UNIT - II

QUE : WHAT IS SOFTWARE REQUIREMENT SPECIFICATION OR WHICH DETAILS CONTAINS SRS

Software requirement specification is the repertory (or data dictionary) of system.

OR

Software requirement specification is the written documentation which contains details description of developed software.
(for example : srs contains details about table, input, output and process of software)

QUE : GIVE THE USE OR ADVANTAGES OR NEED OF SRS DOCUMENT

Different people need the SRS document for very different purpose. Some of the important categories of users of the SRS document and their needs are as follow:

Users, customers, and marketing personnel:
User get the way for operating the new software.
the customer get some initial information about new software.
the marketing personal need to understand the requirement that they can explain to the customers.

Software developers:
-The software developers refer to the SRS document to make sure that they are developing exactly what is required by the customers.
-The software developers refer to the SRS document and take some guide line for developing new software.

The engineers:
Their goal is to ensure that the requirement are understandable from a functionality point of view, so that they can test software and validate its working. They need that the functionality be clearly described , and the input and output data identified precisely.

User documentation writers:
It provide help to write the users manuals about new developed software.

Project managers:
They want to ensure that they can estimate the cost of the project easily by referring to the SRS document and that it contains all the information required to plan the project.

Maintenance engineers:
The SRS document helps the maintenance engineers to understand the functionalities of the system.
A clear knowledge of the functionalities can help them to understand the design and code.
Also, a proper understanding of the customer's requirements would enable them to determine what modifications to the system's functionalities would be needed for a specific purpose.
Many software engineers in a project consider SRS document as a reference document.

it is often more appropriate to think of the SRS document as the documentation of a contract between the development team and the customer.
the SRS document can be used to resolve any disagreements between the developers and the customers that may arise in the future.
The SRS document can even be used as a legal document to settle disputes between the customers and the developers.

I.m.p.

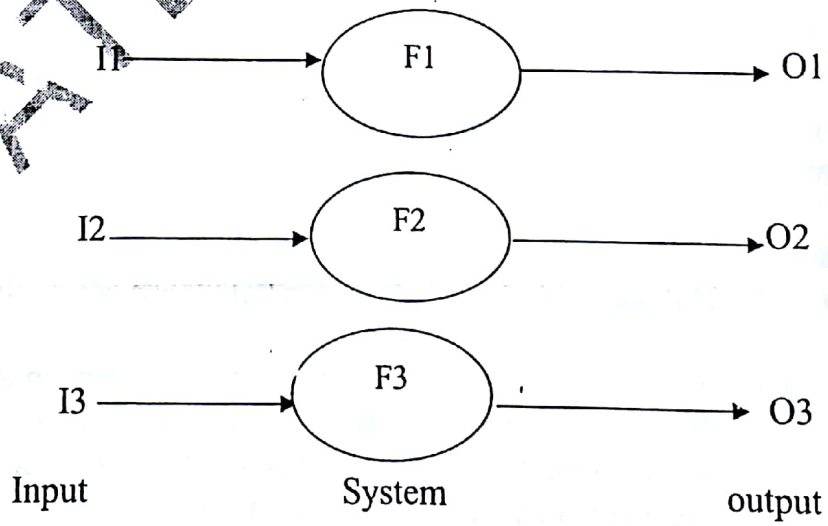
QUE : CHARACTERISTICS OF A GOOD SRS DOCUMENT:

The skill of writing a good SRS document usually comes from the experience gained from writing SRS documents for many problems.
some of the identified desirable qualities of SRS document are the following:

Concise:
-The SRS documents should be concise and at the same time unambiguous, consistent, and complete.
-irrelevant descriptions reduce readability and also increase error possibilities.

Structured:
It should be well-structured. A well-structured document is easy to understand and modify.

Black box view:
This means that the SRS document should specify the external behavior of the system and not discuss the implementation issues.
This vise with which a requirement specification is written, has been shown in figure.
observe that in figure the SRS Document describes the output produced for the different type of input And a description of the processing required to produce for the output from the input (show in ellipses) and the internals of the system (show as a rectangle) are not discussed at all.
The SRS document should vise the to developed as a black box, and should specify the externally visible behavior of the system. For this reason, the SRS document is also called the back box specification of a system.
Figure



Traceable :
It should be possible to trace a specific requirement to the design element that implement it and vice versa. It should be possible to trace this requirement and vise versa.

Response to undesired events:
It should characterize acceptable response to undesired events. These are called system response to exceptional conditions.

Verifiable:
-All requirement of the system as documented in the SRS document should be verifiable.
-This means that it should be possible to determine whether or not requirement have been met in an implementation.
Requirement such as the system should be user-friendly is not verifiable.
-On the other hand, when the name of a book is entered, the software should display the location of the book, if the book is available is verifiable. Any feature of the required system that is not verifiable should be listed separately in the goals of the implementation section of the SRS document.

QUE : GIVE THE EXAMPLE OF BAD SRS DOCUMENTATIONS:
Some of the important categories of problems that many SRS documents suffer from are as follows:

1) Over–specification:
Over-specification occurs when you try to address the to aspects in the SRS document.
Over-specification restricts the freedom of the designer in arriving at a good design solution.

2) Forward references:
You should not refer to aspects that are discussed much later in the SRS document.
Forward referencing seriously reduce readability of the specification.

3) Wishful thinking:
This type of problems concern description of aspects which would be difficult to implement.

4) Noise:
The term noise refers to presence of material not directly relevant to the software development.

5) Making bad assumption.

6) Writing implementation instead of requirement.

7) Assign incorrect terms.

8) Assign wrong language.
9) Missing requirement.

QUE : EXPLAIN TYPES OF REQUIREMENT OR CATEGORIES OF REQUIREMENT.

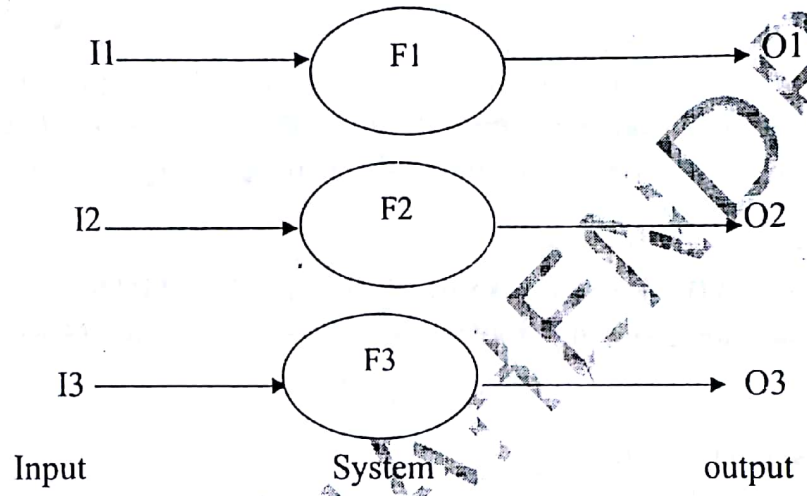
There are three types of requirements are as :

1) **functional requirements.**
2) **The non -functional requirements.**
3) **Goals of implementation.**

The three most important contents of this document are the

1) functional requirements : this document contains details about the input data, the processing required on the input data, and the output data to produced by software.

We give the functional view as following figure :



We express functional requirements in mathematical notation as :

$$f : I \longrightarrow O$$

Here, I as input domain, which contains input element (i_i),
O as output domain, which contains output value (o_i),
Here function f transforms an element (i_i) in the input domain (I) to a value (o_i) in the output domain (O).

2) The non-functional requirements : identify the performance requirements, the required standards to be followed, etc.

-The SRS document is written using end-user terminology. This makes the SRS documents understandable by the customer.

-Non-functional requirements include reliability issues, accuracy of results, human- computer interface issues and constraints (rules) on the system implementation.

- Non-functional requirements also include constraints on the system implementation such as the specific database management system (DBMS) to be used due to customer request.

-IEEE 870 standard, there are four types of non- functional requirement.
1) external interface requirement (hardware, software and end-user).
2) performance requirements (timing).
3) Constraints.
4) software system attributes.

3) Goals of implementation :
The goals of implementations part of the SRS document offers some general suggestion regarding development.
- suggestion which is used in future requirement and to solve future problems.

QUE : EXPLAIN ROLE OF MANAGEMENT IN SOFTWARE DEVELOPMENT.
Following responsibility of management in software development.

1) Proposal writing.
2) Project planning & scheduling.
3) Project costing.
4) Project monitoring and reviews.
5) Personal selection and valuation.
6) Report writing and presentation.

1) Proposal writing.
Personal describes the objectives of the project and how it will be carried out.
Specify the requirement of software and to collect the required data.
It usually includes cost and scheduling estimates.
It is skill which is acquired by experience.

2) Project planning & scheduling.
Project planning is concerned with identify the activities miles hones and deliverables product by project.
Management is responsible for make the planning for finishing software in time.
He/she make planning for analysis, design and developing of new software.

3) Project costing.
Management person apply economical feasibility analysis and calculate the estimate cost of software.

4) Project monitoring and reviews.
-Project monitoring is continuing project activity. The manager must keep track of the progress of the project and compare actual and planned progress and cost.
-A skilled manager daily discussion with project might reveal a particular problem in finding some software fault.

- The project management reviews concern with reviewing the over all progress and technical development of the project.

5) personal selection

The project manager usually responsible to select skill staff with appropriate experience with budget will be available to work on the project.

6) Report writing and presentation.

The project manager usually responsible for reporting the project to both the client and organization.

The project manager also responsible for creating SRS document of software.

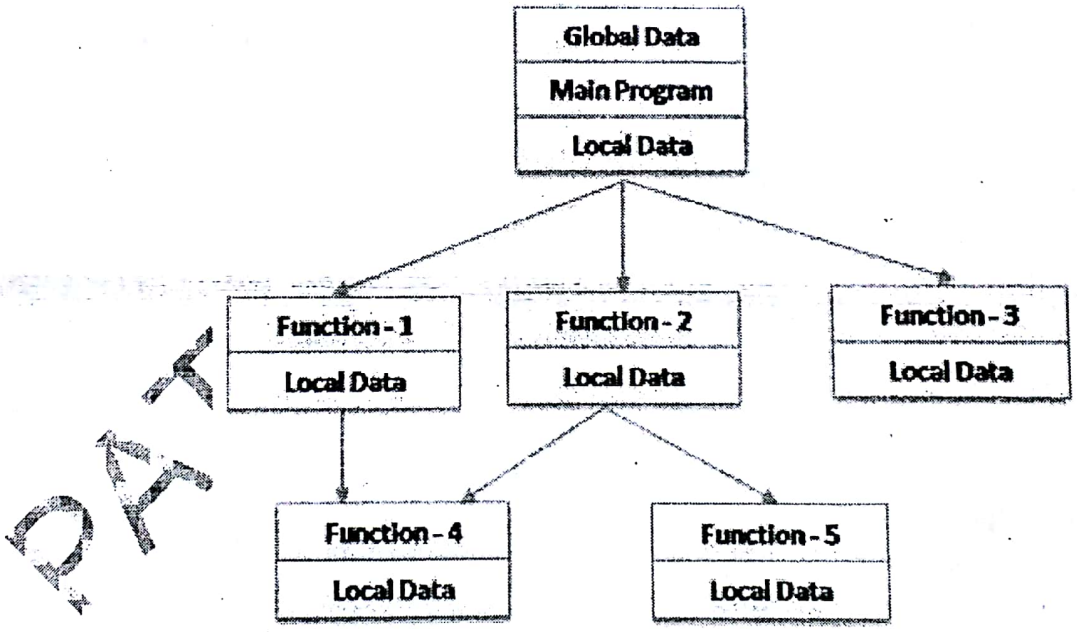
Difference between function oriented and object oriented approach.

QUE : FUNCTION ORIENTED VERSES OBJECT ORIENTED.

Function oriented :

Some characteristics of procedure oriented design

1) large programs are divided into smaller programs knows as functions or modules.
2) most of the functions (or module) share global data.
3) each function (module) may have its own local data.
4) data move openly around the system from function to function.
5) follows top – down approach in program design.

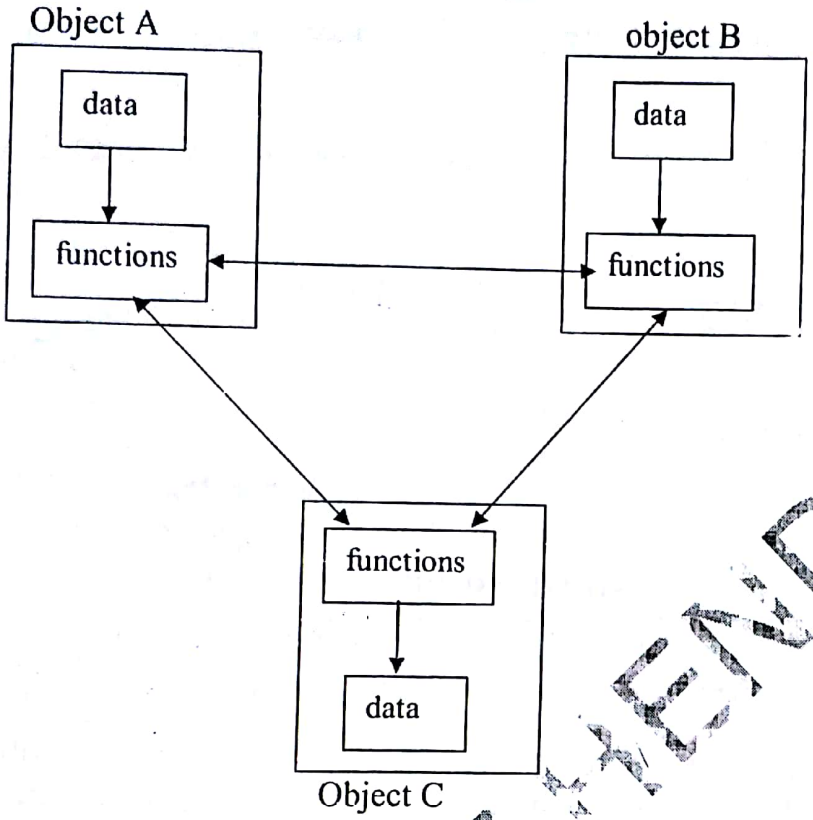


OBJECT ORIENTED.

OOP allows de composition of a problem into a number of entities called object and then build data and function around these objects.

The organization of data and function in object oriented program in shown in fig.

The data of an object can be accessed only by the function associated with that object. However, function of one object can access the functions of other objects.



SOME OF THE STRIKING FEATURES OF OBJECT – ORIENTED DESIGN

1) Emphasis is on data rather than procedure.
2) programs are divided into what are known as objects.
3) data structures are designed such that they characterize the objects.
4) function that operate on the data of an object.
5) data is hidden and can not be accessed by external functions.
6) objects may communication with each other through function.
7) new data and function can easily added whenever necessary.
8) follows bottom – up approach in program design.

METRICS : the purpose of metrics at any point during a development project is to provide quantitative information to the management process so that the information can be used to effectively control the development process.
The metrics define the requirements are.

Quality Metrics :

1. no. of errors.
2. change request frequency.
3. SRS quality attributes.

MEASUREMENT : provide the actual value for the metrics

Size majors

1. Text page majores
2. Function point.
3. Bang metrics.

QUE : ROLL OF METRICS AND MEASUREMENT

1. For the effective project monitoring the information coming from the development process to the management process. It's should be objective and quantitative data about the project.
2. if the information of the is not quantities than subjective judgment will have to be use.
3. the needs of the quantitative data from the process require that software metrics can be use.
4. the software metrics are quantifiable majors on the measure that use to measure different at characteristics of a software system or software development process. all engineering discipline has metrics to quantify various characteristics of their products.
5. Software metrics is an engineering are, bow the software has no physical attributes, and conversion metrics are not help in designing metrics for software.

Metrics.
Provide a quantifiable of some property.

Measurements
Provide the actual value for the metrics.

Designing
Designing is the first step in moving from the problem domain towards the solution domain. it is essential bridge between requirement specification and the final solution for satisfying the requirement.
It is the way that you can accurately translate stakeholder requirement in to finish software product or system.
Designing is use in two way.

1) Verb.
2) Noun.

Verb.
It represent the process of design.

Noun.
It represent the result of the design process which is the design for the system,

Design of the system is essentially a blue print or plan for solution for the system.
The goal of the design process is to produce a model or representation of a system which can be use later to build the system the produce model call design of the system.

The design process for the system has two levels.

Top level design or system design.
If focus is on designing which modules are needed for the system and specification of this module. how the module should be inter connected call the top level designing or system designing.

Detail design or logic design.
The internal design of the module or how the specification of the modules can be satisfied is decided call detail design or logic design.
Detail design is the extension of system design.

Principle of design. (Strategy of design)

1.) design of system is corrected.
2.) Design should clearly verifiable complete and traceable.
3.) Simplicity means creating simple design.
4.) Design should cohabit uniformity and integration.
5.) Design should be structure to accommodate change.

Problem portioning

When we solve the small problem the entire problem can be tackle at one but for solving the larger problem the basic principle is the time tasted the "divide and concur".
It means the problem can be divided into small peaces(modules) and each peace can be word separately.
The main goal of the partition is the divided problem is manageable small peace and each peace can be small separately.
This is the belies that the cost of solving the entire problem is more than the sum of the cost of solving the all that peace.
The partition is also use for simplicity and understandability of design.
The maintenance is minimize if each part in the system can easily related to the application and each peace can be modify separately.

Problem partition is also aims for design verification.
It is essential for solving the complex problem, it ideas to hierarchy design.
It can be represented as hierarchy of component.

What is design pattern.

Pattern is a name of inside which conversion the sense of prove solution to receiving problem within certain contact.
It desirable design structure that solve the particular design problem within specified context it forces that may have import of the manner in which the pattern is apply for use.
It provide the description that enable to designer that whether the pattern is applicable or not for the current work.
Whether the pattern can be reuse or not (save the designing time).
Pattern can be serving a guide for developing.
A designing pattern can be characterized as "a three part rule which express a relation between a certain context a problem and a solution .
For software design context allows the reader to understand the environment in which the problem resides and what solution might be appropriate within that environment.
A set of requirements including limitations and constrains acts as a system of forces that influences how the problem can be interpreted within its context and how the solution can be effectively applied.

Characterized an effective design pattern in the following

It solve a problem : patterns capture solutions, not just abstract principles or strategies.
It is a proven concept : pattern capture solution with a track record, not theories or speculation.
The solution isn't : many problem solving techniques (such as software design paradigms or methods) try to derive solutions from the first principles. the best patterns generate a solution to problems indirectly a necessary approach for the most difficult problem of design.
It describe a relationship: pattern don't just describe modules, but describe deeper system structures and mechanism.
The pattern has a significant human component(minimize human interventions) : all software serves human comfort or qualify of life: the best patterns explicitly appeal to aesthetics and utility.

Architectural patterns:
Architectural patterns describe broad based design problems that are solved using a structural approach .

Data patterns
Data patterns describe recurring data oriented problems and the data modeling solutions that can be used to solve them.

Component patterns

Components patterns also referred to as design patterns address problem associated with the development of subsystem and of a components, the manner in which they communicate with one another and their placements with in a larger architecture.

Interface design patterns

Interface design patterns describe common users interface problems and their solution with in a system of forces that includes the specific characteristics of and users.

Web Application Patterns:

Web Application patterns address a problem set that is encountered when building web applications and often incorporates many of the other patterns categories just mentioned.

Creational Patterns:

Creational patterns focus on the "Creation, composition and representation" of objects.

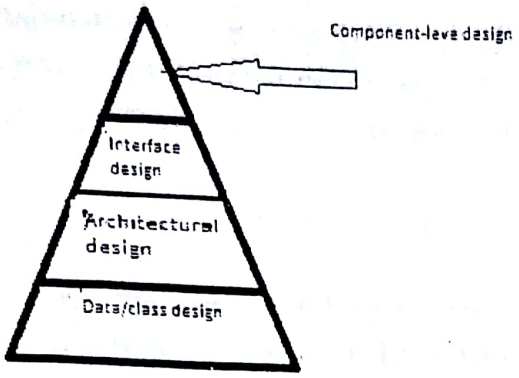
Structural Patterns:

Structures patterns focus on problems and solutions associated with how classes And objects are organized and integrated to build a larger structure.

Behavioral Patterns:

Behavioral patterns address problems associated with the assignment of responsibility Between object and the manner in which communicate is effected between object.

Design Model:
Software requirements have been analyzed and modeled; software design is the last software engineering action within the modeling activity and sets the stage for construction (code generation and testing).



Design Model

Date/Class design:
The data/class design transforms class models in to design class realization and the requisite data structures required to implement the software. The objects and relationship define in the CRC diagram and the detailed data content depicted by class attributes and other notation provide the bases for the data design action part of class design may occur in conjunction with the design of software architecture. More detailed class design occurs as each software component is designed.

Architectural design:
The architectural design defines the relationship between major structural elements of the software, the Architectural styles and design patterns that can be used to achieve the requirements defined for the system, and the constrains that affect the way in which Architecture can be implemented. Architectural design representation the frame work of a computer based system is derived from the requirements model.

Interface design:
The interface design describes how the software communicates with the system that interoperate with it, and with humans who use it, an interface implies of information (e.g. data and/or control) and a specific type of behavior.

Component level design:
The component design level design informs structural elements of the software Architecture in to a procedural description of software components. Information obtained from the class based models, and behavioral models serve as the basis for component design.

Note:
The importance of software design can be started with a signal word-quality only way that you can accurately translate stack holders requirement in to a finished software product or system.

Evaluation of software design

An Evaluation of software design is the continuous process then evaluates the design the number of common characteristics are.

1. Mechanism for the translation of the requirement model in to design representation
2. Notation for representing functional component and interface.
3. Heuristics for requirement and part string guideline for quality easement.

Abstraction:

When you consider a modular solution to any problem then many level of abstraction can be pass.

An abstraction of a component describe the external behavior of a that component without bothering with internal details

An abstraction definition of the component is similar when the component it self.

Abstraction is indispensable of the design process, it is essentially for problem partition

The components are not isolated from each other, they interact with each rather and designers how to specify how the components are interact with each other.

It is also use for existing components as well as the component that are being design.

The abstraction of existing component place the important role in a maintain on phase because, to modify the system the first step to understand. "what the system does and how".

There are two common mechanisms for abstraction for software system.

1) Function abstraction.
2) Data abstraction.

Function abstraction:

The modules specify by the function or it specify.

Example:

The module to component the log of value, it can be abstractly by function log.

To short input array can be represented by the function sorting function.

Data abstraction:

It is entity in the real world

It provides some services to the environment to which it belong.

The entities provide some fix predefines services.