Software Design

- Software Design is a process through which requirements are translated into a representation of software.

Software Design – Data Design

- Data design is the first (and sometimes the most important) of the four design activities that are conducted in software engineering.
- Transforms the information domain model created during analysis into the data structures that will be required to implement the software.
- The data objects and relationships defined in the Entity-Relationship Diagram (ERD) and the detailed data content depicted in the Data Dictionary provide the basis for the data design activity.
Software Design – Architectural Design
- Defines the relationship among major structural elements of the program.
- This design representation (the modular framework of a computer program) can be derived from the analysis model and the interaction of subsystems within the analysis model.

Software Design – Interface Design
- Describes how the software communicates within itself, to systems that inter-operates with it, and with humans who use it.
- The Data and Control Flow Diagrams provide the information required for interface design.

Software Design – Procedural Design
- Transforms structural elements of the program architecture into a procedural description of software components.
- Information obtained from the Process and Control Specifications and the State Transition Diagrams serve as a basis for procedural design.

Data Design
- Data design itself is defined and summarized by Wasserman:
  - The primary activity during data design is to select logical representations of data objects (data structures) identified during the requirements definition and specification phase.
  - The selection process may include algorithmic analysis of alternative structures in order to determine the most efficient design or may simply involve the use of a set of modules (a “package”) that provide the desired operations upon some representation of an object.
Data Design Principles (Wasserman)

1. The systematic analysis principles applied to function and behavior should also be applied to data.
2. All data structures and the operations to be performed on each should be identified.
3. A data dictionary should be established and used to define both data and program design.
4. Low-level data design decisions should be deferred until late in the design process. (Top-down Approach)
5. The representation of data structure should be known only to those modules that must make direct use of the data contained within the structure. (Information Hiding)
6. A library of useful data structures and the operations that may be applied to them should be developed. (Reusability)
7. A software design and programming language should support the specification and realization of abstract data types.

Architectural Design

- To develop a modular program structure and represent the control relationships between modules.
- Combines program structure and data structure, thus defining interfaces that will enable data to flow throughout the program.
- Data flow-oriented design is an architectural design method that allows a convenient transition from the analysis model to a design description of program structure.

Data Flow Oriented Design

- The transition from information flow (such as DFD) to structure is typically accomplished as a five step process:
  1. The type of information flow (either transform or transaction flow) is established.
  2. Flow boundaries are indicated.
  3. The DFD is mapped into program structure.
  4. Control hierarchy is defined by Factoring.
  5. The resultant structure is defined using design measures and heuristics.

Data Flow Types: Transform Flow

- Information must enter and exit software in an “external world” form;
  - E.g. data being typed on a keyboard, pictures on a computer graphics display.
- Externalized data coming must be converted into an internal form for processing.
- Information entering the system along paths that transform external data into an internal form are called Incoming Flow.
- At the kernel of the software, a transition occurs: Incoming data are passed through a transform center and begin to move along paths that lead out of the software. Data moving along these paths are called Outgoing Flow.
Data Flow Types: Transform Flow

- A segment or part of a data flow diagram demonstrates these characteristics, transform flow is present.
- **Transform Analysis** will then be applied to change this DFD into an architectural representation.

![Transform Flow Diagram](image)

Data Flow Types: Transaction Flow

- Information flow is characterized by a single data item, called a transaction, that triggers other data flow along one of many paths.
- The transaction is evaluated and, based on its value, flow along one of many action paths is initiated.
- The hub of information flow from which many action paths flow from is called a transaction.

![Transaction Flow Diagram](image)

Transform Mapping

- Transform mapping is a set of design steps that allows a DFD with transform flow characteristics to be mapped into a predefined template for program structure.
Transform Mapping (Design Steps)

1. Review the fundamental system model
2. Review and refine DFD for the software
3. Determine whether the DFD has transform or transaction flow characteristics
4. Isolate the transform center by specifying incoming and outgoing flow boundaries
5. Perform "First-Level Factoring"
6. Perform "Second-Level Factoring"
7. Refine the "First-Cut" program structure using design heuristics for improved software quality

Transform Mapping (Step 1)

- Review the fundamental system model.

Transform Mapping (Step 2)

- Review and refine data flow diagrams for the software.

Transform Mapping (Step 3)

- Determine whether the DFD has transform or transaction flow characteristics.
Transform Mapping (Step 4)

- Isolate the transform center by specifying incoming and outgoing flow boundaries.
  - Incoming and outgoing flow boundaries are open to interpretation, and different designers may select slightly different points in the flow as boundary locations.

Transform Mapping (Step 5)

- Perform “first-level” factoring.
  - Factoring results in a program structure in which top-level modules perform decision making and low-level modules perform most input, computational and output work.
  - Middle-level modules perform some control and do moderate amounts of work.
Transform Mapping (Step 6)

- Perform “second-level” factoring.
  - Individual transforms (bubbles) of a DFD are now mapped into appropriate modules within the program structure.
  - In the next step, modules can be combined or split apart to achieve a good level of cohesion and coupling within the program structure.

Transform Mapping (Step 7)

- Refine the first iteration program structure using design heuristics for improved software quality.
  - Modules can be exploded or imploded to produce sensible factoring, good cohesion, minimal coupling, and most importantly, a structure that can be implemented.
Transform Mapping is a set of design steps that allows a DFD with transaction flow characteristics to be mapped into a predefined template for program structure.

**Transaction Mapping**

1. Review the fundamental system model.
2. Review and refine DFD for the software.
3. Determine whether the DFD has transform or transaction flow characteristics.
4. Identify the Transaction center and the flow of characteristics along each of the action paths.
5. Map the DFD in a program structure amendable to transaction processing.
6. Factor and refine the transaction structure of each action path.
7. Refine the "First-Cut" program structure using design heuristic for improved software quality.

Transaction Mapping (Step 1)

- Review the fundamental system model.
Transaction Mapping (Step 2)

- Review and refine DFD for the software

Transaction Mapping (Step 3)

- Determine whether the DFD has transform or transaction flow characteristics.

Transaction Mapping (Step 4)

- Identify the Transaction center and the flow of characteristics along each of the action paths.
  - The transaction center lies at the origin of a number of action paths that flow radically from it.
Transaction Mapping (Step 5)

- Map the DFD in a program structure amendable to transaction processing.
  - Transaction flow is mapped into a program structure that contains an incoming branch and a dispatch branch. Structure for the incoming branch is developed in the same way as transform mapping.
  - The structure of the dispatch branch contains a dispatcher module that controls all subordinate action modules.

---

Transaction Mapping (Step 6)

- Factor and refine the transaction structure of each action path.
  - Each action path of the data flow diagram has its own information flow characteristics.
  - In other words, it is possible for an action path to demonstrate transform flow characteristics, and therefore necessitate transform mapping.
Transaction Mapping (Step 7)

- Refine the first iteration program structure using design heuristics for improved software quality.