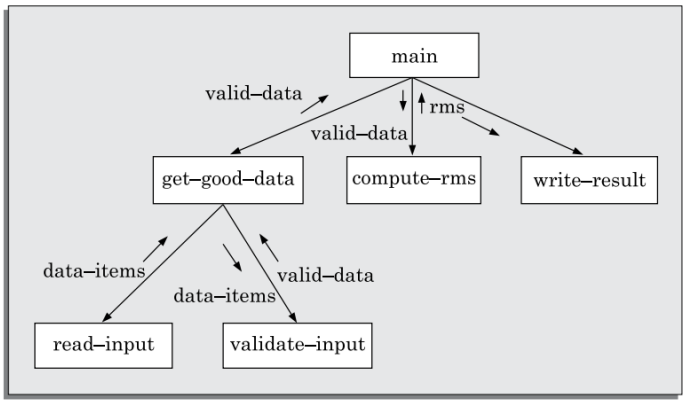
Slide 10 : Structure Chart

Made by Md. Mehedi Hasan Rafy

Structure Chart

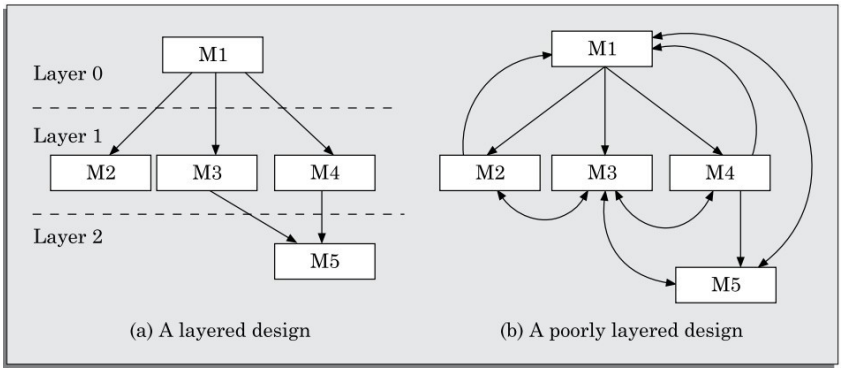
- Shows software architecture with module hierarchy and relationships.
- Displays which module calls which, and what parameters are passed.
- Easy to implement in code later.
- Focuses on module structure and interaction, not on detailed procedures.
- Complements DFD by showing control flow and module breakdown.

Structure Chart: Symbols



- ***Rectangular Box****:* Represents a module (with its name).
- ***Invocation Arrow****:* Shows control is passed from one module to another.
- ***Data Flow Arrow****:* Small arrow with data name; shows data passed between modules.
- ***Library Module****:* Double-edged rectangle; used for shared/reusable modules.
- ***Selection (Decision)****:* Diamond symbol; only one of the connected modules is called based on a condition.
- ***Repetition (Loop)****:* Loop line on arrows; shows repeated module calls.

Structure Chart Rules



- One root module at the top.
- No mutual calls: If A calls B, B cannot call A.
- Modules are arranged in levels (layers).
- Lower-level modules should not know about higher-level modules (abstraction).
- Two higher modules can call the same lower module (shared use allowed).

Flowchart vs Structure Chart

Flowchart	Structure Chart
Shows control flow	Shows module structure
Hard to identify modules	Clearly shows modules
No data flow shown	Shows data flow between modules
Emphasizes sequence	Suppresses sequence

Transformation of a DFD Model into Structure Chart

To convert a DFD into a structure chart, two main strategies are used:

1. ***Transform Analysis***
   - Used when input data is transformed into output through a series of steps.
   - Common in data-processing systems.
   - Focuses on input, central processing, and output modules.

2. ***Transaction Analysis***
   - Used when the system has multiple types of user-initiated operations (transactions).
   - Control module determines which transaction to execute.
   - Each transaction has a separate processing path.

Key Point:
Before applying these, analyze each DFD to decide whether transform or transaction strategy is suitable.

Whether to Apply Transform or Transaction Processing

To decide which analysis to use for converting a DFD to a structure chart:

Transform Analysis

Apply when:

- All input data flow into a single bubble.
- All data is processed in similar ways.
- Typically used at lower levels of DFDs.
- Suitable for simple, linear processing.

Transaction Analysis

Apply when:

- Different input data flows go to different bubbles.
- System handles multiple types of operations.

- Each type corresponds to a distinct transaction/functionality.
- More appropriate for top-level DFDs or complex systems.

Transform Analysis

Transform analysis helps convert a DFD into a structure chart by identifying the key functional parts.

Steps in Transform Analysis: Divide the DFD into three logical regions:

1. ***Input (Afferent Branch)***

   Transforms *physical input → logical data structures*

   E.g., reading user input, parsing files

2. ***Processing (Central Transform)***

   The main computation or decision-making part of the system

3. ***Output (Efferent Branch)***

   Transforms *logical data → physical output*

   E.g., formatting for display, writing to printer or file