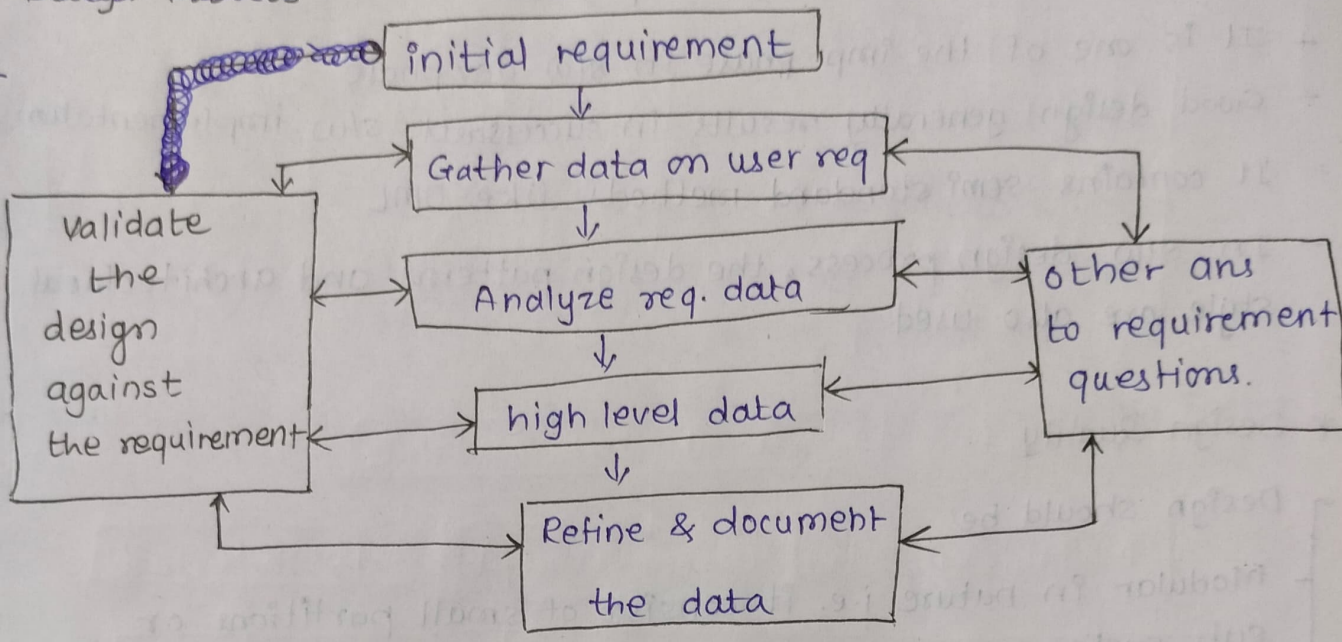
// Hare Krishna //

UNIT III

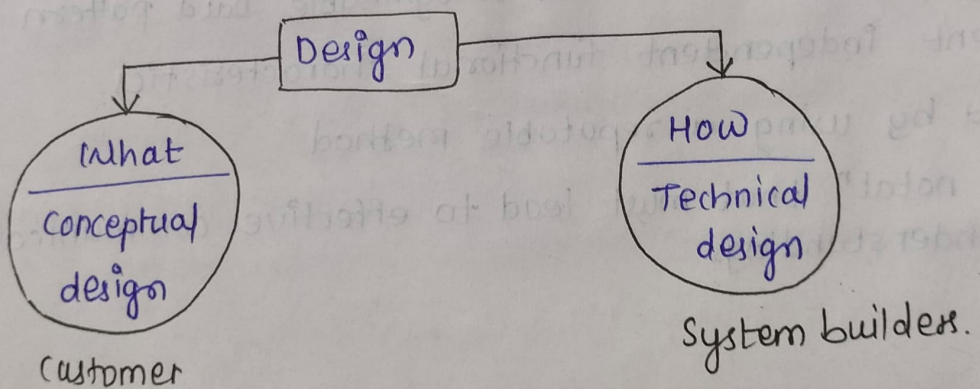
1) Design Engineering :

- Is to produce a model or representation that should show the fimness, delight and cammodity.
- S/W design is more creative than analysis.
- Prblem solving activity.

* Design Process



⇒ Conceptual desinger & Technical designer.



Customer

System builders.

- The design needs to be correct and complete
- understandble
- Right level
- Maintainable

Includes :
- Algorithms
- Diagrams
- formulae design
- low level component.

- It is one of the imp phase in s/w dev phase
- Good design generally results in successfull s/w implementation
- It contains semi standard methods like UML
- In s/w design process, the design patterns and architectural style are also used.

Design Quality :

Design should be

Modular in nature i.e. it consist of small partitions or sub system.

Represent data, architecture, interface & components in distinct way

Contain appropriate DS and recognisable Data pattern.

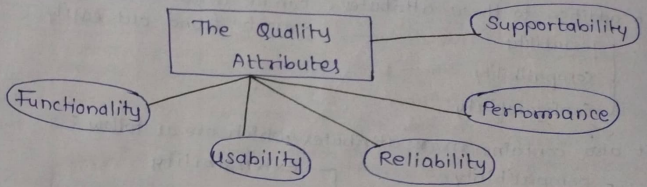
Represent independent functional characteristics.

Derived by using a reputable method.

use a notatn that must lead to effective communication and understandings.

Quality Attributes

FURPS



1) Functionality :-
- Is an imp aspect of any s/w system
- It is generally evaluated by the feature set and capabilities of that s/w.

2) Usability :-
- Is best evaluated by considering following factor
  - Human factor
  - overall aesthetics
  - Consistency & documentatn

3) Reliability :- Is accessed by measuring following parameters :
- Frequency and severity of failure
- Accuracy of o/p result
- Mean-time-to-failure (MTTF)
- Recovery from Failure
- Predictability of the program.

4) Performance :- Is evaluated by considering following characteries.
- Speed
- Responce time
- Resource consumption.
- Throughput & Efficiency.

5) Supportability :- collectively combines following 3 imp attributes
- Extensibility
- Adaptability
- Serviceability.

Design Process

- S/w design is done with the help of set of primitive components.
- Includes :
  - Algorithms
  - Diagrams
  - formulae design
  - low level component.
- It is one of the imp phase in s/w dev phase
- Good design generally results in successfull s/w implementation
- It contains semi standard methods like UML
- In s/w design process, the design patterns and architectural style are also used.

Design Quality :

Design should be

Modular in nature i.e. it consist of small partitions or sub system.

Represent data, architecture, interface & components in distiny way

Contain appropriate DS and recognisable Data pattern.

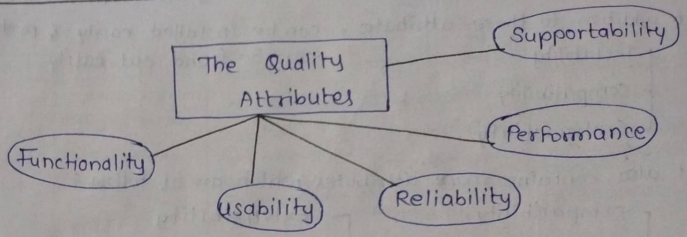
Represent independent functional characteristics.

Derived by using a reputable method.

use a notat^n that must lead to effective communication and understandings.

The Quality Attributes

FURPS



1) Functionality :-
   - Is an imp aspect of any s/w system
   - It is generally evaluated by the feature set and capabilities of that s/w.

2) Usability :- Is best evaluated by considering Following Factor
   - Human factor
   - overall aesthetics
   - Consistency & documentat^n

3) Reliability :- Is accessed by measuring following parameters :
   - Frequency and severity of failure
   - Accuracy of o/p result
   - Mean-time-to-failure (MTTF)
   - Recovery From Failure
   - Predictability of the program

4) Performance :- is evaluated by considering following characteries
   - Speed
   - Responce time
   - Resource consumption.
   - Throughput & Efficiency.

5) Supportability :- collectively combines Following 3 imp attributes
   - Extensibility
   - Adaptability
   - Serviceability.

Design Process

- s/w design is done with the help of set of primitive components.
  - Includes :
    - Algorithms
    - Diagrams
    - formulae design
    - low level component.
- It is one of the imp phase in s/w dev phase
- Good design generally results in successfull s/w implementation
- It contains semi standard methods like UML
- In s/w design process, the design patterns and architectural style are also used.

Design Quality :

Design should be

Modular in nature i.e. it consist of small partitions or sub system.

Represent data, architecture, interface & components in distinct way

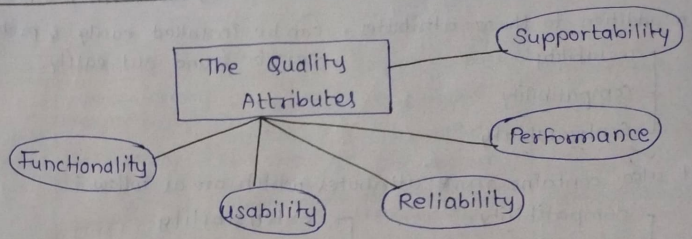
Contain appropriate DS and recognisable Data pattern.

Represent independent functional characteristics.

Derived by using a reputable method.

use a notatn that must lead to effective communication and understandings.

The Quality Attributes: FURPS



1) Functionality :-
   - Is an imp aspect of any s/w system
   - It is generally evaluated by the feature set and capabilities of that s/w.

2) Usability :- Is best evaluated by considering Following Factor
   - Human factor
   - overall aesthetics
   - consistency & documentatn

3) Reliability :- Is accessed by measuring following parameters:
   - Frequency and severity of failure
   - Accuracy of o/p result
   - Mean-time-to-failure (MTTF)
   - Recovery From Failure
   - Predictability of the program.

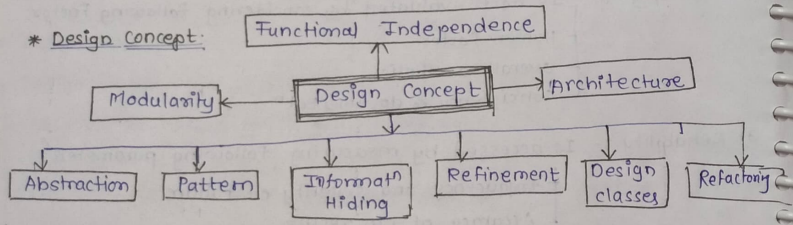
4) Performance :- is evaluated by considering following characteries.
   - Speed
   - Response time
   - Resource consumption.
   - Throughput & Efficiency.

5) Supportability :- collectively combines following 3 imp attributes
   - Extensibility
   - Adaptability
   - Serviceability.

- Points in Supportabity point are defined by the term maintainability.
- In addition to these attributes, can be installed easily & problem can be found out easily.
  - Testability
  - Compatibility
  - Configurability

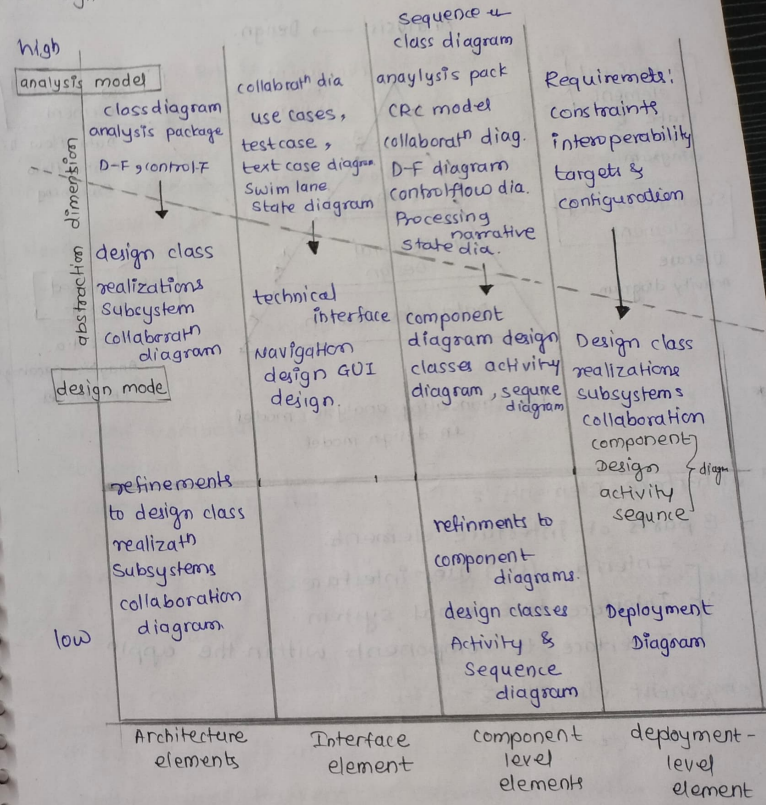
- It also contains more attributes which are as follow :
  - compatibility
  - Fault tolerance
  - Reusability
  - Security
  - Scalability
  - Extensibility
  - Modularity
  - Robustness
  - Portability.

* Design Concept:



1) Abstraction :- data, procedure, control
2) Architecture :- the overall structure of the software
3) Patterns :- "conveys the essence" of a proven design solution.
4) Modularity :- compartmentalization of data & function.
5) Information hiding : Controlled interface.
6) Functional Independence : high cohesion and low coupling
7) Refinement : Elaboration of detail for all abstraction.
8) Refactoring : Improve design without effecting behaviour.
9) Design classes :- create a new set of design
   define the Analysis.

The Design Model.



Process Dimension

* Data Elements :-
- Architectural level → DB & Files
- Component level → DS

* Architectural Element :-
1) Applicat^n Domain
2) Analysis model
3) Available styles & patter

* DESIGN ENGENEERIL

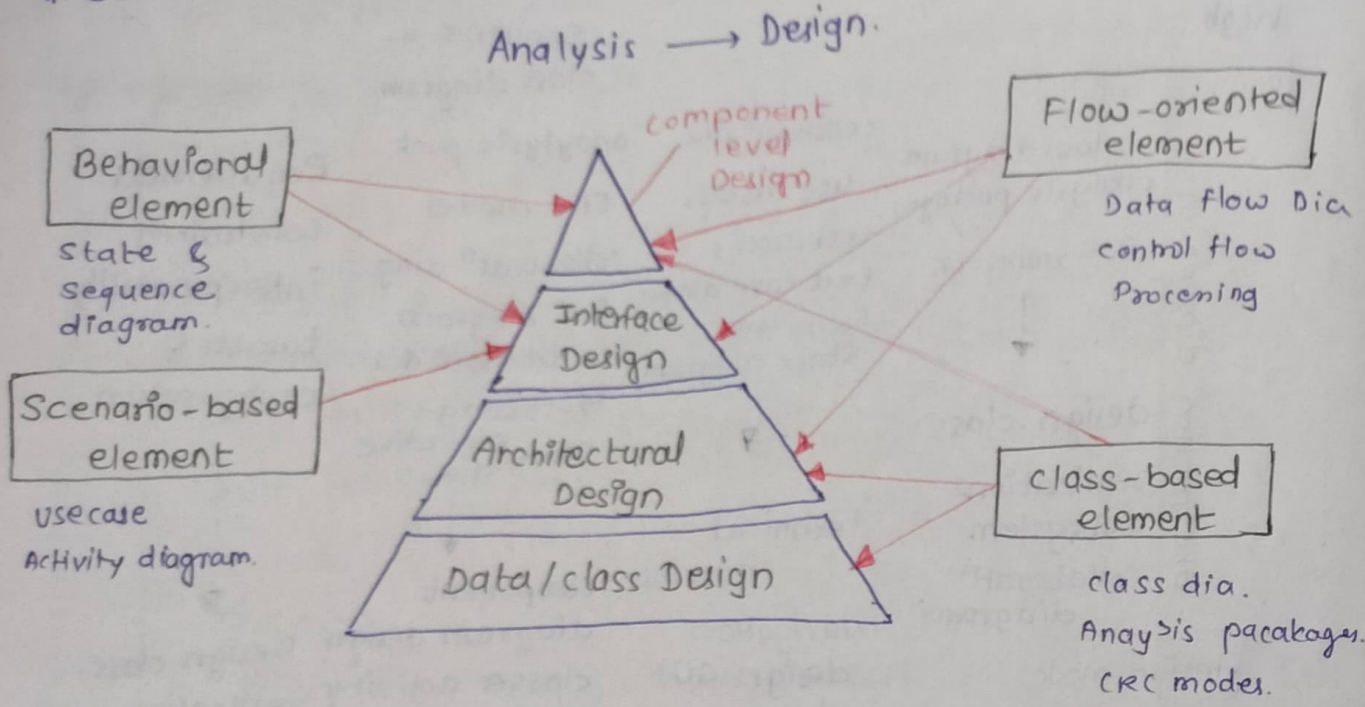


Fig: translating analysis model in design model.

* Interface elements :
  - 3 parts of interface elements.
    - Interface (UI) user interface
    - Interface to external system
    - Interface to components within the applⁿ

* component element :-
  -

* Deployment element :-
  - How s/w functionality & subsystem will be allocated

* Design Pattern :-

- Elements of Reusable Object - oriented s/w
- Their work largely based on Alexander's The timeless Way of building
- Focuses on some special element of design
- Focuses on component to component communication

* Iterator :-

-

* Idioms :-

- Usually implement the algorithms element of a component
- Also implements some mechanism for communication between components.

* Architecture Design

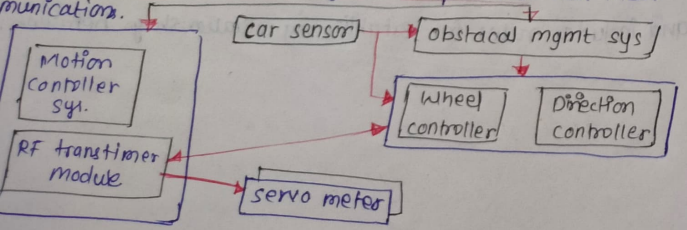
AD is concerned with understanding how a system should be organised and designing the overall structure of the system.

An early stage of the system design process

epresents the link between specification & design processes

n practice there is overlap between the processes of quirements and architectural design & is carried out in rallel with some specification activities.

Involves identifying major system components and their munications.



* Architecture Design & component level design

- The design process for identifying the sub-system making up a system and the framework for sub-system control and communication is Architecture design.
- The o/p of this design process is a description of s/w architecture

* Architecture design decision

- Architecture design is a creative process, where we can design a system organization that will satisfy the functional & non-functional requirement
- So the process differs depending on the type of system being developed.
- Think of architectural design as a series of decision to made rather than a sequnce of activities.
- However, a number of common decision span all design processes and these decisions affect the non-functional charateristics of the system.

Based on their experience & knowledge, system architects have to consider these fundamental questions:

- Is there a generic application architecture that can be used ?
- How will the system be ~~decomposed~~ distributed ?
- What architectural sytes are appropriate?
- What approch will be used to structure the system?
- How will the system be decomposed into modules?
- What control strategy should be used ?
- How will the architectural design be evaluated ?
- How should the architecture be documented?

* Architecture & system characteristics

- For distributed system across many different computer, the choice of distribution architecture is a key decision that affects.

1) Performance - localized critical op. within small no. of components
2) Security - use layered Architecture
3) Safety - in small no. of sub-system
4) Availability - Redundant → replace & update components without ⊗ system
5) Maintainability - Use fine-grain, replaceable components.

* Architecture View :-

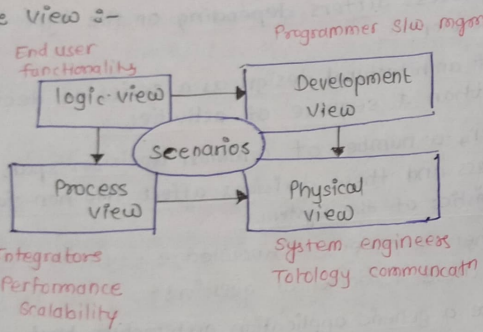
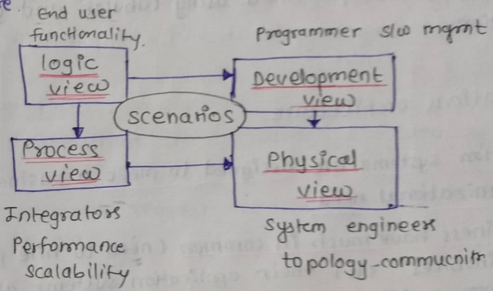


Fig: View Architecture.

- Architecture models may be used to document a design so it can be used as a basis for more detailed design and implementation
- 2 issues may araised :-
  - What views or prespectives are useful when desiging and documenting a system's architecture?
  - What notation should be used for describing A. models.

Each Architectural model only shows one view or prespective of the system.

- Might show how a sys. is decomposed into modules,
  - How the runtime processor interact or different ways in sys which sys components are distributed across a n/w
- For both design & documentation
- We usually need to present multiple views of the s/w architecture.



- There are 4 fundamental architectural views, which are related using use cases or scenarios :-

→ logic View :- Shows the key abstraction in the system as object or object classes.

→ Process view :- shows how, at run-time, the sys. is comp of interacting processes. (making judgement Non-function req-a

→ A development view :- shows how the s/w is decomposed development, that is the breakdown of s/w into components.

→ A physical view :- shows the system hardware & how s/w components are distributed across the Processors in the system.

* Architectural Pattern :-

- Pattern are means of representing, sharing & reusing knowledge.
- An architectural pattern is a stylized description of good design practice, which has been tried and tested in different environments.
- Patterns should include information about when they are and when they are not useful
- Patterns may be represented using tabular and graphical descriptions.

* Applications architecture.

- Application systems are designed to meet a business or an organizational need.
- As bussiness have much in common (need to hire people, issue invoices etc) their application systems also tend to have a common architecture that reflects the application requirements.
- A generic application architecture is an architecture for type of s/w system that may be configured and adapted to create a system that meets specific requirements
- Application reuse is possible without re-implementation
- A system for supply chain managment can be adapted for any type of supplier, goods.

Use of application Architecture

- As a starting point for architectural design. If you are not familiar with the type of application you are developing.
- As a design checklist. If you have developed an architectural design, you can compare this with the generic design.
- As a way of organizing the work of the development team.
- As a means of assessing components for reuse, by comparing with the generic structure.
- As a vocabulary for talking about application types, using the concepts identified in the generic architecture.

* Transaction Processing System.

- Developed to facilitate the transaction executed by the customer.
- It process customer request for info from a DB or request to update a DB.

* Language Processing System.

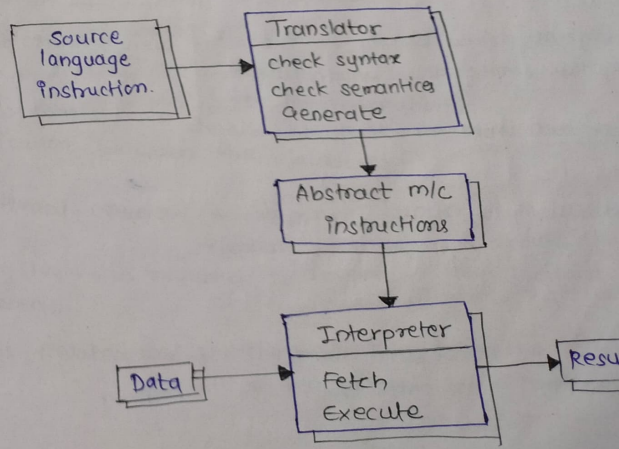
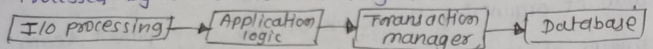


fig : Architecture of LPS

* Transaction Processing System.

- Process user request for information from a DB or request to update the DB.
- User make asynchronous requests for service which are then Processed by a transaction manager.

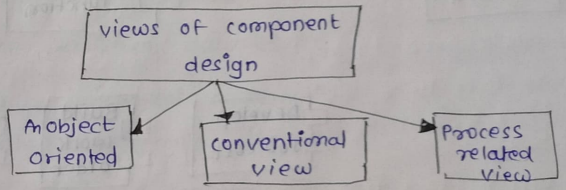


* Languge processing system

- To translate a natural language into artificial / other language.
- In S/w engineering compiler translate the artifical programming code into a machince code.

* Component level Design:

- A component is a basic building block for the computer s/w
- It is a higher level abstraction defined by their interface
- It helps in achieving the objectives & requirements of Sys' to be built
- An individual s/w component is a s/w package, a web server or a module that encapsulates a set of related functions.
- CLD comes after AD is completed.
- Possible to represent the CLD by using some programming languages these programs can be created by using th ADM.
- CLD is a Alternate approch to ADA.
-



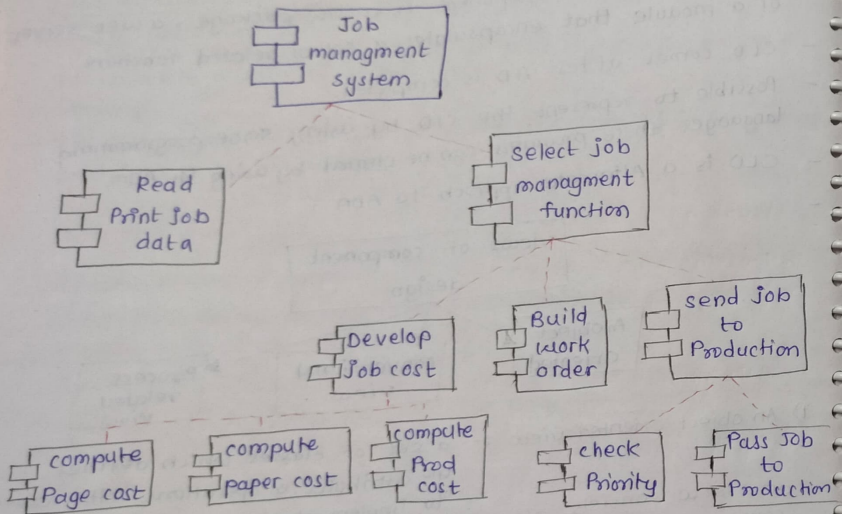
1) An object oriented view :- a set of classes which defines the attributes & operations with respect to implementation
- OOV s/w consists of details of all the messages used for communication between the classes.

2) Conventional view :- A component consist of a functional element program or a module that has Processing logic, DS required to implement the inertace between the components.

3) Process - Related View :- The main focus is on the reusability of the existing s/w components.

* Designing class-based components.

- As we seen o/o s/w engineering approch focuses on component level design and analysis classes and interfaces between the classes.
- The design details are discussed in following.



Basic Design Pattern

1) The Open-closed Principle (OCP)
2) The liskov substitution Principle (LSP)
3) Dependency Inversion Principle (DIP)
4) The Interface Segregation Principle (ISP)
5) The Release Reused Equivalency Principle (REP)
6) The common closure Principle (CCP)
7) The common reused Principle (CRP)

*Component level Design steps

- 1): Identify those design classes that are related to problem domain only
- 2): Identify classes that are related to infrastructure domain. The classes and components in this step include GUI components Os components & object & data mgmt components.
- 3]: Identify all design classes that donot have reusable components & elaborate these design classes.
  - The message details are also specified when the components

step 1 : Identify design classes in problem domain
step 2 : Identify infrastructure design classes
step 3 : Elaborate design classes
step 4 : Describe persistent data source.
step 5 : Elaborate behavioral represent^n
step 6 : Elaborate development diagram.
step 7 : Refactor design & consider alternatives.

* User Interface Design

- User interface is front-end applicatn view to which user interacts in order to use the s/w
- The s/w become more popular if its user interface is :-
  1) Attractive
  2) Simple to use
  3) Responsive in short time
  4) clear to understand
  5) Consistent on all interface screens.

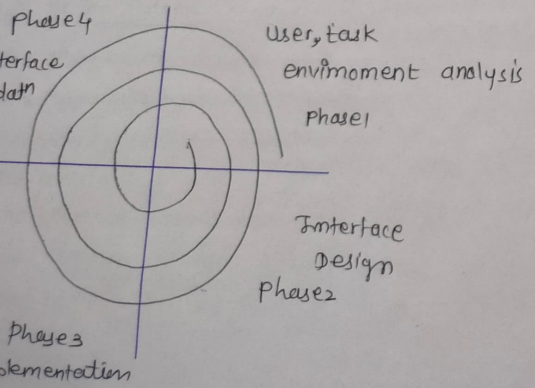
types

command line

- provieds command prompt
- where user types the command & feeds to the system.
- The user needs to remember the syntax of command & its use

Graphical User Interface

- Provide simple interactive interface to interact with the system.
- GUI can be combination of both h/w & s/w
- Using GUI, user interprets that s/w.



* Characteristics of Good UI

- Clarity
- Concision
- Efficiency
- Forgiveness
- Attractive
- Consistency
- Responsiveness

* Enlist the golden rules for User Interface Design

1) Place the user in control.
2) Reduce the user's memory load.
3) Make the interface consistent.

1) Place the user in control :-

- To simplify the mode of interaction
- To simplify implementatn of interface

- Define the interaction modes insuch a way that is easy the user not forced to perform unnecessary or undesired actions.
- Provide for flexible interaction: Diff. people will use different interaction mechanisms, Some might use keyboard commands, some might use mouse, some might use touch screen, etc. all interaction mechanism should be provided.
- Allow user interaction to be interruptable & undoable : user must be able to do undo operations.
- streamline interaction as skill level advances & allow the interaction to be customized : user should get chance to customize the interface as user wants which allows diff. interaction mechanism so that user doesn't feel bored
- Hide technical internals from casual users : user should not aware about interna technical detail of sy
- Design for direct interaction with objects that appear onscreen

+ Reduce the user's memory load :

- Reduce demand on short-term memory : When user are involved in some complex task the demand on short-term memory is significant.
- Establish meaningful defaults : Always initial set of defaults should be designed provided to the average user.
- Define shorcuts that are intuitive : Mnemonics should be used by the user.
- The visual layout of the interface should be based on real-world metaphor : - Anything you represent on a screen if it is metaphor for real-world entity then user should easily understand.
- Disclose information in a progressive fashion ; The interface should be behavior should be presented first at a high level abstraction

+ Make the interface consistent

- Allow the user to put the current task into a meaningful context : Many interfaces have dozens of screens so it is imp. to provide indicators consistently.

Maintain consistency across a family of application. : The development of some set of applications all should follow & implement the same design, rules so that consistency is maintained among applications

If past interactive models have created user expectation do not make changes unless there is a compelling reason.

Interface Design steps & Analysis.

- Interface design is an iterative process like all other sw engineering design process.
- IA — all tasks required by end-user - have to be identified in detail & interface activity starts.
- steps :-

1) Using info developed during interface analysis ; define interface objects & actions.
2) Define events that will cause the state of the UI to change -
3) Depict each interface state, as it will actually look to the end user
4) Indicate how the user interprets the state of the System from informatem provided through the interface.

UNIT VI

Recent Trends in SE

1) SCM, Risk mgmt, Technology evaluatn, process trends, collaborative develop, s/w reuse, test-driven develop

* **SCM** - System Configuration Managment
- It is the task of tracking and controlling changes in the s/w
- The process of identifying and defining the s/w configuration items in a system
- controlling the release and change of these items throughout the system lifecycle,
- Recording & Reporting the status of configuration items and change requests & verifying the completenen.

* SCM Basics

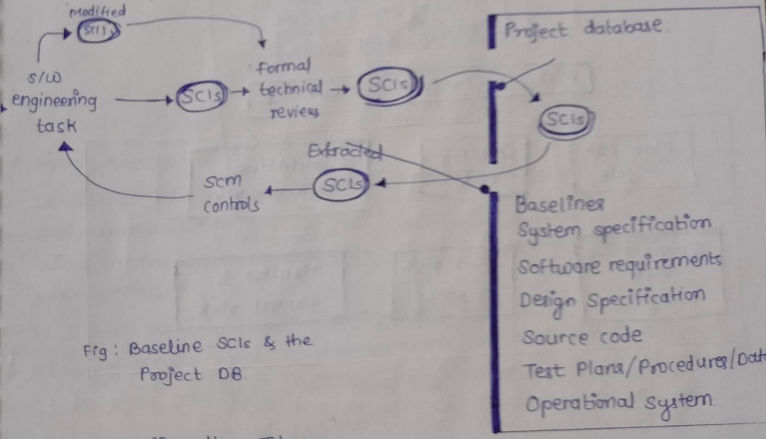
4 basic elements that should exist when a configuration managment system is developed

1) Component elements : the tools in the file managment system uses the s/w configuration item.

2) Process elements : The process elements or the procedures uses the s/w effective approch towards the change mgmt in engineering and use of computer s/w

3) Construction element :- The automated tools are used in construction or the development process and ensuring the validated components should be assembled

* Baselines
- The change is the only constant in s/w development life cycle
- The customer want to modify the requirements as the model gets ready
- since in the beginning, even customer is not fully aware of the Product requirement
- As the development begins, castomers needs lot of changes in the requirement



Frg : Baseline SCIs & the Project DB

SCI S/W conffguration Items

- Basically SCI ie s/w configuration item is the internal integral part of s/w engineering development process.
- It is a part of large specification or we can say that one test case among large specification or we can say that one test case among large suite of test cases.
- In fact the SCI is a document or the program like C++ functions or a Java applet.

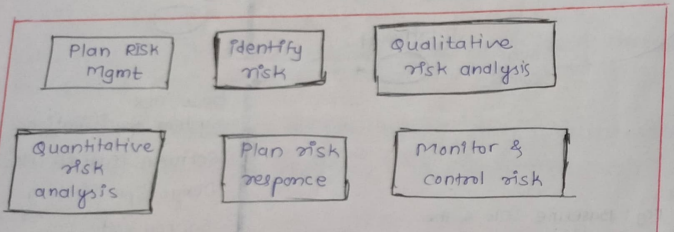
Features :- 1) Versioning
2) Dependency tracking & change mgmt
3) Requirement tracking
4) Configuration mgmt
5) Audit trails.

* Risk managment :-

- Risk is an uncertain even that may have +ve or -ve impact on Project
- Risk mgmt is the Process of identifying & migrating risk

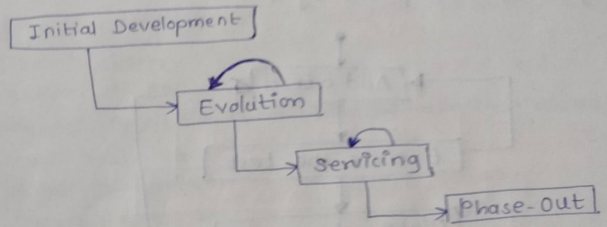
Why is it imp ?- Risk affects of our project
- Affect on budgets
- Affect on schedule
- Affect your scope

Risk in Project Managment



Technology evolution :-

- The process of developing a s/w product using s/w engineering Principles and methods is refferred to as s/w evolution



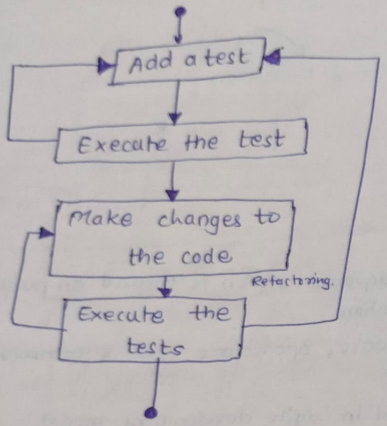
* Process trends.

- It is s/w development which is focused on public availability
- and communication
- used in freeware, opensource. s/w & common based peer product
- It is also used in agile development model

* Test driven Development

- It is slw development process which relies on repetition of very short development cycle

-



Test-Driven Development is a process of developing and running automted test before actual developing of the application

Global slw challenges :-

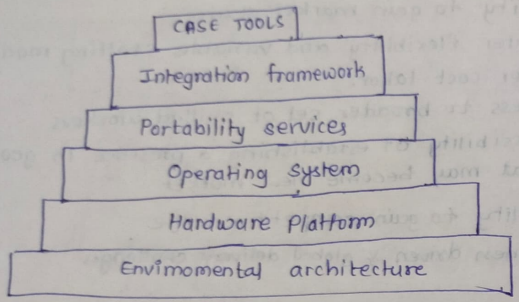
- Communicatn breakdown
- Co-ordinatn breakdown
- Control breakdown
- Cohesion barries
- Cluture clash
- Ability to gain market shares
- Greater flexibility and variable staffing model
- lower cost labor.
- Access to broader set of skilled workers
- Possibility of establishing a presence in geographics that may become new market
- Ability to gain competitive edge
- Busines driven & global delivery challenges

* Collaborative Development

- It is a online meeting space where a slw development Project stakeholders can work together.
- No matter what timezone or region they are in to discuss, document & produce project deliverables
- Although growing from a tool base in the slw development sector,
- The CDE has been taken up in other sectors
- With typically geographical dispersed,
- where it is beneficial to be able to collaborate across the web, including automotive & aeronautical engineering movie production and civil engineering.

* CASE Computer-Aided S/W Engineering

- CASE tools Developed to help s/w eng.
- Managers and all the s/w practitioners in all the s/w activities related to s/w dev. phase.



Components of CASE Tools

CASE categories

1) Upper case
2) Lower case
3) Integrated case.

CASE tools.

CASE tools

- class of s/w which is useful to automate the different activities in life cycle

- Business process engineering tools
- Project planning tools
- Risk analysis tools
- Project mgmt tools
- Requirement tracking tools
- Sys. S/W. tools
- Quality assurance tools
- DB mgmt tools
- Prototyping tools
- Programming tools
- web development tools
- Static analysis tools
- Dynamic analysis tools
- Test mgmt tools.

Functions of CASE tools

1) Analysis
2) Design
3) Code generation
4) Documentation

* Agile tools

1) JIRA
2) kanban

1) JIRA

- Is a agile project mgmt tool
- supports agile methodology
- be it scrum
- JIRA s/w is part of family of products designed to help teams of all types manage work.
- JIRA is top recommendation for agile s/w development teams.
- JIRA is a project mgmt tool used for issues & bugs tracking system.
- It is widely used as an issue-traking tool for all types of testing.

* COCOMO Model cocomo model

- cocomo stands for constructive cost model
- It is one of the very famous model which is used to estimate the cost of the project.
- Barry Boehm has devised s/w estimation models
- cocomo model has been evolved into more comprehensive model called COCOMOII
- Application composition model. - used early stage of development.
- Early design stage model : Once the requirements are stabilized & basic architecture constructed, then early
- Post architecture model : used during development of s/w
- cocomo II models also require sizing info like other estimation model for the s/w.
  - > object points
  - > Function points
  - > Line of source code.

object points — indirect s/w measure
Computed using the counts of
- Screenshots taken at user interface