5. HIPO makes the transformation of input to output visible.

There are two parts of a complete HIPO package. First the hierarchy chart which represents the top-down structure of the program. For each and every module that is shown in the hierarchy chart, an input-process-output (IPO) chart is used that describes the input, output and the processes performed by the modules.

The HIPO/IPO package format consists of the following:

1. **Hierarchy chart**: Visual table of contents that shows the structure of the diagram and the relationship of the function in a hierarchical manner. It summarizes the primary tasks to be performed by the system. Each box represents a module. The modules call its subordinates and returns control to its parents.
2. **Visual table of contents**: It is a more formal name for the hierarchy chart.
3. **IPO chart**: It is prepared to document each module present in the hierarchy chart.
4. **Overview diagram**: It describes the major functions. It references the detailed diagram needed to expand the functions adequately. It is a high level IPO chart. It summarizes the inputs, outputs and processes to be performed by the modules. Overview diagram acts as planning tools. They usually don't appear in the complete documentation process.
5. **Detail diagram**: It contains an extended description section that amplifies the process steps and references the code associated with each process step. It is a low-level IPO chart. It shows how particular input and output data elements are linked to particular processes.

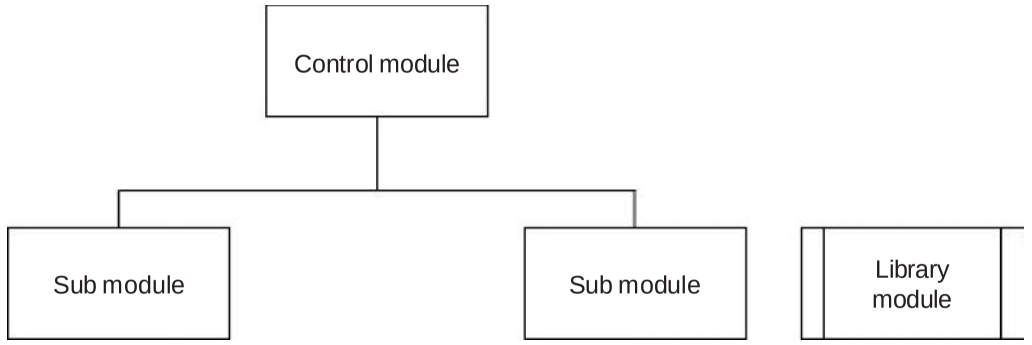


Figure 9.3: Modules

Conditional call:

It represents that based on some condition control module can select any of the sub module. It is presented in figure 9.4.

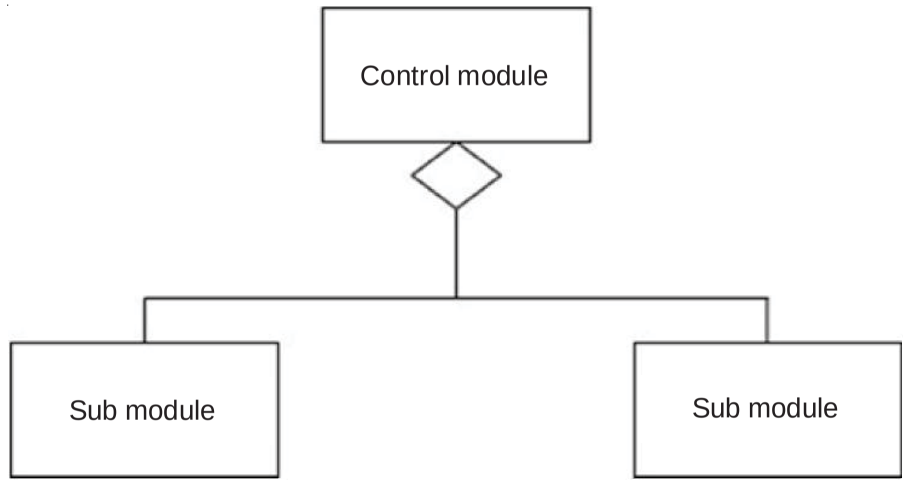


Figure 9.4: Conditional call

Loop

It represents a module to be executed repeatedly by the sub module till some condition becomes false. The loop in the module is represented by a curved arrow as depicted in figure 9.5.

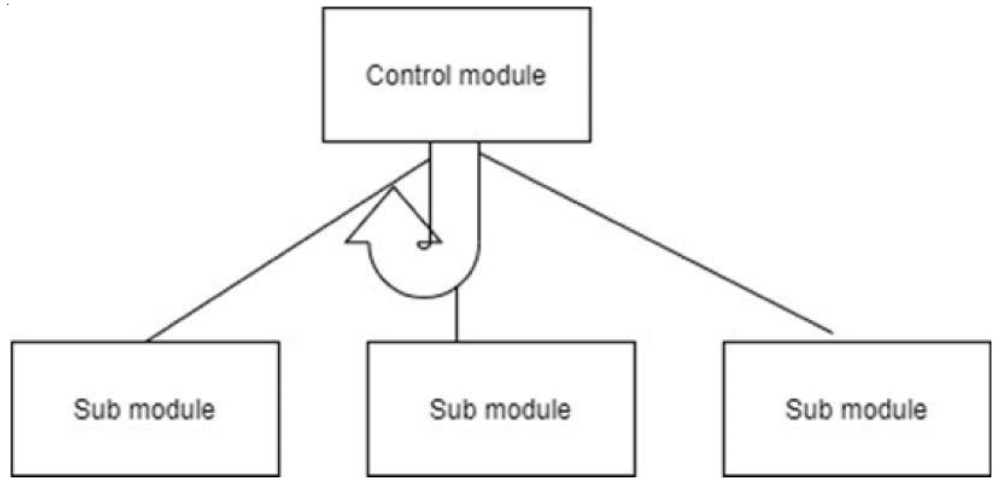


Figure 9.5: Module loop

Data flow

It represents the data flow between the modules. It is represented by an arrow with an empty circle at the end. Figure 9.6 shows module data flow.

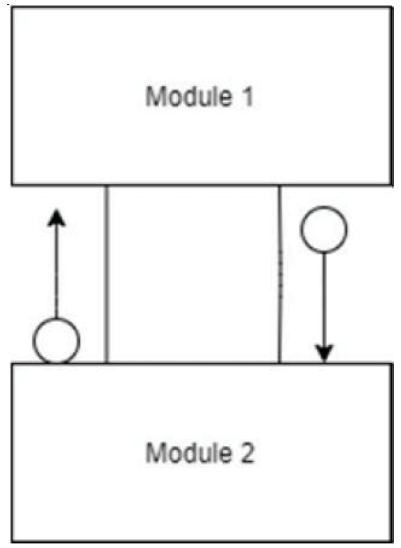


Figure 9.6: Module data flow

Control flow

It represents the flow control between the modules. It is represented by an arrow with a filled circle at the end as presented in figure 9.7.



CHECK YOUR PROGRESS

Q1. Mention the main differences between logical and physical design?

Q2. Draw the structure chart of a food ordering system.

Q3. What is the difference between coupling and cohesion?

9.6 FORM-DRIVEN METHODOLOGY

In structured design, a hierarchy chart represents a good program design if it meets the criteria of cohesion and coupling. Each module should perform a single function (cohesion) and should be independent of the rest of the program (coupling). Each criterion calls for more details. This prompts the analyst to develop input/process/output (IPO) charts for each module in the hierarchy chart.

9.6.1 HIPO And IPO Chart

HIPO is a form-driven technique. Here standard documents are used to document the information. It consists of a hierarchy chart and a set of associated input/process/output chart. HIPO follows top-down decomposition. It describes clearly the data input and output from processes. It also defines the data flow composition. HIPO/ IPO charts weredeveloped by IBM. Figure 9.12 presents Hierarchical plus Input-Process-Output (HIPO) technique and Input-Process-Output (IPO) diagrams

Purpose of HIPO/IPO:

1. To understand structure and functions of systems.
2. To provide easily understandable description of various parts of the system.
3. State the function to be performed by the programmer rather than specifying program statement to be used to perform the function.
4. Provide the visual description of input to be used and output to be produced for each level of the diagram.

Cohesion

The inter-module strength is known as cohesion. It is the measure of the closeness of the relationship between its components. The amount of dependency of the components of a module with one another is defined by cohesion. Functionally cohesive modules are the best modules. Coincidentally cohesive modules are the worst modules. Functional cohesion is said to occur when all the components of the module are grouped together for a single, well-defined purpose. Coincidental cohesion is said to occur when components of modules are grouped together arbitrarily without any meaningful relationships. The system designer must ensure the following in order to achieve good cohesion—

- Essential processes should not be split into fragmented modules.
- Unrelated processes represented as procedures on data flow diagram should not be gathered together into meaningless modules.

Different types of cohesion are—

- Logical cohesion: It occurs when many logically related data elements grouped together in the same component.
- Temporal cohesion: It occurs when a component that is used to initialize a system or a set of data elements perform several functions in sequence but the functions are related to each other in terms of time.
- Procedurally cohesion: It occurs when components are grouped together to maintain order or procedure.
- Sequential cohesion: It occurs when the output of one component is the input of another component.

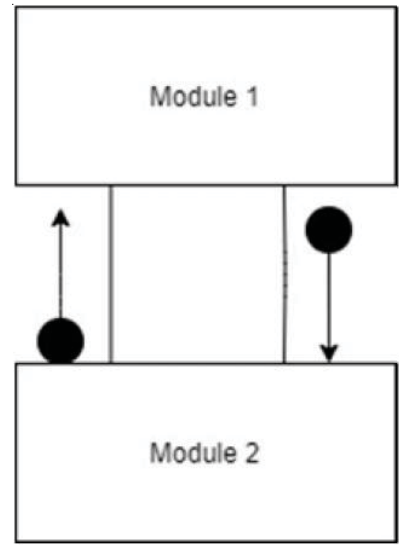


Figure 9.7: Module control flow

Physical storage

It represents storage of all information in a place as shown in figure 9.8.

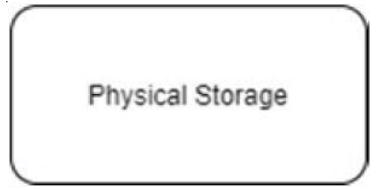


Figure 9.8: Physical storage

Let us draw the structure chart to calculate the average of numbers. The structure chart in figure 9.9 shows the data flow between the modules. “Calculate average” is the control module. “Input numbers, Average and Output average” are the sub modules.

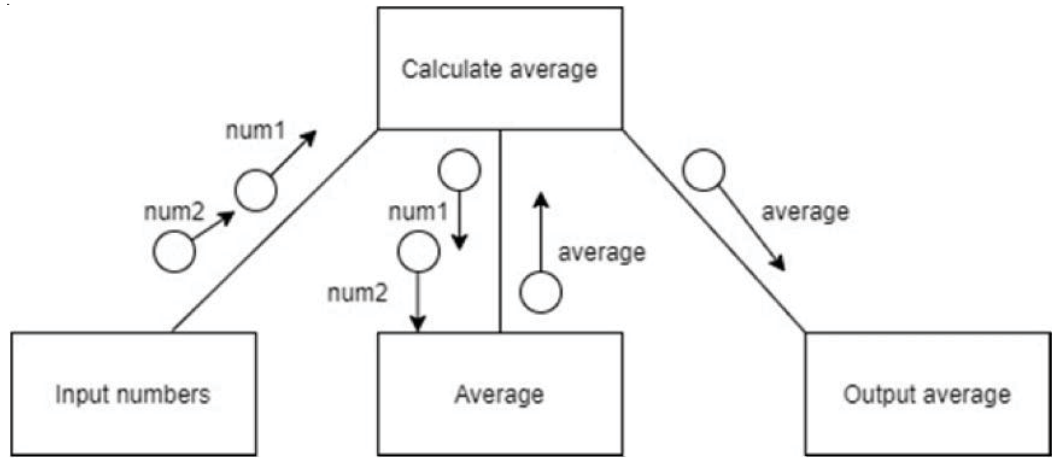


Figure 9.9: Structure chart to calculate average

Objectives of using structured charts are:

1. Encourage top-down design.
2. Support concept of modules and identify appropriate modules.
3. Show size and complexity of the system.
4. To identify number of readily identifiable functions and modules within each function.

9.5.3 Factors Effecting System Complexity

In functional decomposition approach to structured design, a system is partitioned into smaller modules so that each module is made manageable. In order to create a proper quality of system software, it is very important to develop a good design. A system is considered to be of good design if the complexity and cost expenditure in software development is minimum. In the evaluation of the modules, two criteria are considered: Coupling and cohesion. They help in determining the complexity of the system.

Coupling:

The module's connection to other module is called coupling. It defines the degree of dependency of each module on another. Stronger the coupling between modules, the more difficult it is to implement and maintain the system. In high coupling systems as

depicted in figure 9.10, modules are highly dependent on one another. Changes to one subsystem may lead to high impact on the other subsystems.

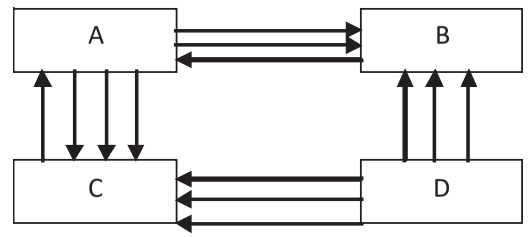


Figure 9.10: High Coupling

In low coupling systems (depicted in figure 9.11), modules are less dependent on each other or almost independent. A change in one system doesn't affect the other subsystems.

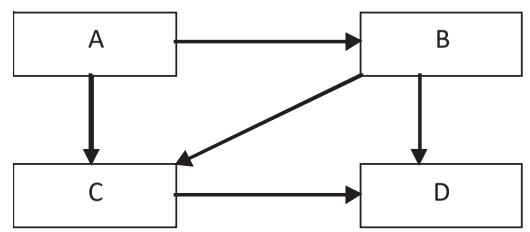


Figure 9.11: Low Coupling

Coupling measures:

1. Content coupling: Here one module actually modifies another module. The modified componentbecomes completely dependent on the modifying component.
2. Common coupling: Here amount of coupling is reduced by organizing the system design so that the data are accessible from a common data store.
3. Control coupling: Here one module passes parameter to control the activity of another module.
4. Stamp coupling: Here data structures are used to pass information from one module to another.
5. Data coupling: Here only data is passed between modules.