[ch7] Design concept and principles

⇒ S/w design can be defined as the process of developing the structure of any s/w system, on top of which whole software system has to be built. The process of building the framework that helps in the development of a successfully working full fledged s/w systems is known to be s/w design.

→ It can be defined as the process of translating the s/w requirement specified in the requirement gathering stage, into blue print for the development of application.

→ S/w design is one of the first three job done in development process.

There are 4 types of s/w design

ⓐ Data design ⓑ Architectural design ⓒ Interface design
ⓓ Procedural design

↳ Each components of the analysis model (elements) gives information that will be helpful for building the four design model

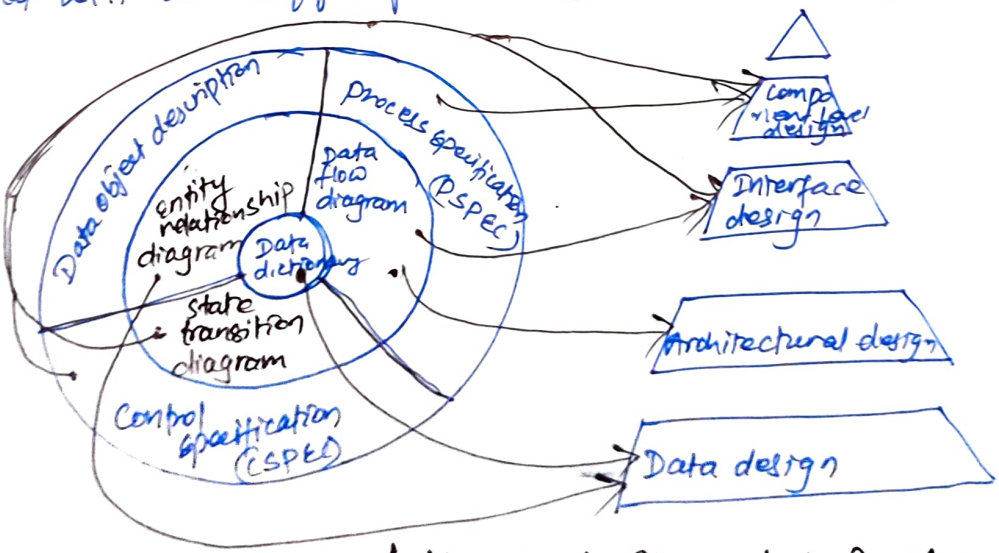


Fig:- Translating analysis model in s/w design

Here the above figure shows the flow of information from the s/w analysis model while designing s/w.

(a) Data design

↳ Data design transforms the information model created during analysis into the data structure required for implementing in s/w design. The data objects, its attribs. life and the relation between those data objects provides a base for data design activity.

↳ Data design is often included with Architectural design and component level design.

(b) Architectural design

↳ Architectural design defines the relationship between major structural element of s/w, the design patterns, and the constraints.

Here "the design" pattern can be used to achieve the requirements defined for the system.

constraints affect the way in which design patterns can be applied.

(c) Component level design (31)

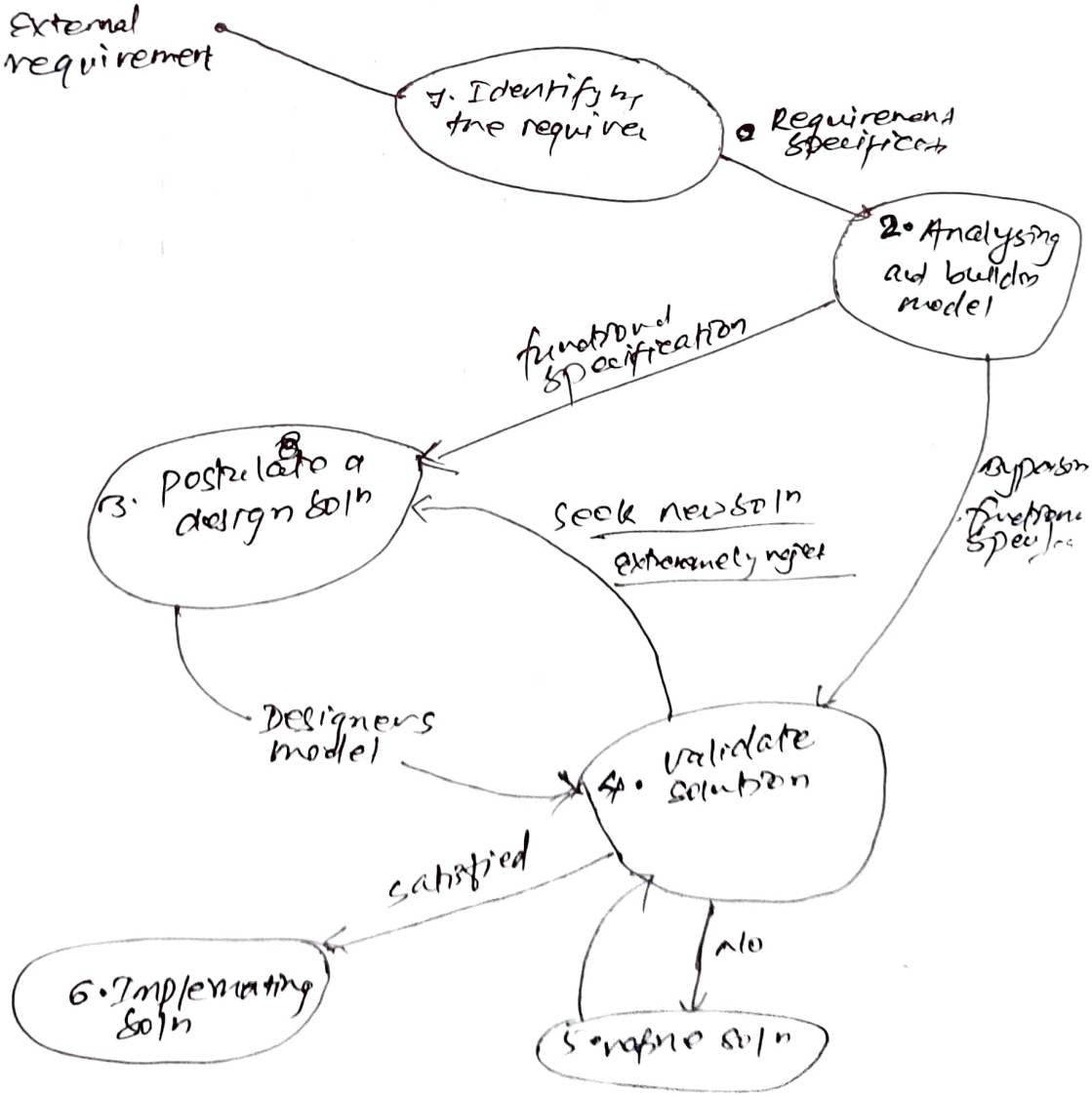
↳ Component level design transform the structural elements of s/w architecture into a procedural description of s/w. The info obtained from PSPEC, CSPEC, state transition

Importance of s/w design

↳ (a) etc

Design process

↳ It is the process of converting the requirement in to a s/w designing blue print.



Design principles

They are the principles that a successful s/w design must be following.

(a) s/w design should not suffered from tunnel vision.
(b) s/w design should be traceable to analysis model.
(c) s/w design should not reinvent the wheel.
(d) s/w design should minimize the intellectual distance between s/w and problem.
(e) design should exhibit uniformity and integration.
(f) design should be structured to accomodate the change.
(g) coding is not design and design is not coding.

Design Concepts

↳ Design concepts can be defined as the fundamentals that acts as a base for the further sophisticated s/w design. The design concepts helps in determining the criterion for partitioning the s/w design into multiple components on which different function specification can be used for enhanced s/w design.

Some of the factors which needs to be focused while designing s/w are:

(a)
- → Abstraction (A)
- → Refinement (R)
- → Modularity (M)
- → s/w Architecture (S)
- → Control Hierarchy (C)
- → Data structures (D)
- → s/w procedure (P)
- → Information hiding (I)

→ Abstraction
→ Refinement
→ Modularity
→ System architech
→ control Hierarchy
→ S/w design
→ O.S.
→ Information hiding

What is compter level design ?

↳ A complete set of s/w component is defined during the architechural design but there are some internal D.S and processing details of each componetn which are not represented at a level of asstraction that is close to code.

So, Component level design defne the d.s, algo, interface charastics allocated to each componna.

objective of complex level design

(a) Identifying the problem domains design class
(b) Identifying the infrastructure design class
(c) Elaborating the design class
(d) Elaborate the behavioural representation, deployment diagram.
(e) Describe persistent data source.

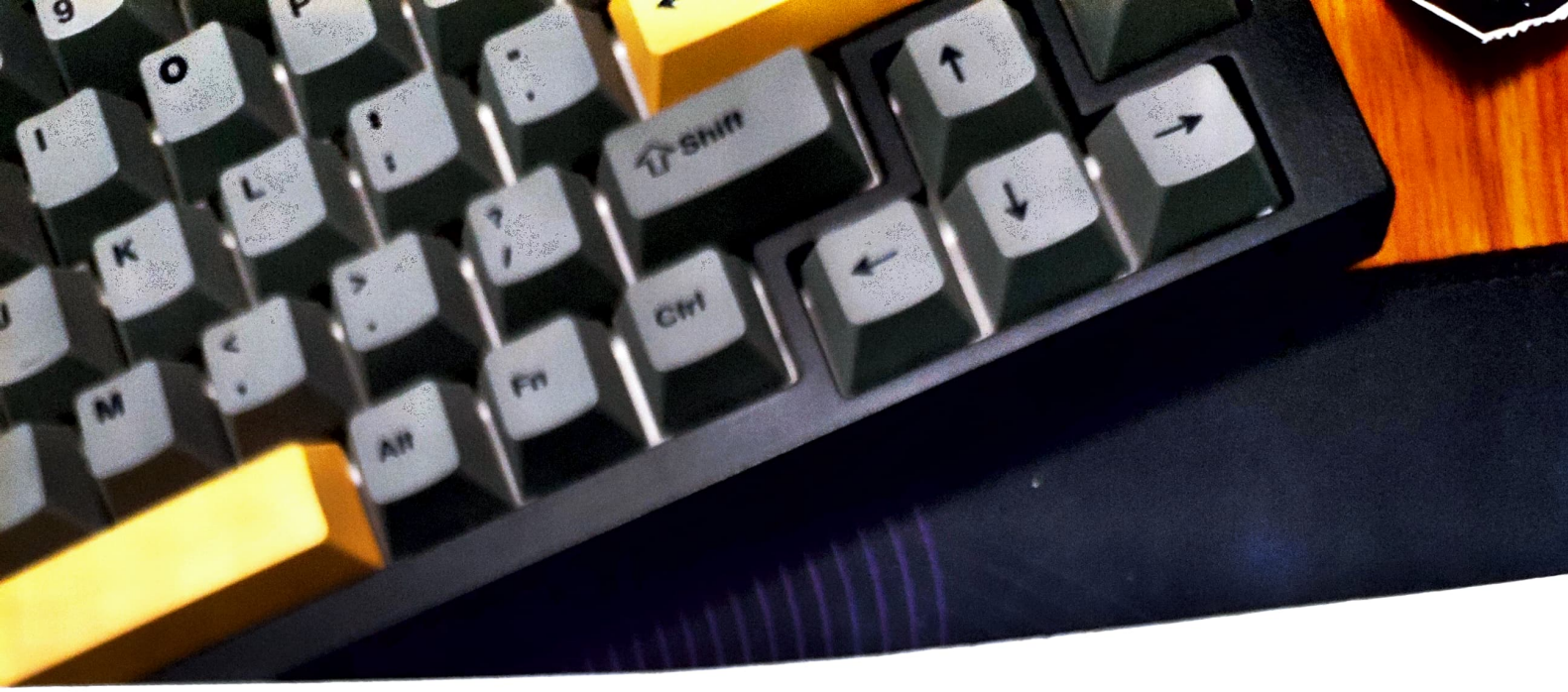
Architechural style

↳ Architechural style defines system catogory that encompasses :

(a) Set of components that performs a function required by a system. Component ↳ part → funct

(b) Set of connectors, that enables communication, coordination and cooperation among componentts,

(c) Semantic models that enables developer to understands the overall property of system.



Generally used Architectures styles

(a) Data centered Architecture
(b) Data flow Architecture
(c) Call and Return Architecture
(d) Object oriented architecture
(e) Layered Architecture

Ch 7 SEE

Q. What is architectural design of s/w ? Explain various architectural styles in brief.

Ans: Architectural design can be defined as the type of s/w architecture design that defines the relationship between major structural elements of s/w, the "design patterns" that can be used to achieve the requirement which have been defined for the system.

Architectural styles describes the system category which encompasses:

(a) Set of component that performs a function required by the system.

(b) Set of connectors that enable "communication, coordination, and coorperation" among components.

(c) Constraints that describes how components can be integrated to form the system.

(d) Semantic (meaningful) model that enable a designer to understand the overall property of the system by doing analysis of known properties.

Generally there are 5 types of architectural styles

1. Data centered Architecture:

Here in this architecture, a data store (file or db) resides at the centre of the architecture and is frequently accessed by other components that add, delete, update or modify the data within the store.

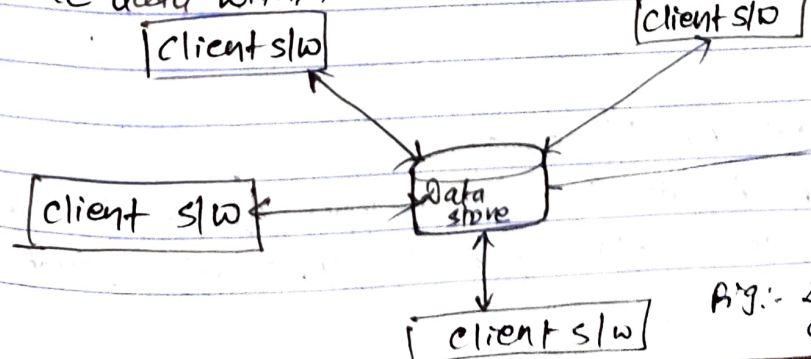


Fig:- Data centred architecture.

How the existing client s/w component can be changed or new s/w components can be added without concern about another client. Hence, Data centered architecture promotes the integration.

Centralized server system are the best example of data centre architecture, where the one server does the response of all the s/w client that request the D.B.

(b) Data flow architecture

↳ It is one of the type of architectural style that is applied when the i/p data are to be transformed through a series of computational components into o/p data. Example: A pipe and a filter pattern where the filter transforms the data independently of other components and pipes are used transmitting data.

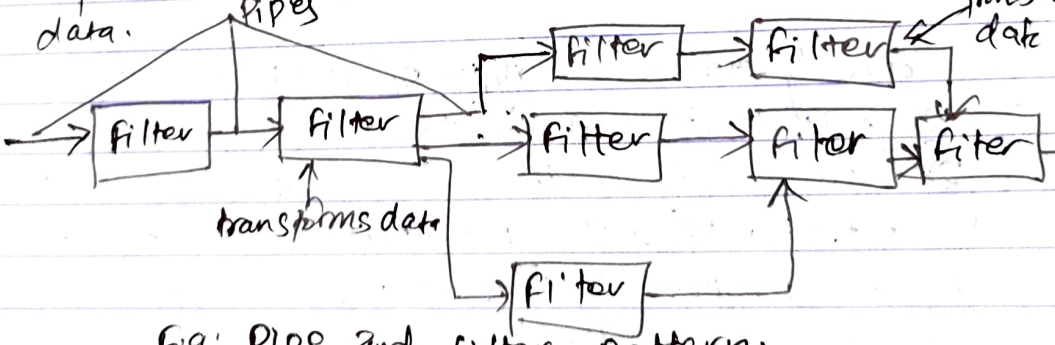


Fig: Pipe and filter pattern.

(c) Call & Return Architecture

⇒ Call & Return Architecture is a type of architecture which is used in a subprogram or subroutine that does

the job like
designer fr
easier to

(a) Main
↳ Decompo
a main
compone
compone

Controller
Subpro

pp subprog

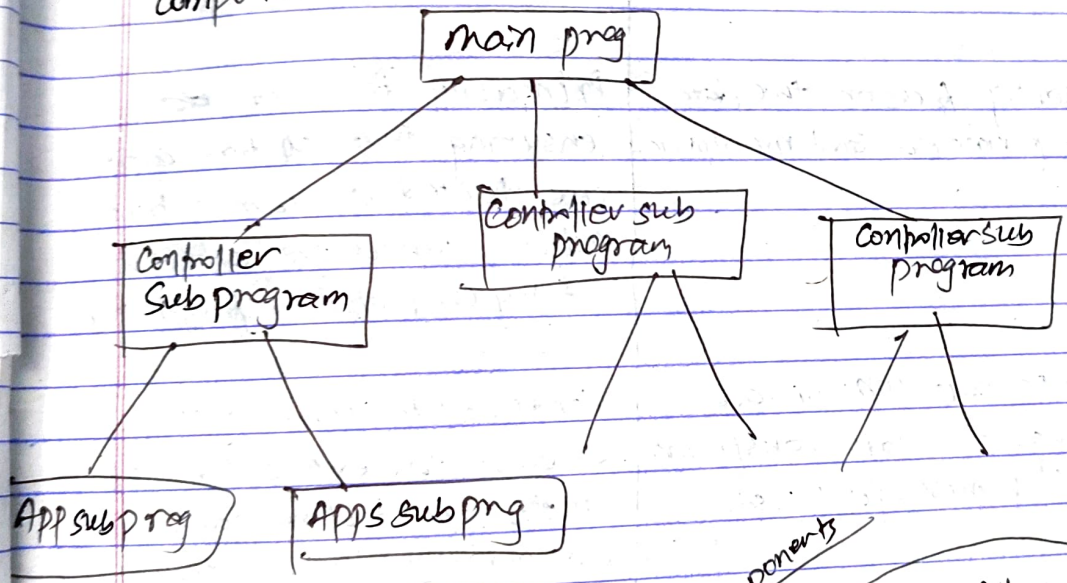
(4) La

↳ Here
are
appro
prog
to

the job like a method in programming. It helps the s/w designer to achieve the structure which is relatively easier to modify and scale.

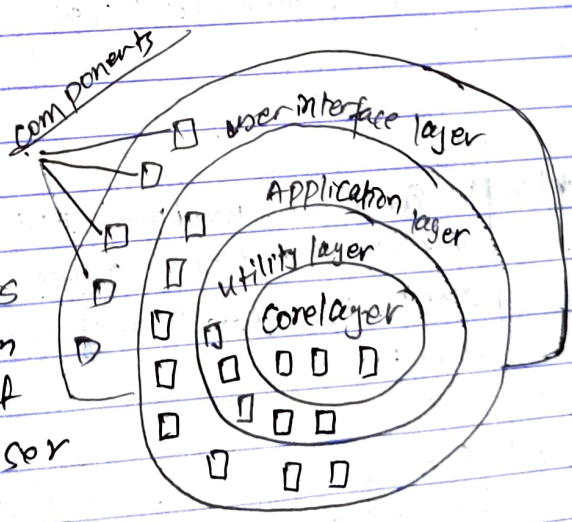
③ Main program / sub program architecture

↳ Decomposes the function into a control hierarchy, where a main program invokes the number of program component, which in turn may involve still other components.



④ Layered architecture

↳ Here no. of different layers are defined, each of them accomplishing operations that progressively becomes closer to m/c instruction set.



	Transform mapping	Transaction mapping
Def:	Transform mapping is the design approach where the data or information is transformed from one representation to another during interaction between components	Transaction mapping is the design approach where each interaction between component is considered as transaction
Focus:	Primarily focus on the data transformation and manipulation.	Primarily focus on ensuring that all transactions are treated as atomic transaction, which are either fully completed or fully rolled back
Communication style	Components communicate through the data transformation & message passing	Components communicate through the exchange of messages and responses, mimicking transactional behaviour
Error handling	Error are handled within the component themselves during the data transformation process.	Error are handled using transactional semantics, enabling rollback or recovery in case of failure.
Application	Often used in scenarios where data ~~needs to be translated or adapted~~ integration or conversion is required	often used in distributed system financial apps or any other system needing consistency guarantee

complexity	Generally simpler to implement as it focuses on data transformation	Can be complicated to implement especially when dealing with distributed system.
Examples:	Data format conversion (XML to JSON), data validation, enrichment.	Money transfer between bank, reservation system for flight booking.

2020 fall 7c

Mapping Requirements into S/W Architecture

↳ As we have already know that requirements are to be mapped into design and there is no any specific technique of mapping.

- Mapping requirements into S/W Architecture (Architectural Mapping) is a process in which the Dataflow diagram (DFD) is transformed to a S/W Architecture. After the construction of level 2 DFD, how our DFD could be converted into S/W Architecture, this process is called as Architectural Mapping.

There are two types of information flow from the input to output, which drives the mapping requirements:
(a) Transform flow (b) transaction flow.

Transaction flow is the flow where one single data items triggers to take one ~~or many~~ more than path

Transform flow is the flow where the data flow along a

st.

end number of [redacted] into paths. Transform flow is divided into 3 parts:

(i) Incoming flow:
Flow where the data comes from external entity towards the system is called incoming flow.

(ii) Transform Center
↳ Transform center is the region that passes the input information towards the output region.

(iii) Outgoing flow
↳ The path that carries the final processed data outward to the s/w is called outgoing flow.

There are different steps involved in Architectural mapping:

Step 1: Review the fundamental system model
Here we just have to talk about the fundamental system. No any clarity is provided, about what is happening in the system. Here only level 0 DFD is drawn.

Step 2: Review and refine data flow diagram for the s/w
↳ Here the level 0 DFD will be converted to the level 2 DFD, providing more clarity of the system.

Step 3: Determine whether the DFD has transaction or transform flow characteristics

step 4: Isolating the transform center by specifying incoming and outgoing flow boundaries.

step 5: Perform "first level factoring".
Till now we have only build up to level 2 DFD and in this steps the DFD will be converted to s/w architecture with first level factoring that only shows the essential components.

step 6: Perform "second level factoring" where we gotta explain about the types of input and types of o/p.

step 7: Refine the architecture with design heuristic.

2cfall 19 what do you mean by design model? List any six design principles. Explain call and return architecture.

↳

Design model in s/w design can be defined as the prototype which reflects the final end product that the clients gets at the end of the project. The design model provide the basic fundamental idea of the final product and it helps the whole s/w dev team to visualize the product. Design model helps in demonstrating the upcoming functional behaviour of system.

The six design principles are:

(a) The design process should not suffer from 'tunnel vision'. i.e. a designer should not focus always on smaller things.

(b) The design should not reinvent the wheel, means a designer should not integrate the functions

(c) The design should exhibit uniformity and integration.

(d) The design should be capable of accomodating the change

(e) Design is not coding and coding is not design

(f) The design should be reviewed to minimize the conceptual (meaning ful) errors.

---

19 fall 5a **Explain SW design process & principles**

↳ The SW design principles are the rules (criteria) which must be followed while designing the SW system Architecture. The SW design principles plays a crucial roles in the design and development of a SW having good functionality and provide the exact output required by the client.

They are the fundamental guidelines and concept that guide the process of creating high quality, maintainable and efficient SW systems.

Some of the basic design principles that needs to be followed while S/w developments are:

1. (i) The design process should not suffer from tunnel vision. It means when developing a large and complex S/w system designer and developer should not always focus on a smaller component instead they need to take care of whole system.

(ii) The design process should be uniform and integrity. It means the whole development and design process be uniform enough to maintain consistency throughout. It should integrate the component seamlessly.

(iii) Design process should not reinvent the wheel. It means the design and development of those system which are already existing and well established, is not preferrable.

(iv) Design process should be capable of accomodating the changes. When a certain changes is need, to tackle those situation design must be flexible enough.

(v) Design is not coding & coding is not design

(vi) The design process must be reviewed to minimize the errors.

⇒ S/w design is an iterative process through which requirements are translated into a "blue print" for constructing the s/w systems. In the design process firstly high level design or abstraction level design is drafted. And with each iteration details in the design is iterated and finally low level design is achieved.

Lets look into some of the points that tells how a good design should be

(a) It must be addressing all the customers requirements and problems and provide the solution.

(b) It must be readable, understandable guide for developers and for other concerned personals.

(c) It should exhibit hierarchical nature.

(d) It should obtain certain level of data and procedural abstraction.

(e) User Interface must be minimal and user friendly - to use & navigate along multiple pages.

17sp 4.a

Component level design

↳ It is a type of s/w design (s/w design) modules which focuses on breaking the whole s/w design into smaller components which acts as an individual modules.

Here the ~~reusables~~ is a modular, small, portable replaceable component and reusable set of well defined functio-

nalities. This component level design describes the communication, interfaces, algorithm and functionalities between each and every components of whole Yw system. Notations like DFD, Activity diagram, Conditional Notations, Tabular forms etc are used in component level design.

Some of the benifits of doing component level design are:

(a) It makes the module reusable.
(b) Ultimately it reduces the cost as the component or modules are reusable.
(c) It is more reliable as the client get interaction with each module of the system.
(d) Maintainability of the system will be easier since the specific component needs the maintain to be performed.
(e) The component level design describes each components functionality more clearly.

Steps need to carried to use component level design

1. Identify all the design class that corresponds to ~~the~~
   (a) problem domain
   (b) Infrastructure domain like GUI class, DBMS class, OS communication class
2. Elaborate all the design class that are not required as reusable components.
3. Describe the persistent data source and identify class that can manage them.

⇒ Develop behaviour and functionalities of the class.
⇒ Elaborate the deployment diagram. eg: H/W, O.S, server
⇒ Factor every component level design representation.

Q fall 65 "Don't rush through it! design is worthy the effort". Justify the statement with some design principles.

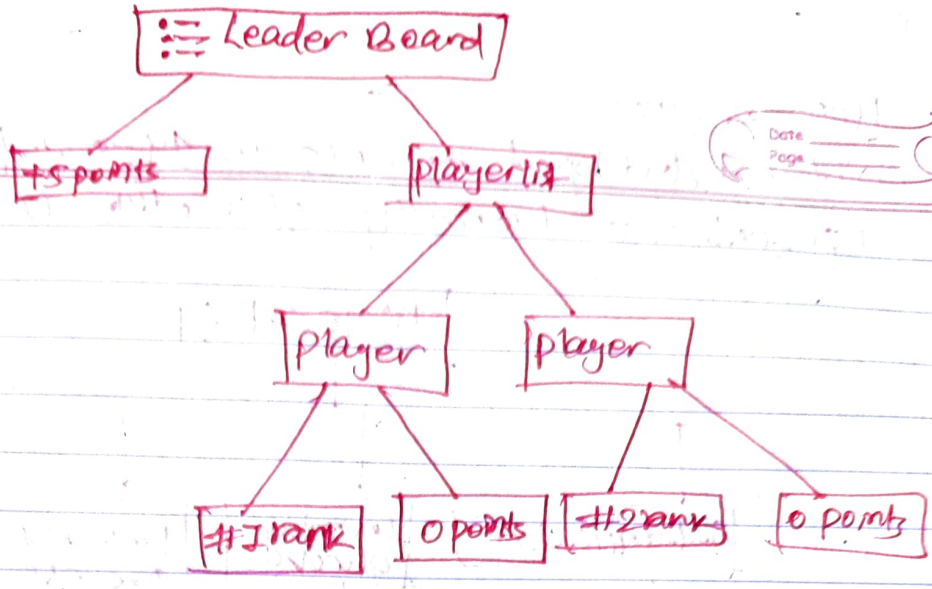
↳

Design Notation

↳ A Notation is a system of using symbols or signs as a form of communication or a short written Notes.

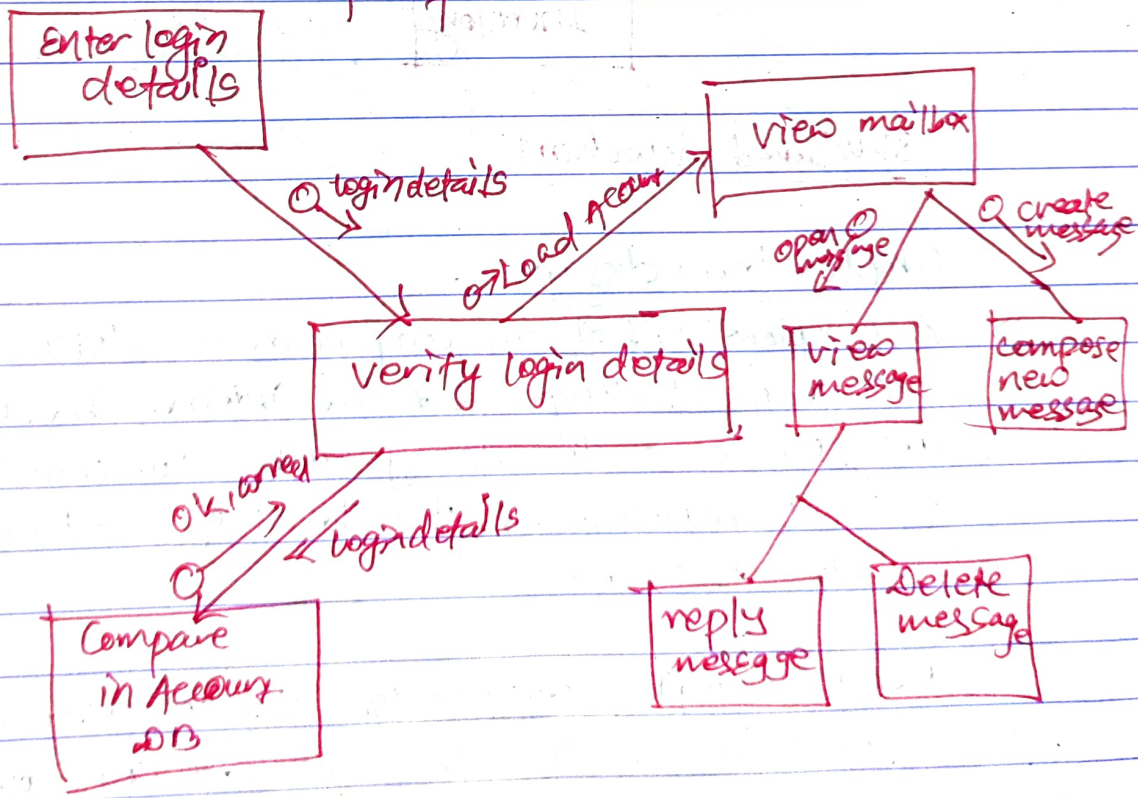
The design notation are types of notation which are used when planning and should be able to communicate the purpose of s/w program without the need of formal code. There are multiple types of design notations:
(a) DFD (b) Pseudo code (c) Structure charts
(d) structured flow charts (e) Decision tables
(f) HIPO diagram etc.

The data flow diagram is a graphical representation of flow of data through an information system. It is capable of depicting incoming data flow, outgoing data flow and stored data.



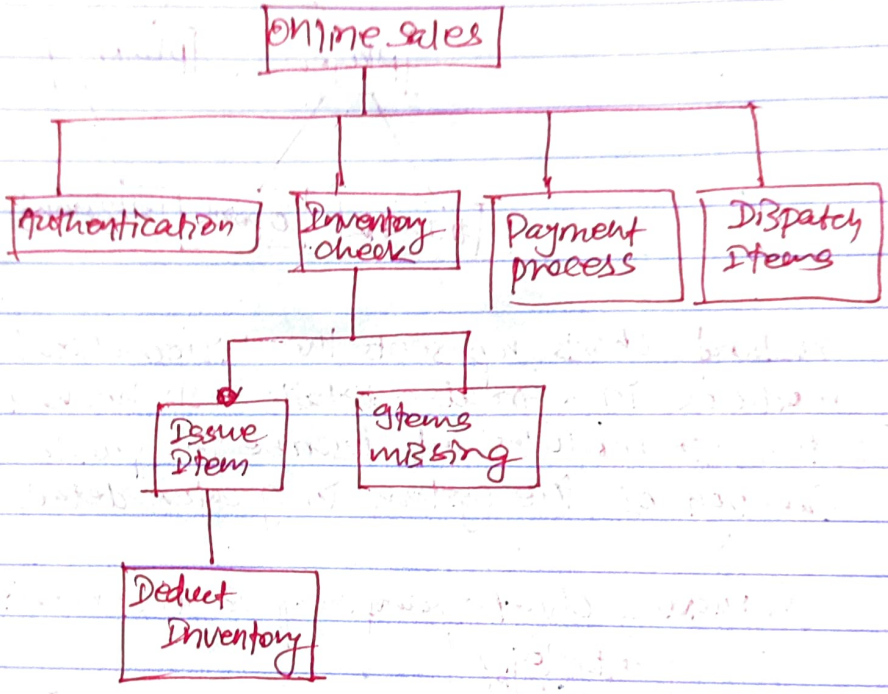
Structural charts represents the hierarchical structure of modules in a greater details. It breaks the entire system into modules and describe the function and sub function of the system in greater details.

Eg: Structure chart example of email server:



HIPO Diagram

↳ It was developed by IBM in 1970 and represents the hierarchy of modules in S/W systems.



Structured Flowchart

Complex level design

↳ A complete set of s/w components is defined during architectural design but the internal mechanisms, d.s, processing details of each s/w components are not represented, at a level of abstraction which is close to code.

Hence complex level design defines the d.s, algos, interface, characteristics and communication mechanism allocated to each component.
The components are the set of collaboration classes.

Design classes

The objective of complex level design are:

(a) Identify the design classes in problem domain
(b) " the infrastructure design classes
(c) Elaborate the design classes.
(d) Describe persistent data sources.
(e) Elaborate behavioural representation.
(f) " deployment diagram.
(g) Refine design and consider alternatives.