THE NOTATION

- Expressive Notation
- Models of OOD
- Logical vs Physical Models
- Static vs Dynamic Models
- Kinds of Diagrams
- Products of OOD

EXPRESSIVE NOTATION

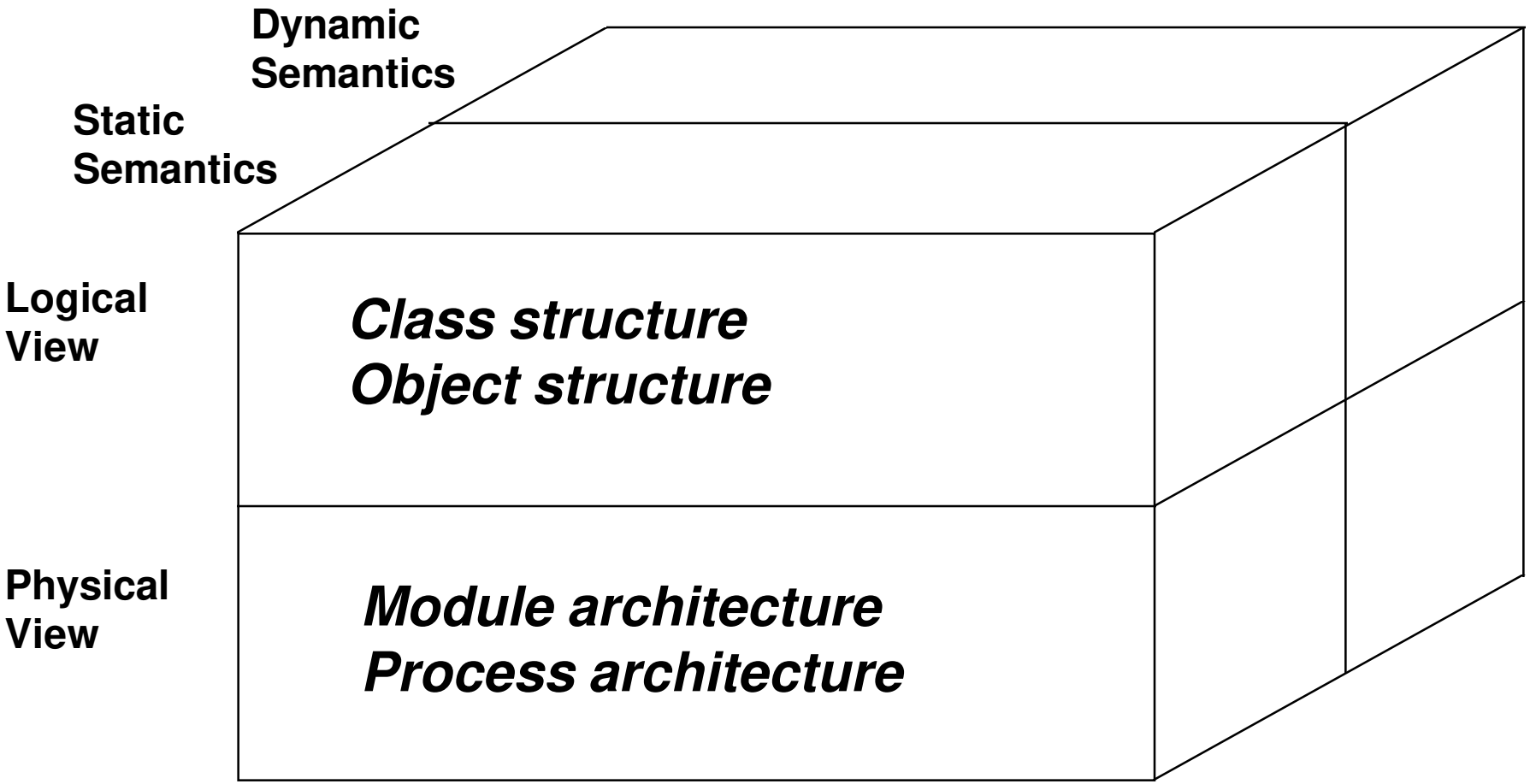
Designing is not the act of drawing a diagram -- a diagram simply captures a design. Having a well-defined and expressive notation is important.

Why is having an expressive notation important?

- **Standard notation makes it possible for one person to formulate a design and then communicate it to others.**
- **A good notation sets the brain free to concentrate on more advanced problems.**
- **An expressive notation makes it possible to eliminate much of the tedium of checking the consistency and correctness of a design by using automated tools.**

However, one picture is often not enough by itself. It is impossible to capture all the subtle details of a complex software system in just one kind of diagram, and in OOD we have found six kinds of diagrams that effectively capture the information we need.

MODELS OF OOD



These models of OOD allow a developer to capture all of the interesting design decisions one must make, and they are complete enough to serve a blueprints for the implementation.

LOGICAL vs PHYSICAL MODELS

A developer must consider the following issues in OOD, and the indicated types of diagrams help him in this task:

Issue	*Diagram*	*View*
What classes exist and how are they related?	Class	Logical
What mechanisms are used to regulate how objects collaborate?	Object	Logical
Where should each class and object be declared?	Module	Physical
To what processor should a process be allocated, and for a given processor, how should its multiple processes be scheduled?	Process	Physical

STATIC vs DYNAMIC SEMANTICS

The dynamic semantics of a designed can be expressed through two additional diagrams:

- **State Transition Diagrams (STD's)**
- **Timing Diagrams**

KINDS OF DIAGRAMS

Class Diagram **- used to show the existence of classes and their relationships**

Object Diagram **- used to show the existence of objects and their relationships**

State Transition Diagram **- used to show the state space of a class, the events that cause a transition from one state to another, and the actions that result from a state change**

Timing Diagram **- used to show the sequence of execution of operations on objects**

Module Diagram **- used to show the allocation of classes and objects to modules**

Process Diagram **- used to show the allocation of processes to processors**

Refer to Chapter 5 of the text for a more thorough discussion of the notation used in the book. Often, the selection of the notation used for these diagrams depends on the automated tools available and the notation they use.

PRODUCTS OF OOD

The top-level view of a design is found in:

- **the topmost class diagram, showing the major abstractions in the logical design**
- **the topmost module diagram, showing the key elements of the physical software design**
- **the main process diagram, showing the key elements of the physical hardware design**

The key mechanisms are found in the object diagrams.

End-to-end connectivity exists:

- **Starting with a process diagram, a processor may be assigned a main program**
- **The main program is shown in a module diagram**
- **The module diagram shows a collection of classes and objects which are shown in their class and object diagrams**
- **The definitions of classes point to the requirements**