Function-Oriented Design

Instructor: Yongjie Zheng
February 19, 2020

CS 441: Software Engineering

How do we design architecture?

- Creativity
    - This requires extensive experience, broad training, ...
- Principles, process, and methods
    - Goals, activities, and principles
    - Design methods: object-oriented design, functional design, and quality-driven design
- Reuse
    - Horizontal reuse: architecture patterns and styles
    - Vertical reuse: product-line architectures

Design Methods

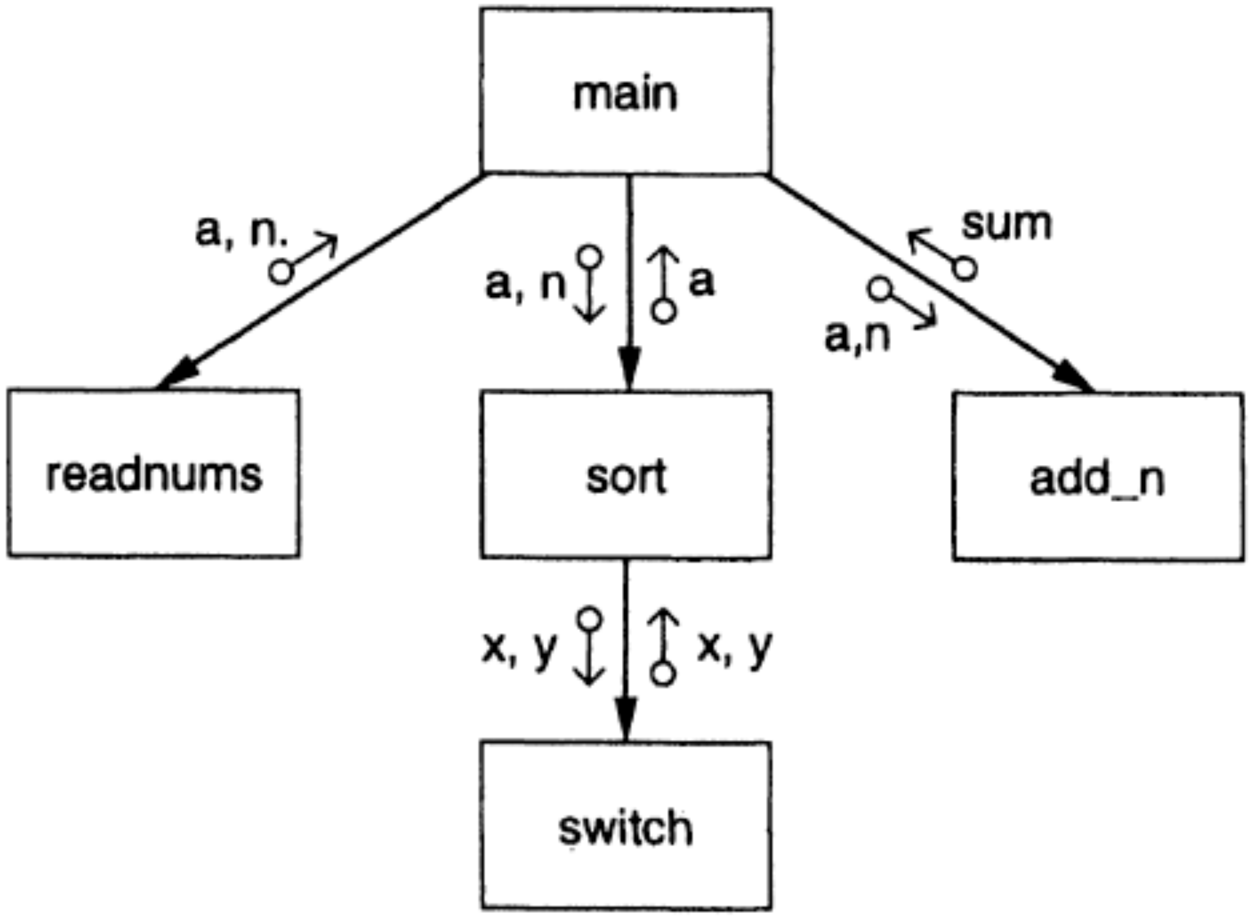
- Function-Oriented Design
  - Views a system as a transformation function that transforms specified input to specified output.
  - Separates data and procedures, and models them separately.
  - The state information is usually in centralized data stores.
- Object-Oriented Design
  - Views a system as a group of objects that interact to satisfy system requirements.
  - Combines data and methods into a cohesive entity.
  - The state information is distributed among system objects.

Function-Oriented Design

- Design notation: structure charts
- Design method: structured design method

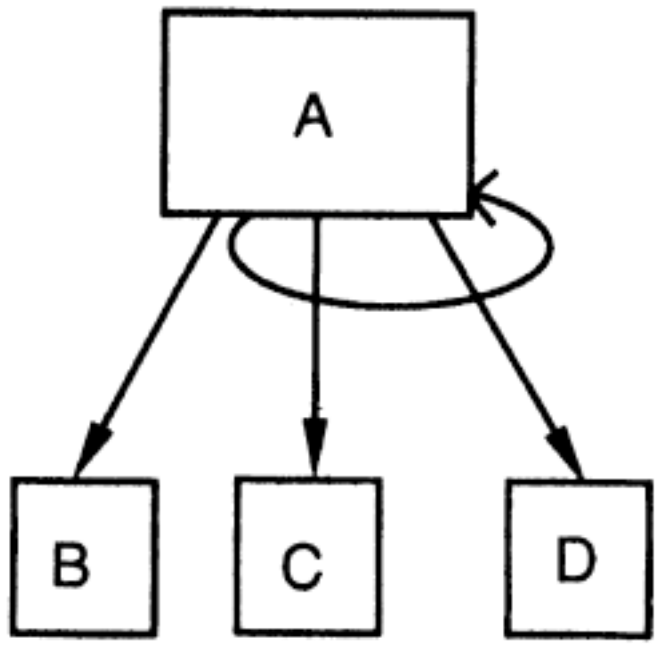
Structure Charts

- In a structure chart, a module is represented by a box with the module name written in the box.
- An arrow from module A to module B represents that module A invokes module B.
  - B is called the *subordinate* of A;
  - A is called the *superordinate* of B.
- The arrow is labeled by the parameters received by B as input and the parameters returned by B as output, with the direction of flow of the input and output parameters represented by small arrows.

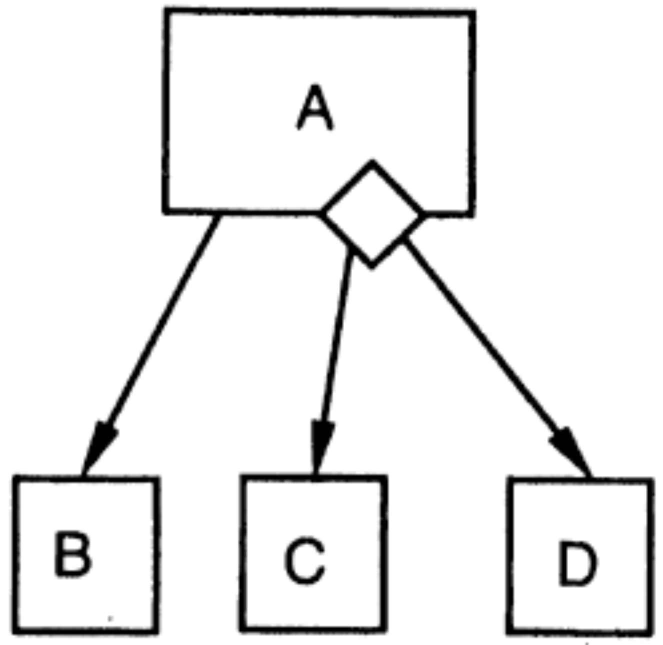


An example of structure charts

Iteration and decision can be explicitly represented.



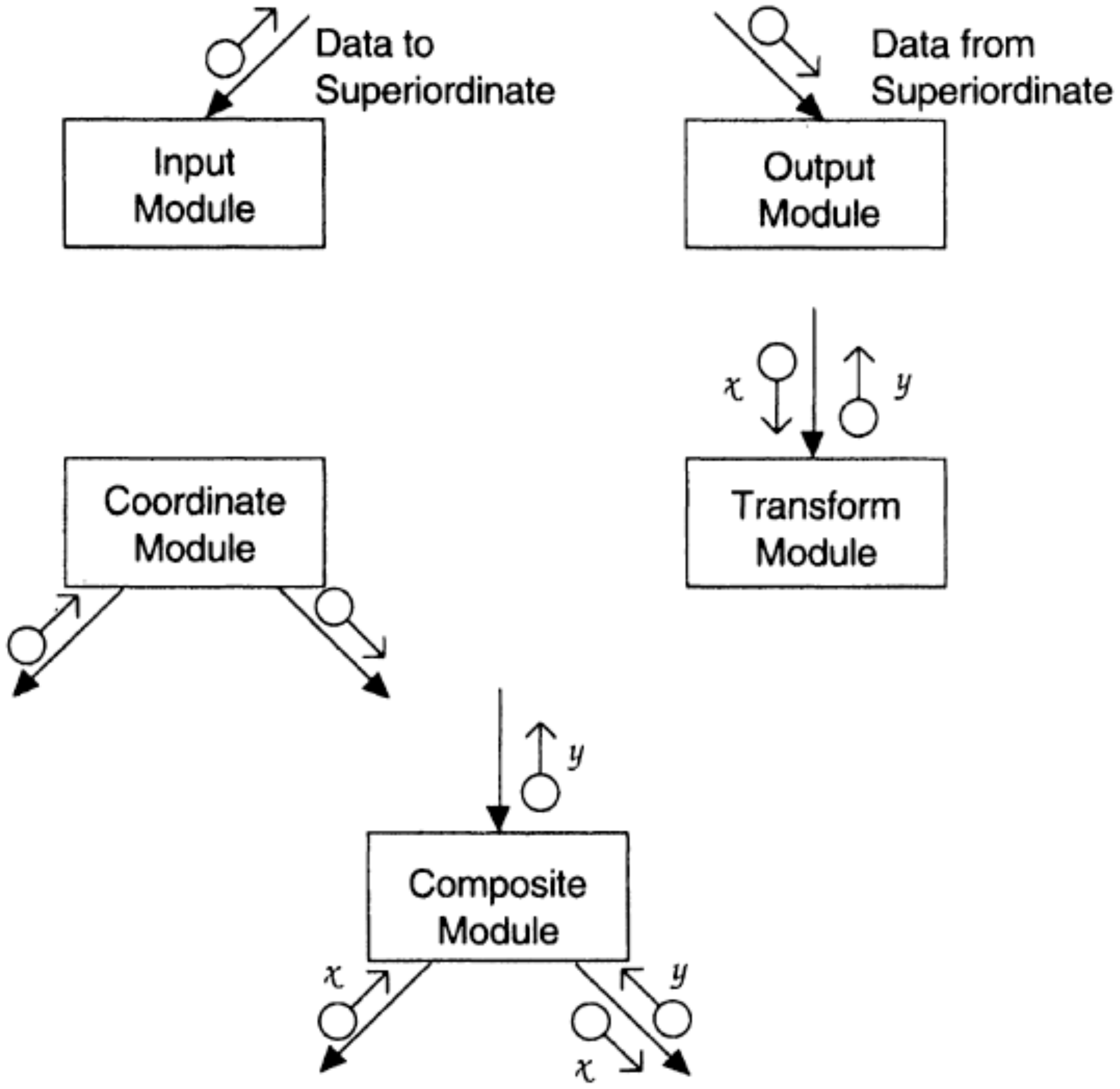
A repeatedly calls modules C and D.



The invocation of modules C and D in module A depends on the outcome of some decision.

Types of modules

- **Input module**: gets the data from the sources and gets it ready to be processed.
- **Output module**: takes the output produced and prepares it for proper presentation to the environment.
- **Transform module**: transforms data into some other form.
- **Coordinate module**: manages the flow of data to and from different subordinates.
- **Composite module**: performs functions of more than one type of module


Data to Superiordinate
Data from Superiordinate
Input Module
Output Module
x
y
Coordinate Module
Transform Module
y
Composite Module
x
y
x

Structured Design Method (SDM)

SDM is a function-oriented design method that views every software system as a transformation function that transforms the given inputs into the desired outputs, and the central problem of designing software systems is considered to be properly designing this transformation function.

Overall strategy: identify the input and output streams and the primary transformations that have to be performed to produce the output. High-level modules are then created to perform these major activities, which are later refined.

Structured Design

Factoring: the process of decomposing a module so that the bulk of its work is done by its subordinates.

A system is said to be completely factored if all the actual processing is accomplished by bottom-level atomic modules and if non-atomic modules largely perform the jobs of control and coordination.

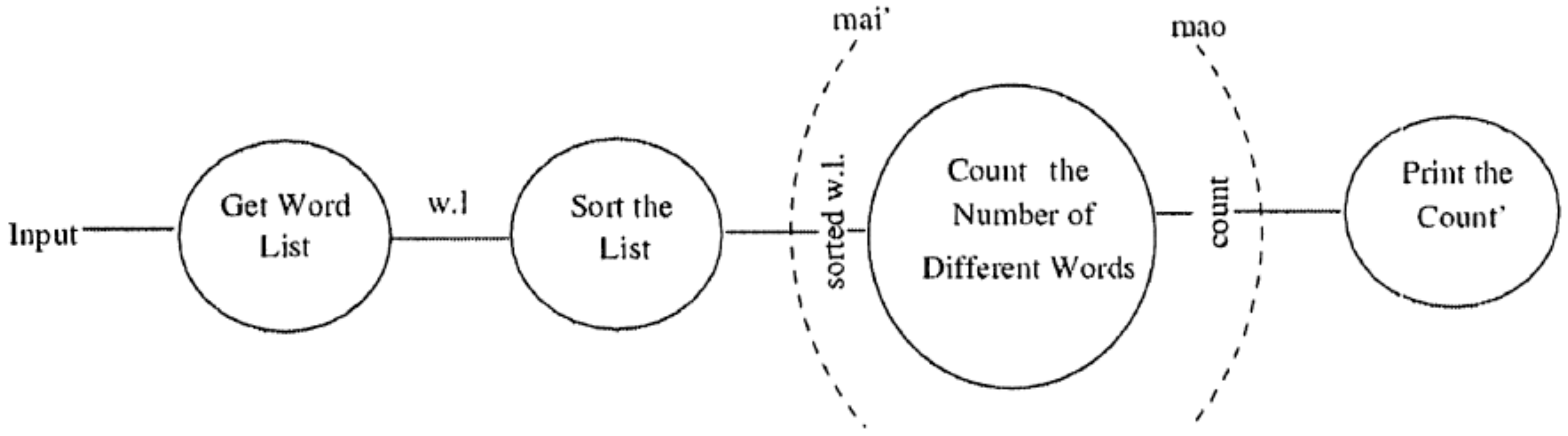
Structured design attempts to achieve a structure that is close to being completely factored.

Four steps of structured design

1. Restate the problem as a data flow diagram.
2. Identify the input and output data elements.
3. First-level factoring.
4. Factoring of input, output, and transform branches.

Step 1: Restate the Problem as a Data Flow Diagram (DFD)

A DFD shows the major transforms that the software will have and how the data will flow through different transforms.



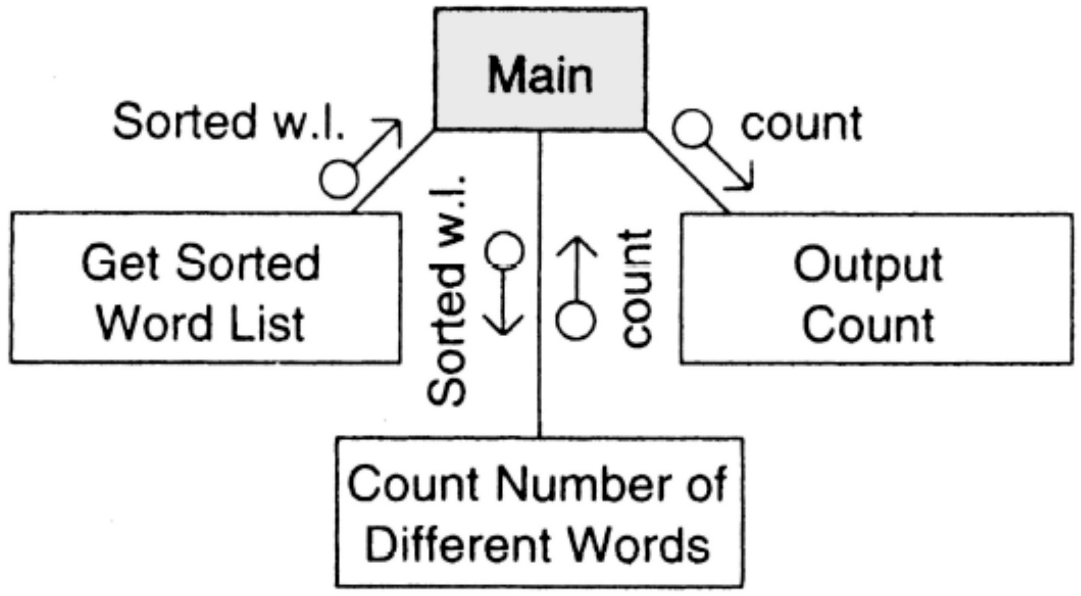
Example: count the number of different words in an input file.

Step 2: Identify the Most Abstract Input and Output Data Elements

- **Most Abstract Input** (MAI): the data elements in the data flow diagram that are farthest from the physical inputs but can still be considered inputs to the system.
- **Most Abstract Output** (MAO): the data elements that are farthest from the actual outputs but can still be considered outgoing.
- **Central Transform**: the transforms left between MAI and MAO that take MAI and transform it into MAO.

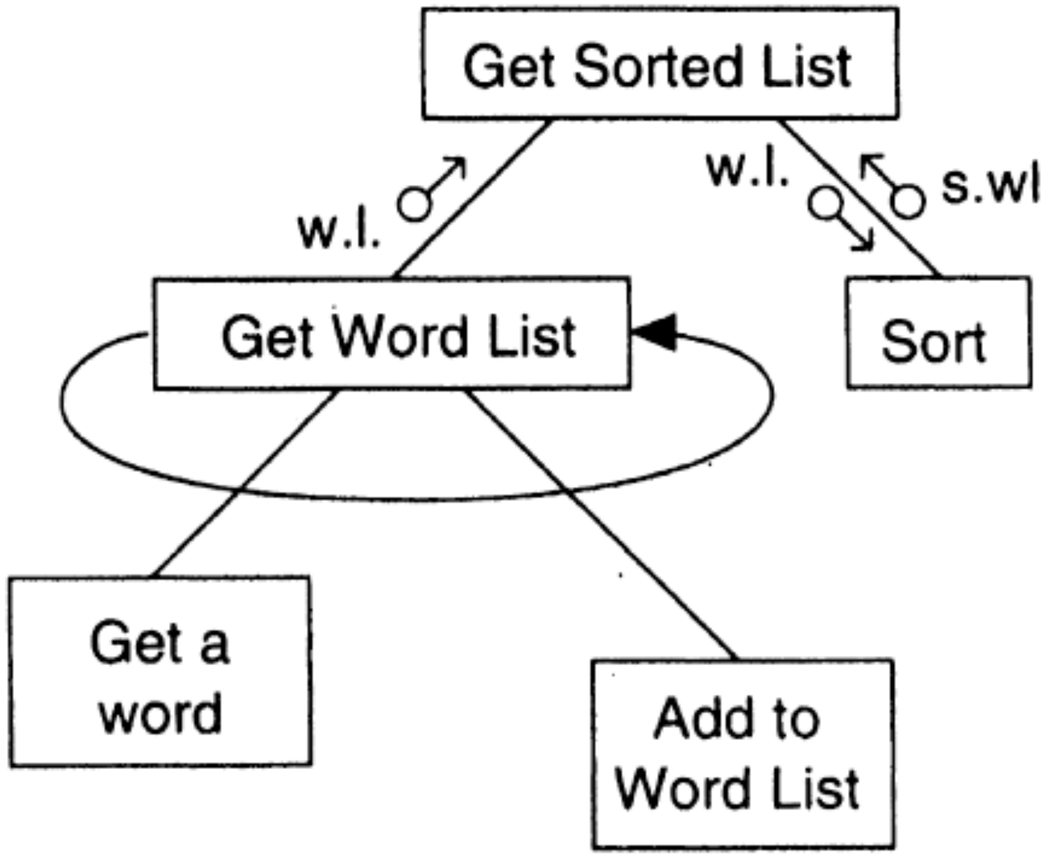
Step 3: First-Level Factoring

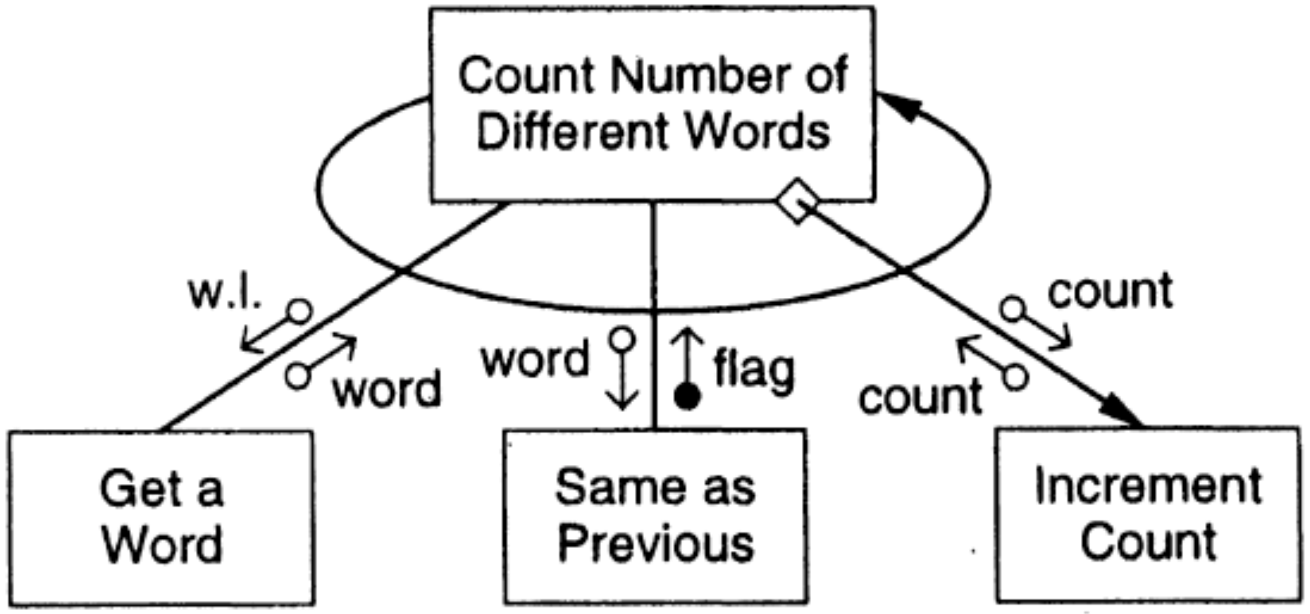
- First, create a main module.
- For each MAI data item, create an immediate subordinate module that delivers data to the main module.
- For each MAO data item, create an immediate subordinate module that accepts data from the main module.
- For each central transform, create an immediate subordinate module that accepts data from the main module, and then return the appropriate data back to the main module.



Step 4: Factoring the Input, Output, and Transform Branches

- For an input module, we look at the previous transform applied to the input to bring it closer to MAI. This now becomes the central transform, and an input module is created for each data stream going into this transform.
- Similar strategy applied to output modules.

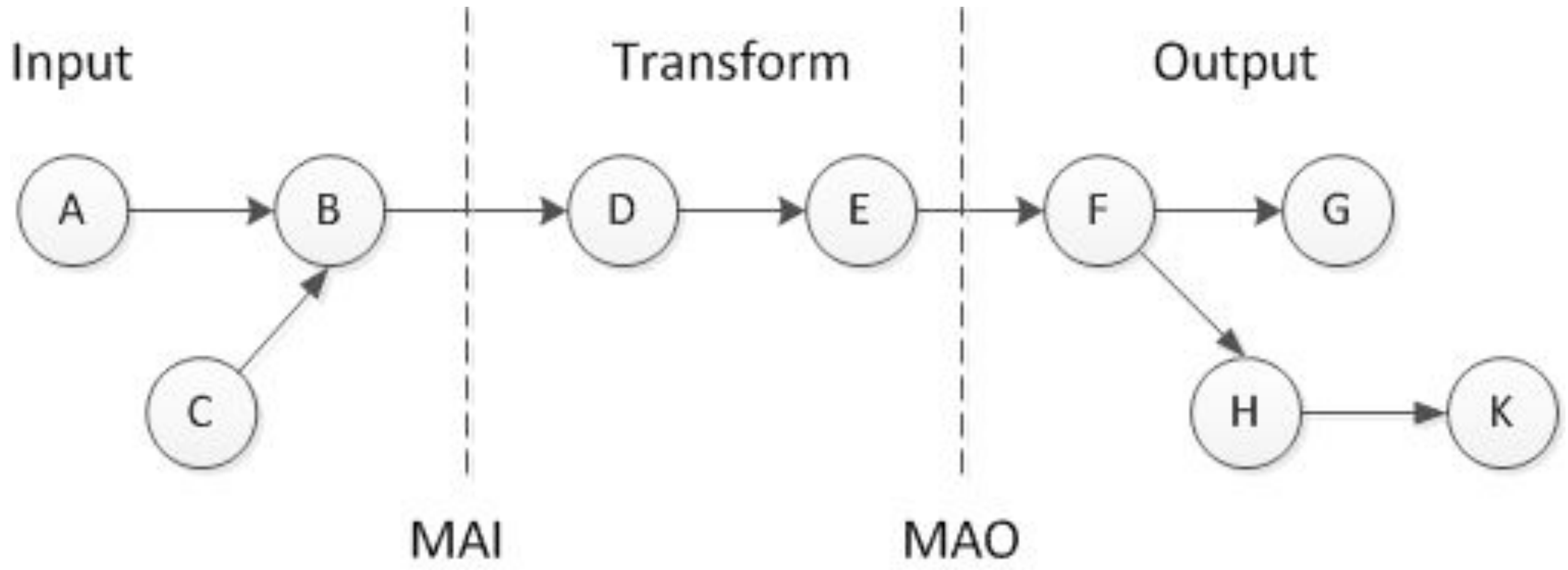




Additional Considerations

- Module size
- “Fan-in” and “Fan-out” of modules
  - Fan-in of a module is the number of arrows coming in the module, or the number of its superordinates.
  - Fan-out of a module is the number of arrows going out of that module, or the number of its subordinates.
  - A high fan-in means hat the module is used by many modules either because it includes different functions (bad) or because it contains a common function (good).
  - A high fan-out means that the module coordinates too many modules and may be too complex.

Exercise



Based on the provided data flow diagram, finish the remaining two steps of structured design and draw the final structure chart.

Reference

- The materials about function-oriented design is based on Chapter 6 of the book "An Integrated Approach to Software Engineering" by Pankaj Jalote.