INTRODUCTION TO SYSTEM DESIGN

Systems design is the process of defining elements of a system like modules, architecture, components and their interfaces and data for a system based on the specified requirements. It is the process of defining, developing and designing systems which satisfies the specific needs and requirements of a business or organization.

System design is the process of designing the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system.

System design is the phase that bridges the gap between problem domain and the existing system in a manageable way. This phase focuses on the solution domain, i.e., *"how to implement?"* It is the phase where the SRS document is converted into a format that can be implemented and decides how the system will operate.

Primary objective of systems design

The purpose of system design is to have a better understanding of the organizations needs and requirements and to create a documented set of specifications to enable the implementation be consistent with architectural entities as defined in models and views of the system architecture. System design is used to define, organize, and structure a solution based on those needs and requirements.

The goal of the design process is to produce a model or representation of a system, which can be used later to build that system. The produced model is called the design of the system. The design of a system is essentially a blueprint or a plan for a solution for the system.

Elements of a System

- *Architecture* - This is the conceptual model that defines the structure, behaviour and more views of a system. We can use flowcharts to represent and illustrate the architecture.
- *Modules* **-** This are components that handle one specific tasks in a system. A combination of the modules makes up the system.

- *Components* - This provides a particular function or group of related functions. They are made up of modules.
- *Interfaces* **-** This is the shared boundary across which the components of the system exchange information and relate.
- *Data* **-** This the management of the information and data flow.

Major Tasks Performed During the System Design Process

1. *Initialize design definition*

- Plan for and identify the technologies that will compose and implement the systems elements and their physical interfaces.
- Determine which technologies and system elements have a risk to become obsolete, or evolve during the operation stage of the system. Plan for their potential replacement.
- Document the design definition strategy, including the need for and requirements of any enabling systems, products, or services to perform the design.

2. *Establish design characteristics*

- Define the design characteristics relating to the architectural characteristics and check that they are implementable.
- Define the interfaces that were not defined by the System Architecture process or that need to be refined as the design details evolve.
- Define and document the design characteristics of each system elements.

3. *Assess alternatives for obtaining system elements*

- Assess the design options
- Select the most appropriate alternatives.
- If the decision is made to develop the system element, rest of the design definition process and the implementation process are used. If the decision is to buy or reuse a system element, the acquisition process may be used to obtain the system element.

4. *Manage the design*

- Capture and maintain the rationale for all selections among alternatives and decisions for the design, architecture characteristics.
- Assess and control the evolution of the design characteristics.

Types of System Design

Logical Design

Logical design pertains to an abstract representation of the data flow, inputs, and outputs of the system. It describes the inputs (sources), outputs (destinations), databases (data stores), procedures (data flows) all in a format that meets the user requirements.

While preparing the logical design of a system, the system analyst specifies the user needs at level of detail that virtually determines the information flow into and out of the system and the required data sources. Data flow diagram, E-R diagram modelling are used.

Physical Design

Physical design relates to the actual input and output processes of the system. It focuses on how data is entered into a system, verified, processed, and displayed as output.

It produces the working system by defining the design specification that specifies exactly what the candidate system does. It is concerned with user interface design, process design, and data design.

It consists of the following steps −

- Specifying the input/output media, designing the database, and specifying backup procedures.
- Planning system implementation.
- Devising a test and implementation plan, and specifying any new hardware and software.
- Updating costs, benefits, conversion dates, and system constraints.

Architectural Design

It is also known as high level design that focuses on the design of system architecture. It describes the structure and behaviour of the system. It defines the structure and relationship between various modules of system development process.

Detailed Design

It follows architectural design and focuses on development of each module.

Conceptual Data Modelling

It is representation of organizational data which includes all the major entities and relationship. System analysts develop a conceptual data model for the current system that supports the scope and requirement for the proposed system.

The main aim of conceptual data modelling is to capture as much meaning of data as possible. Most organization today use conceptual data modelling using E-R model which uses special notation to represent as much meaning about data as possible.

Inputs to System Design

System design takes the following inputs −

- Statement of work
- Requirement determination plan
- Current situation analysis
- Proposed system requirements including a conceptual data model, modified DFDs, and Metadata (data about data).

Outputs for System Design

System design gives the following outputs −

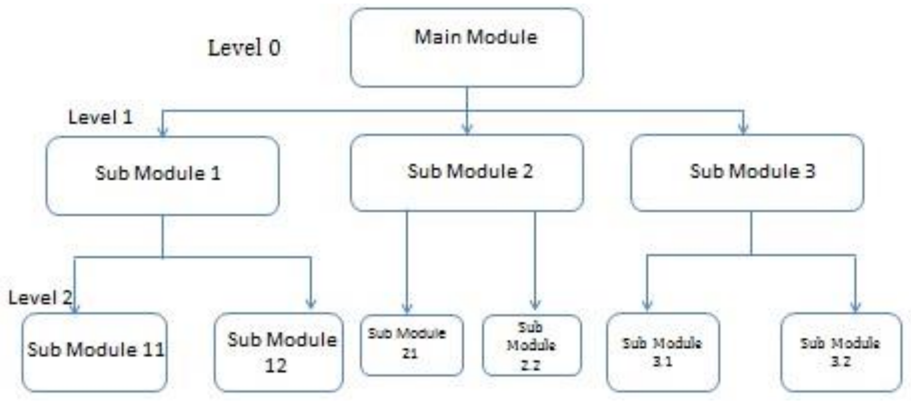
- Infrastructure and organizational changes for the proposed system.
- A data schema, often a relational schema.
- Metadata to define the tables/files and columns/data-items.
- A function hierarchy diagram or web page map that graphically describes the program structure.
- Actual or pseudocode for each module in the program.

- A prototype for the proposed system.

Design Strategies

Top-Down Strategy

The top-down strategy uses the modular approach to develop the design of a system. It is called so because it starts from the top or the highest-level module and moves towards the lowest level modules.
In this technique, the highest-level module or main module for developing the software is identified.
The main module is divided into several smaller and simpler submodules or segments based on the task performed by each module.
Then, each submodule is further subdivided into several submodules of next lower level.
This process of dividing each module into several submodules continues until the lowest level modules, which cannot be further subdivided, are identified.

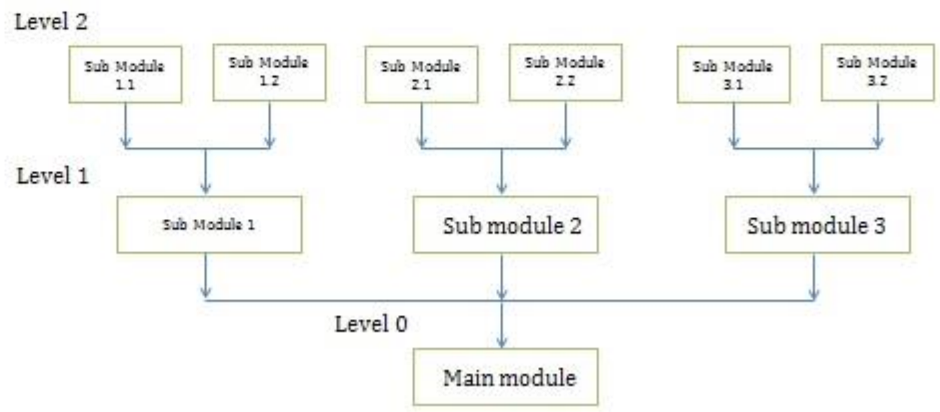


Bottom-Up Strategy

Bottom-Up Strategy follows the modular approach to develop the design of the system. It is called so because it starts from the bottom or the most basic level modules and moves towards the highest-level modules.
In this technique,

- The modules at the most basic or the lowest level are identified.
- These modules are then grouped together based on the function performed by each module to form the next higher-level modules.
- Then, these modules are further combined to form the next higher-level modules.

- This process of grouping several simpler modules to form higher level modules continues until the main module of system development process is achieved.

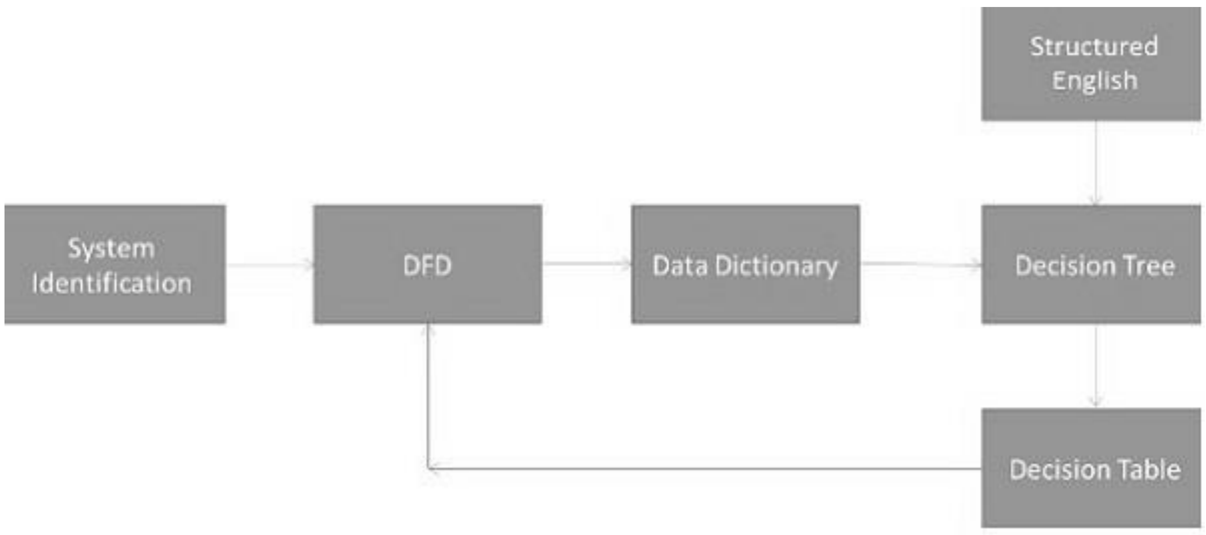


Structured Design

Structured design is a data-flow based methodology that helps in identifying the input and output of the developing system.

The main objective of structured design is to minimize the complexity and increase the modularity of a program. Structured design also helps in describing the functional aspects of the system.

In structured designing, the system specifications act as a basis for graphically representing the flow of data and sequence of processes involved in a software development with the help of DFDs. After developing the DFDs for the software system, the next step is to develop the structure chart.



Modularization

Structured design partitions the program into small and independent modules. These are organized in top-down manner with the details shown in bottom.
Thus, structured design uses an approach called modularization or decomposition to minimize the complexity and to manage the problem by subdividing it into smaller segments.

Advantages

- Critical interfaces are tested first.
- It provides abstraction.
- It allows multiple programmers to work simultaneously.
- It allows code reuse.
- It provides control and improves morale.
- It makes identifying structure easier.

Factors Affecting System Complexity

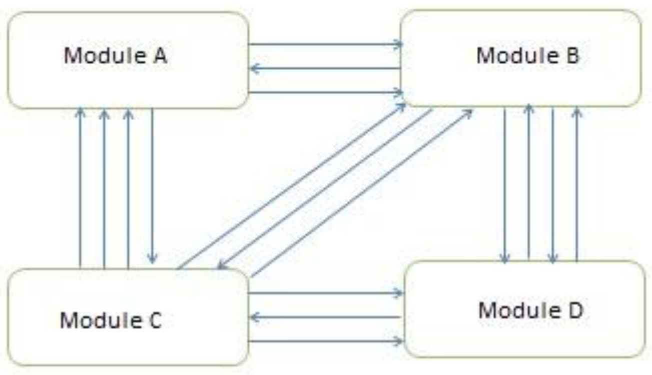
To develop good quality of system software, it is necessary to develop a good design. Therefore, the main focus on while developing the design of the system is the quality of the software design.
A good quality software design is the one, which minimizes the complexity and cost expenditure in software development.
The two important concepts related to the system development that help in determining the complexity of a system are **coupling** and **cohesion**.

Coupling

Coupling is the measure of the independence of components. It defines the degree of dependency of each module of system development on the other.
In practice, this means the stronger the coupling between the modules in a system, the more difficult it is to implement and maintain the system.
Each module should have simple, clean interface with other modules, and that the minimum number of data elements should be shared between modules.

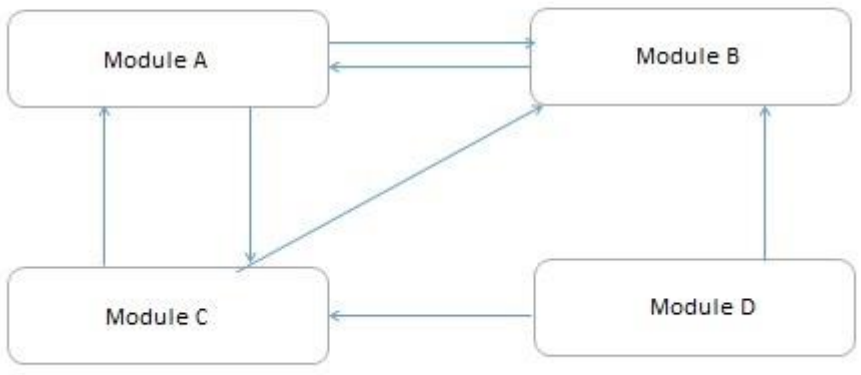
High Coupling

These types of systems have interconnections with program unit's dependent on each other. Changes to one subsystem leads to high impact on the other subsystem.



Low Coupling

These types of systems are made up of components which are independent or almost independent. A change in one subsystem does not affect any other subsystem.



Coupling Measures

- *Content Coupling* − When one component actually modifies another, then the modified component is completely dependent on modifying one.

- *Common Coupling* − When amount of coupling is reduced somewhat by organizing system design so that data are accessible from a common data store.
- *Control Coupling* − When one component passes parameters to control the activity of another component.
- *Stamp Coupling* − When data structures is used to pass information from one component to another.
- *Data Coupling* − When only data is passed then components are connected by this coupling.

Cohesion

Cohesion is the measure of closeness of the relationship between its components. It defines the amount of dependency of the components of a module on one another.
In practice, this means the systems designer must ensure that −

- They do not split essential processes into fragmented modules.
- They do not gather together unrelated processes represented as processes on the DFD into meaningless modules.

The best modules are those that are functionally cohesive. The worst modules are those that are coincidentally cohesive.

Types of cohesion

- *Coincidental cohesion* is found in a component whose parts are unrelated to another.
- *Logical Cohesion* − It is where several logically related functions or data elements are placed in same component.
- *Temporal Cohesion* − It is when a component that is used to initialize a system or set variables performs several functions in sequence, but the functions are related by timing involved.
- *Procedurally Cohesion* − It is when functions are grouped together in a component just to ensure this order.
- *Sequential Cohesion* − It is when the output from one part of a component is the input to the next part of it.

Constraints in Design

A design constraint can be seen as a non-functional requirement which the final product should meet.

A non-functional requirement is a constraint placed on the system or on the development process.

Constraints limit the developer's choices when designing the system-to-be. Therefore, it is necessary to know exactly the boundaries the developer should work within before the design starts.

If failing to find and note down the limits, time may be wasted on trying different designs that would not be possible for the specific project.

Check lists are useful for identifying non-functional requirements.

Type of Non-functional Requirements

- User interface and human factors
  - What type of user will be using the system?
  - Will more than one type of user be using the system?
  - What sort of training will be required for each type of user?
  - Is it particularly important that the system be easy to learn?
  - Is it particularly important that users be protected from making errors?
  - What sort of input/output devices for the human interface are available, and what are their characteristics?

- Documentation
  - What kind of documentation is required?
  - What audience is to be addressed by each document?

- Hardware considerations
  - What hardware is the proposed system to be used on?
  - What are the characteristics of the target hardware, including memory size and auxiliary storage space?

- Performance characteristics
  - Are there any speed, throughput, or response time constraints on the system?

- Are there size or capacity constraints on the data to be processed by the system?

- Error handling and extreme conditions
  - How should the system respond to input errors?
  - How should the system respond to extreme conditions?

- System interfacing
  - Is input coming from systems outside the proposed system?
  - Is output going to systems outside the proposed system?
  - Are there restrictions on the format or medium that must be used for input or output?

- Quality issues
  - What are the requirements for reliability?
  - Must the system trap fault?
  - Is there a maximum acceptable time for restarting the system after a failure?
  - What is the acceptable system downtime per 24-hour period?
  - Is it important that the system be portable (able to move to different hardware or operating system environments)?

- System modifications
  - What parts of the system are likely candidates for later modification?
  - What sorts of modifications are expected?

- Physical environment
  - Where will the target equipment operate?
  - Will the target equipment be in one or several locations?
  - Will the environmental conditions in any way be out of the ordinary (for example, unusual temperatures, vibrations, …....)?

- Security issues
  - Must access to any data or the system itself be controlled?
  - Is physical security an issue?

- Resources and management issues
  - How often will the system be backed up?
  - Who will be responsible for the back up?
  - Who is responsible for system installation?
  - Who will be responsible for system maintenance?

<u>N/B:</u>

Evaluating Designs

When is a design correct?

- If it can be shown to capture all the functions of the requirements document?
- If it captures all the users' requirements?

What makes a design a good design?

- It is correct, complete, consistent, realistic and readable

OTHERS

The price of the product:

Consider how much the product should cost.

This includes sales price, development cost, hardware cost, etc.

Development time:

Consider the time it must take to develop the system-to-be.

When is the production expected to begin?

When is the sale expected to begin?

Performance:

Consider the performance of the product when it is to be used. Is it an expensive high-quality product, or is it a low-price product which is to be sold in large quantities?

If it is a product for which a customer has paid a large amount of money, no defects are accepted.

If the system-to-be is a product which must be safe in use, optimal performance is necessary and malfunctions are not accepted.

Reliability and lifetime:

The reliability and lifetime must be considered.
If the system-to-be is a product which must be safe to use, the reliability and lifetime are paramount; no failures are accepted.

Already developed parts:

In case you need to reuse already developed parts, you need to take this into account early in the project phase, as it will have an effect on the selection of the technical platform.
The reused parts may not fit completely into the system-to-be. You have to consider how they worked in previous projects.

- Do they fit into the new project?
- Was it the design you intended, or did some parts have defects?
- Are they good enough, or will you have to design new parts?

Service and maintenance:

Consider service and maintenance. No matter how well-developed the system-to-be is, it will sometimes fail or must be maintained.
Maintenance can be different sorts of adjustments of the system-to-be, e.g., replacing old hardware or changing software.