Structured Chart

A structure chart is a design tool used to illustrate the hierarchical relationships and interactions between the components of a software system. It is often used in the context of structured design and shows the organization of different modules and functions within the program. The chart helps to visualize the decomposition of the software into modules and sub-modules, making it easier to understand the system's architecture

Structured charts are a visual representation tool to design and communicate the structure of algorithms or processes. It represents the hierarchy and flow of control within a program.

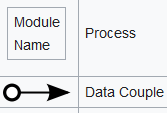
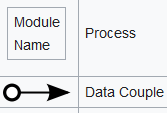
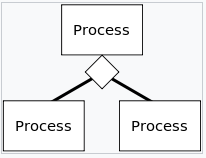
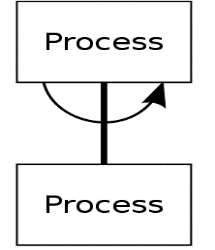
Each box in the chart represents a module or a function, and arrows depict the flow of control between them.

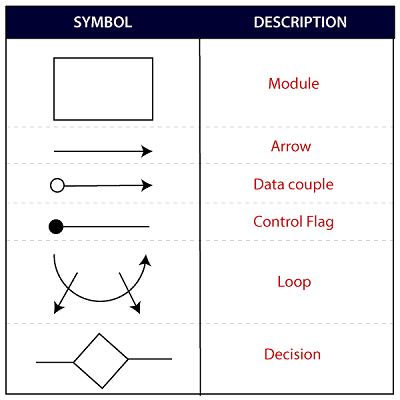
Also used to visually represent the modular structure of a program.

They help programmers understand the relationships and interactions between subsection of the program.

Symbols used

Different symbols are used to represent different elements in a structured chart.

* **Module name:** Represents a process/module/function. It signifies a specific action or operation that is carried out as part of the program. Example this might be something that calculates the average of some figures, or prints out some pay slips
* **Data couple** An arrow is used to depict the flow of control or data between different processes or functions. The direction of the arrow indicates the sequence in which processes are executed. Data being passed from module to module that needs to be processed.
* **Diamond:** Represents a decision point (if-else condition). It indicates a branching point in the program, where the flow of control depends on a certain condition being true or false
* Using the semi circular arrow we can represent iteration. The arrow encompasses a link to a module, implying that module is executed multiple times



* **Modules**: Represented by rectangular boxes, these are the components of the software system.
* **Calling Relationships**: Arrows or lines between the boxes show which modules call or are called by other modules.
* **Data Flow**: Sometimes, data flow between modules is represented by arrows, with annotations describing the nature of the data being passed.
* **Control Flow**: The sequence and conditions under which various modules are activated.

Benefits

* facilitate a modular approach to software design, breaking down the system into discrete, manageable modules or components.
* Provide a clear and concise visual representation of the system's components and their relationships, making it easier to understand complex systems.
* Serve as a communication tool among various stakeholders, including developers, designers, analysts, and managers, enabling them to discuss and collaborate on the system's design effectively.
* Structure charts are part of the system documentation, which can be useful for new team members to get up to speed, and for maintenance purposes after the system has been developed.
* Clearly show the hierarchical relationship between various modules, including parent-child relationships and the levels of abstraction within the system.
* Help in visualizing the control flow between different modules, making it easier to follow the flow of data and control through the program.
* By planning and visualizing the system structure before the actual coding starts, structure charts can help to reduce the number of errors and omissions.
* Make it easier to update and maintain the software, as the impact of changes can be more readily assessed and managed.
* Structure charts can help identify common modules that can be reused in different parts of the system, reducing redundancy and improving consistency.
* They aid in decomposing a complex problem into simpler sub-problems that can be more easily managed and solved.