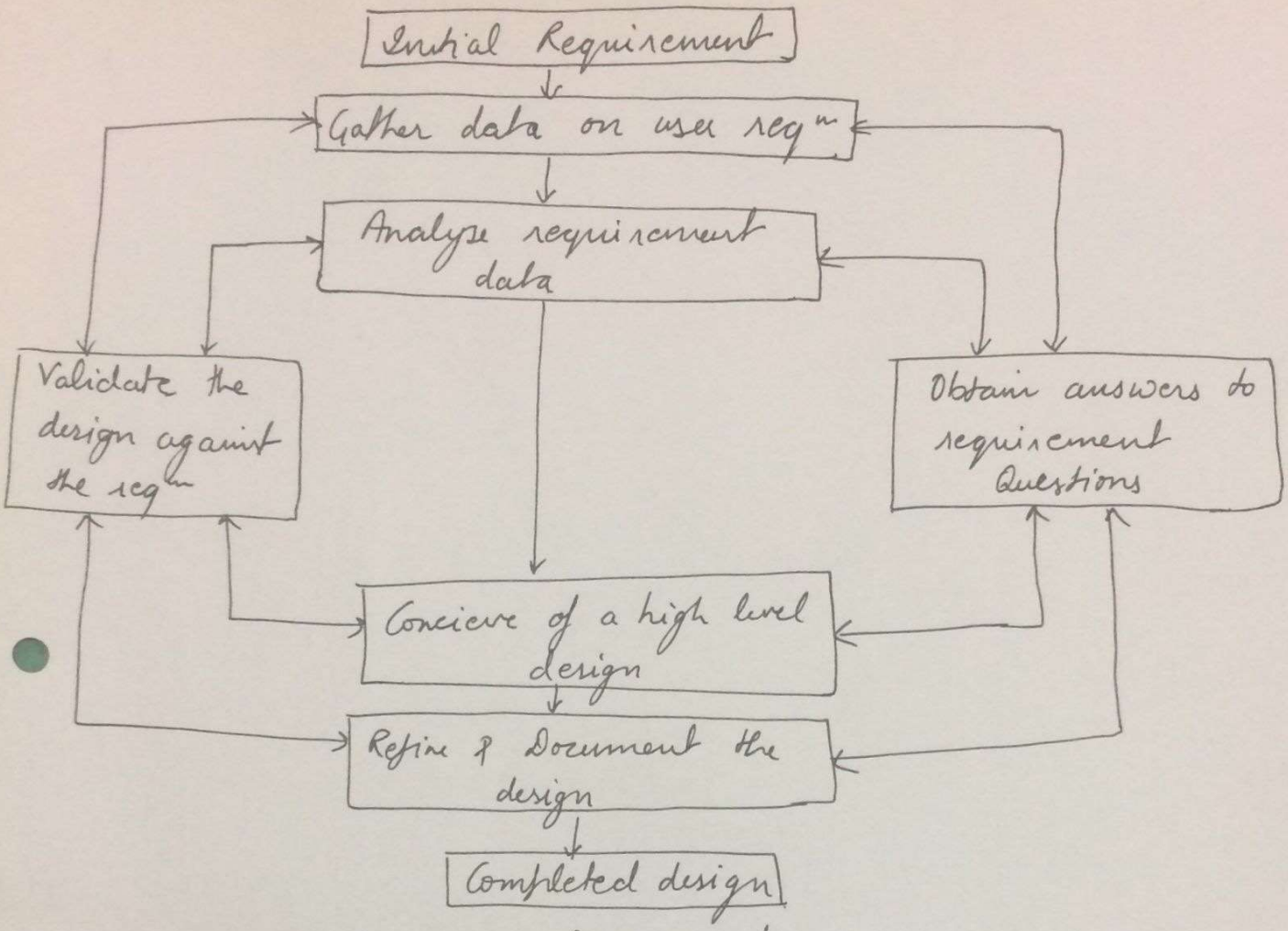


The design process translates / transforms the SRS document into a design document.

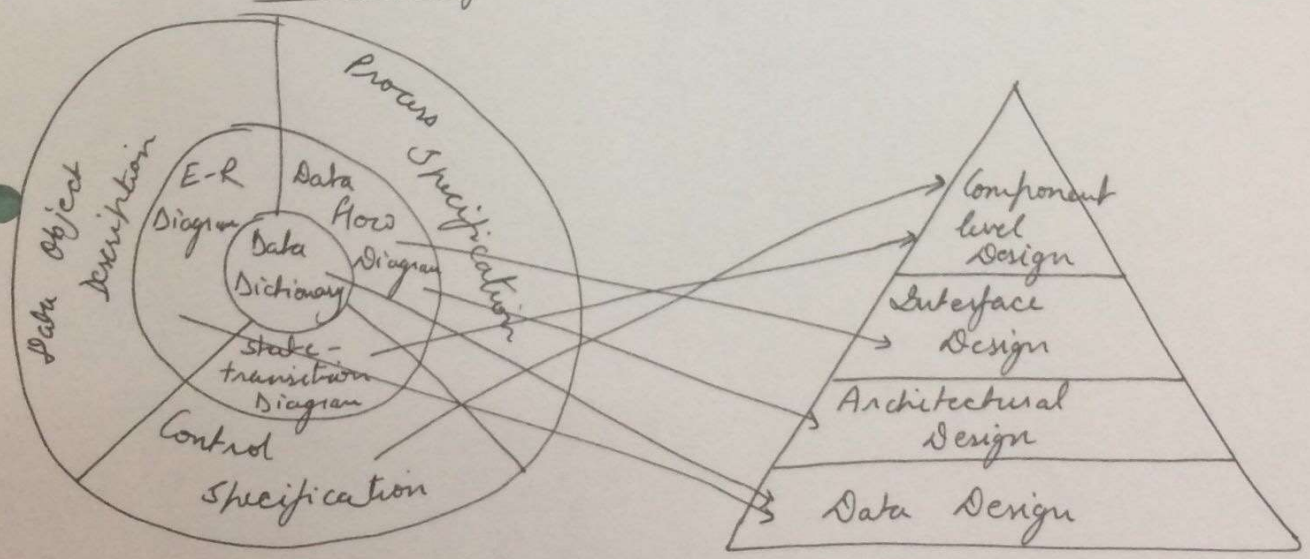
→ **Outcomes of a design Process** -

Following items are designed and documented during the design phase :-

1. Different Modules Required.
2. Control Relationship among modules.
3. Interface among modules.
4. Data structure of the individual modules.
5. Algorithms required to implement the individual module.



S/w Design Framework.



The Analysis Model

The Design Model.

Translating the Analysis model into Design Model.

Lecture - 21

Content:- Characterstics of S/w, Low-level design - Modularization

Characterstics of a good S/w -

→ Correctness
→ Understandability
→ Efficiency
→ Maintainability.

S/w Design Process Failure -

Types of S/w design process failures:

i) Non-Confirmation to user needs & design constraints.
ii) Inadequate design documentation.
iii) Workable & Poor design.

Low-level Design -

Modularization / Modularity -

→ It means dividing the S/w into seperate names and addressable components often called modules.

→ These modules are integrated to satisfy user reqn.

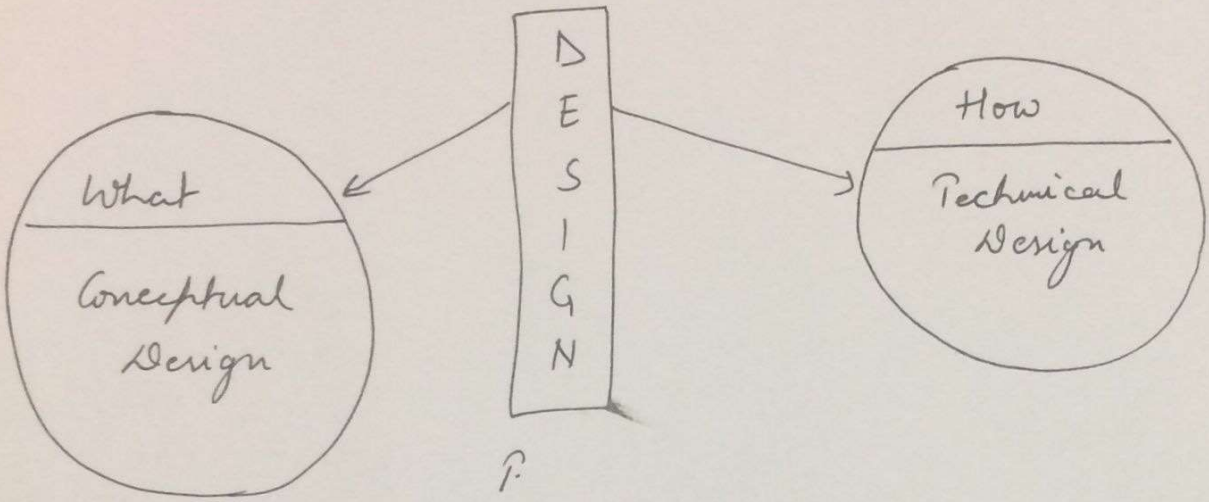
→ It allows a single program to be intellectually managed.

→ It is decomposition of big problem into manageable tasks called modules.

→ Each module can be understood seperately.

→

Software Design -



A two part design process

Conceptual Design - (Describes)

- → Where will the data come from?
- → What will happen to the data in the system?
- → How will the system look to the users?
- → What choices will be offered to users?
- → What is the timings of events?

Technical Design - (Describes)

- → H/w Configuration
- → S/w Needs
- → Communication Interface.
- → I/O of the system
- → S/w architecture.
- → N/w architecture.

Advantages of Modularity -

- → Design clarity
- → Ease of implementation
- → Reduces complexity
- → Eases debugging & testing.
- → Eases documentation
- → Eases maintenance of s/w product.
- → changes can be accomodated
- → Reuse of modules.

Properties of Modularity - (Modular System)

- → Well defined Sub system.
- → Well defined purpose.
- → Can be seperately ~~combined~~ compiled and stored in a library.
- → Module can use other module.
- → Modules should be easier to use than to build.
- → Simpler from outside than from inside
- → Well defined interfaces.

Need for Modularity -

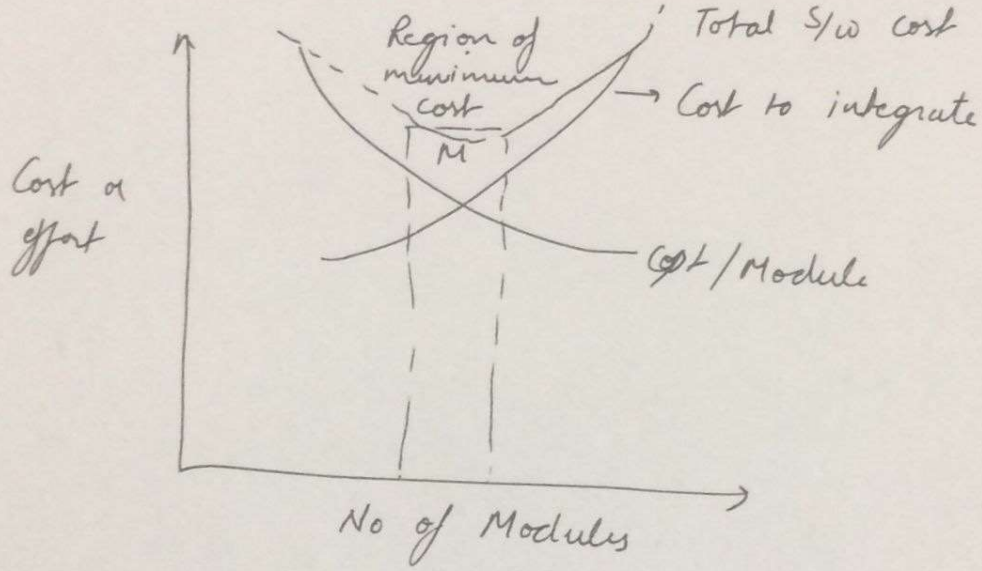
Primary motive is that a s/w should be easy to manage & fairly easy to enhance.

It is required in all stages of s/w development

- → Design Phase
- → Debugging
- → Testing
- → Maintenance
- → Independent Development.
- → s/w Reuse.

If we subdivide S/w indefinitely the effort required to develop it will become negligibly small.

However as the no. of modules grow, the effort (cost) associated with integrating the modules also grow.

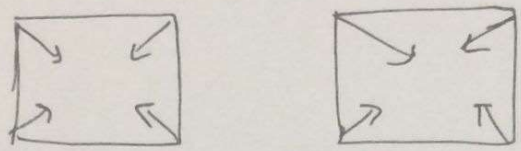


Lecture No - 22

Content :- Coupling & Cohesion Measures

Cohesion -

→ It is a measure of the degree to which the elements of a module are functionally related.



Cohesion in Module = Strength of relations within the modules.

→ Greater the Cohesion, better the ~~program~~ program design.

Types of Cohesion - (7 types)

① Co-incidential Cohesion -

→ Unplanned & random cohesion.

→ result of breaking the program into smaller modules.

② Logical Cohesion -

→ When logically categorized elements are put together into a module.

③ Temporal Cohesion -

→ When elements of module are organised such that they are processed at a similar point in time.

④ Procedural Cohesion -

→ When elements of a module are grouped together which are executed sequentially in order to perform a task.

⑤ Communicational Cohesion -

→ When elements of module are grouped together which are executed sequentially and work on same data.

⑥ <u>Sequential Cohesion</u> -

→ When elements of modules are grouped together because the output of one element serves as the input to another and so on.

⑦ <u>Functional Cohesion</u> -

→ It is considered to be the highest degree of cohesion and it is highly expected. Elements of module in functional cohesion are grouped because they all contribute to a single well-defined function. It can also be reused.

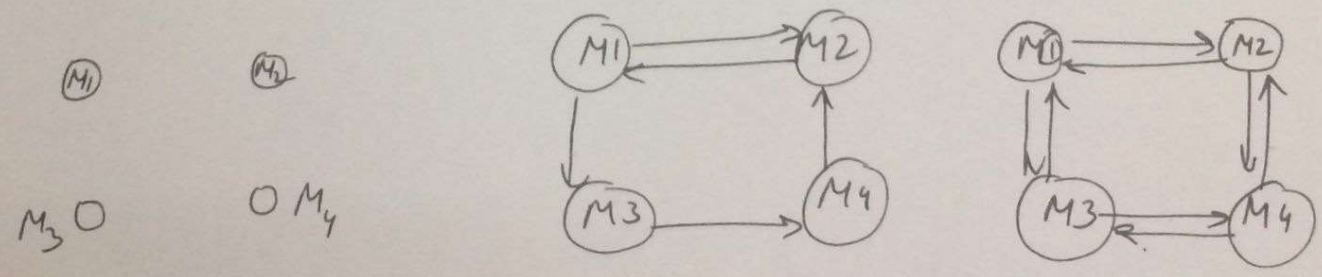
<u>Coupling</u> -

→ It is a measure of degree of interdepence or interaction b/w the two modules.

→ Two modules are said to be highly coupled in below two conditions

① If the func calls b/w two modules involves passing large chunks of shared data.

② If the interactions occur through some shared data,



<u>No Coupling</u>
(No dependency)

<u>looscly Coupled</u>
(some dependencies)

<u>highly Coupled</u>
(Many Dependencies)

Properties of Coupling -

→ Non Negativity

→ Null Value

→ Merging of Modules

→ Designing Module Activity.

Types of Coupling - (5 types)

① Content Coupling

→ When a module can directly access or modify or refer to the content of another module

② Common Coupling -

→ When multiple modules have read and write access to some global data.

③ Control Coupling -

- Two modules are called controlled coupled if one of them decides the function of the other module or changes its flow of execution.

④ Stamp Coupling -

→ When multiple modules share data structure (common) by means of passing data (as parameter). If a module passes data structure as parameter.

Ideally no coupling is considered to be the best and high Cohesion.

Lecture No -23

Contents :- Design Structure Charts :- Pseudocode, Flow charts.

For a function oriented design the design can be represented graphically or mathematically by the following ways :-

* Data Flow Diagrams
* Data Dictionaries
* Structure Charts
* Flowcharts
* Pseudocodes.

Structure Charts -

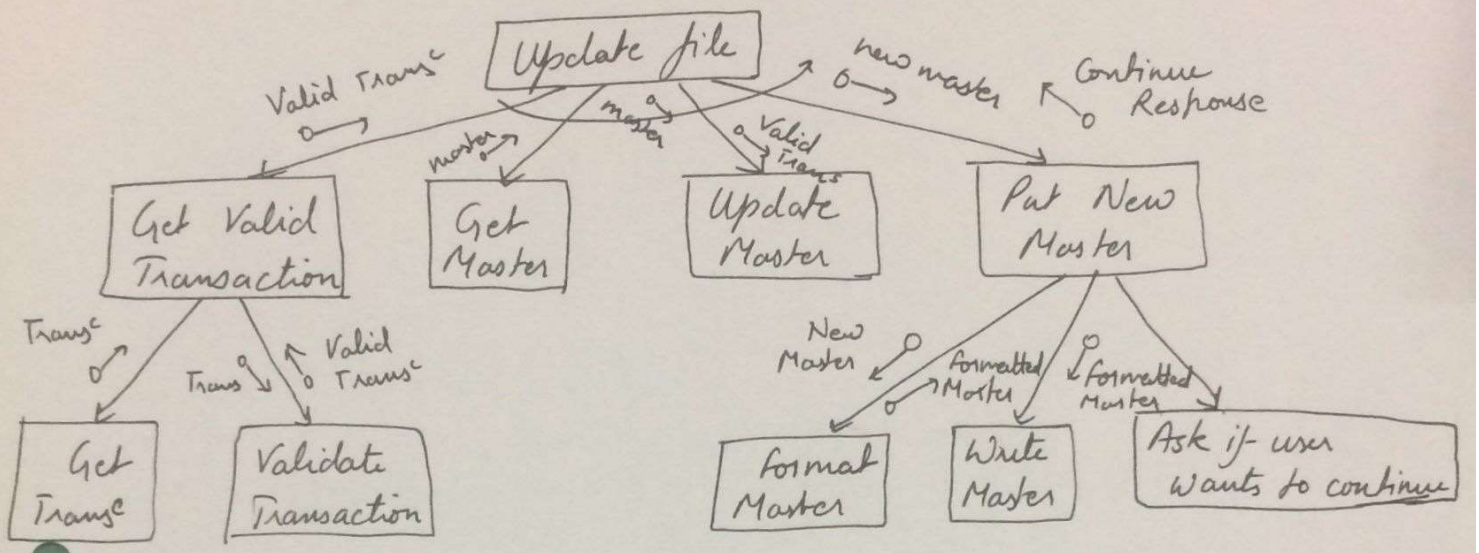
→ It is a graphical representation of function oriented design.

→ The structure of a program is made of modules and interconnection b/w modules.

Notations - |

* o→ Data
* o→ Control
* [Module]
* [| library Module |]
* (Physical Storage)
* [ ] Repetitive cell of Module
* ◇ Diamond - for conditional cell of module.

Example of Structure Chart - for Updating a file



Pseudocode -

It can be used in both the preliminary & detailed design phases. Using pseudocodes the designer describes system characteristics. using short, concise, english language phrases that are structured by key words such as if-then-Else, do-while, While-do, End.

Flowchart -