DESIGN METHODOLOGIES AND GRAPHICAL NOTATION

- Diagramming Notations
- Data Flow Analysis Methods
- Data Flow Diagrams
- Data Dictionary and Its Content
- Functional Analysis Methods
- Function Diagrams
- State Transition Diagrams
- Object Diagram Conventions
- Entity Relationship Diagrams
- Object Interaction Diagrams
- Booch Diagrams
- Design Methodologies

Objectives of Module 2

- Present and discuss many of the common diagram notations used during requirements analysis and design:
  - Data Flow Diagrams (DFD's)
  - Function Diagrams
  - State Transition Diagrams (STD's)
  - Entity Relationship Diagrams (ERD's)
  - Object Interaction Diagrams (OID's)
  - Booch Diagrams
- Present and discuss the concept of the Data Dictionary and its content
- Present and discuss several common design methodologies:
  - Data Flow-Oriented Design
  - Data Structure-Oriented Design
  - Object-Oriented Design
  - Real-Time Design

DIAGRAMMING NOTATIONS

Many diagramming notations are used during both requirements analysis and design:

- **Data Flow Diagrams**
- **Function Diagrams**
- **State Transition Diagrams**
- **Entity Relationship Diagrams**

Other diagramming notations are intended specifically for design and are often language-specific. These notations are often used when the implementation language is Ada:

- **Object Interaction Diagrams**
- **Booch Diagrams**

The applicaiton of these and other diagramming notations is a part of an organization's software development process.

While "a picture is worth a thousand words," several different kinds of pictures are necessary to analyze the requirements for and design industrial-strength software. Consequently, several different notations have emerged to show different aspects of a system.

What do we want to know about a system in order to understand its requirements and then design software to implement it? Here are just a few:

- Functions performed by the system
- Data which flows in a system and its attributes
- The states of a system and the events which cause the transition from one state to another
- The entities which comprise a system and their attributes and relationships

After we understand these and other aspects of a system, our design can also be expressed graphically. Similar information is shown in the diagrams representing the design along with:

- The objects and classes of objects which comprise the design of the system
- The relationships between the objects in the design and how they interact with each other
- The dependencies between the classes of objects in the system

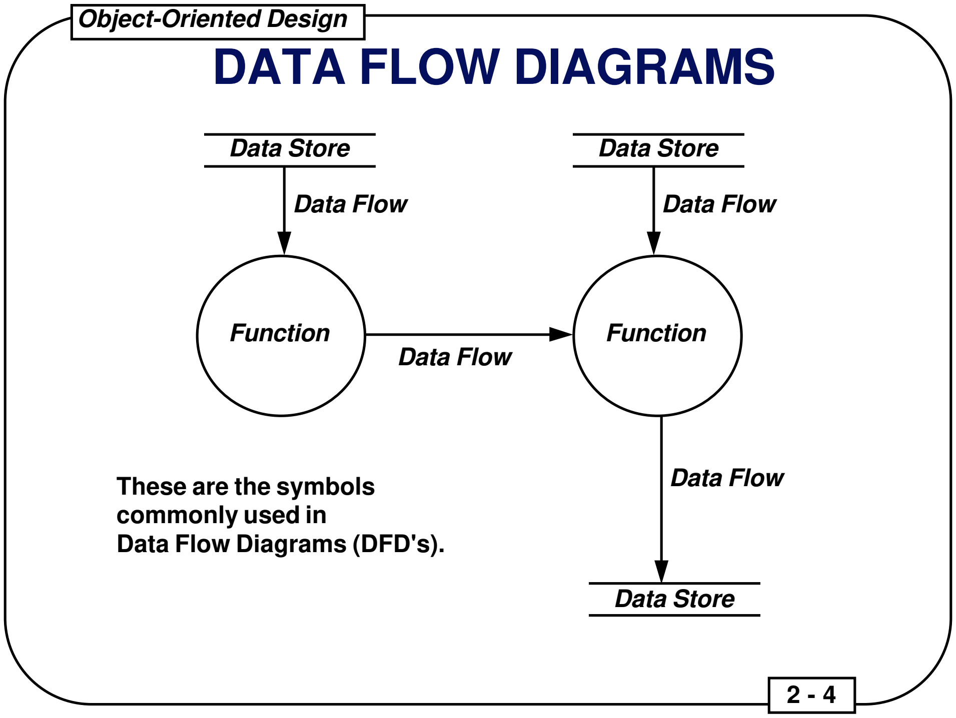
DATA FLOW ANALYSIS METHODS

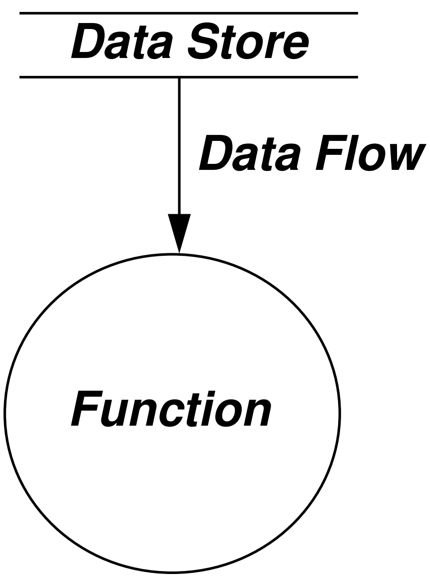
✓ **Data Flow Diagrams tell us:**

- Data Sources and Sinks in the System
- Flow of Data in the System
- Functions which Transform the Data in the System
- Functions which cause Data Transactions in the System

✓ **Data Dictionary tells us:**

- Attributes of the Data in the System
- Other Information about the Data in the System

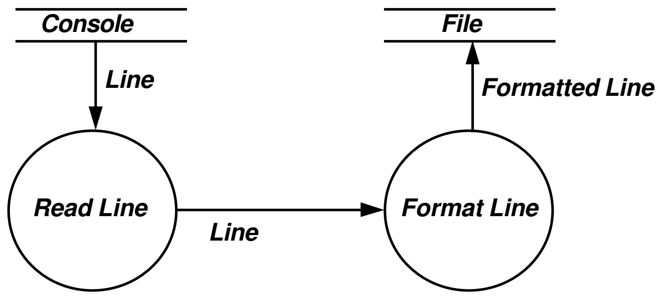




Three basic symbols are in a DFD:

- A data store, which is a source or a sink of data
- A data flow, which shows the flow of data within a system
- A function, which usually indicates a transform or a transaction on the data flowing in the system

DATA FLOW DIAGRAMS EXAMPLE



DATA DICTIONARY AND ITS CONTENT

- **Each class of objects in the system and its attributes**
- **Each singular object (i.e., if placed into a class, the class would have only one instance) and its attributes**
- **Key constants and their attributes**
- **Subprogram parameters and their attributes**

A Data Dictionary is usually maintained in a database. Fields which comprise a record for a data dictionary entry are include:

- Name of entity
- Type (class) of entity, such as integer, 40-character string, etc. The type of an entity includes or implies information on its size (such as 16 bits or 40 bytes)
- Units, such as meters or liters/second
- Key attributes not covered in the Type or Units fields
- Comments

Computer-Aided Software Engineering

(CASE)

Tools

CASE tools provide two fundamental capabilities in an integrated fashion:

1. The ability to create diagrams using certain notations
2. The ability to create a data dictionary and associate items on the diagrams with entries in the data dictionary

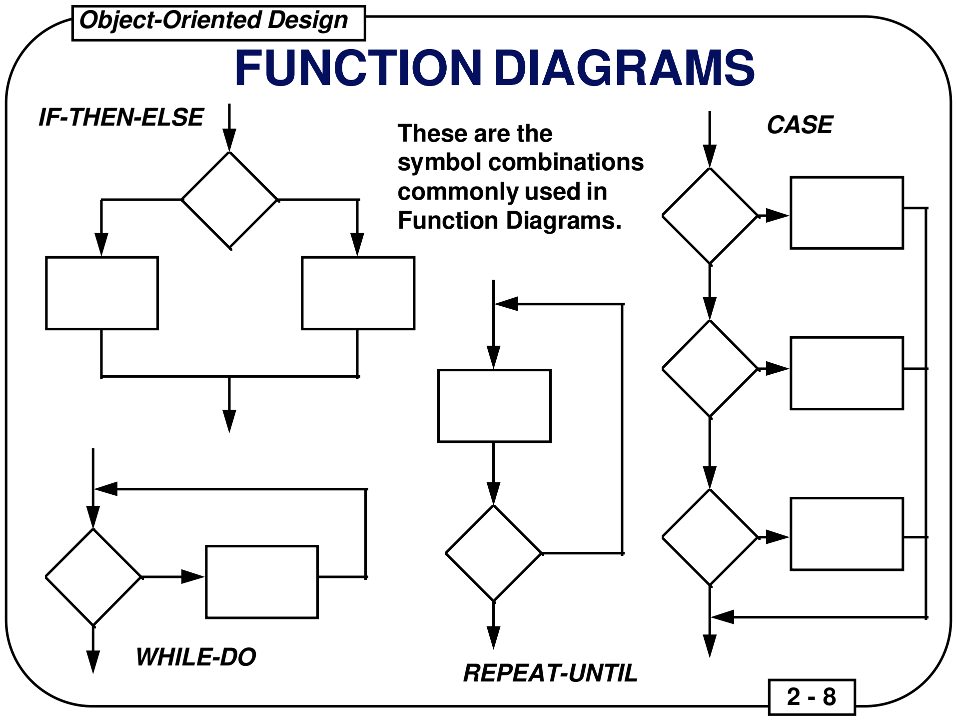
FUNCTIONAL ANALYSIS METHODS

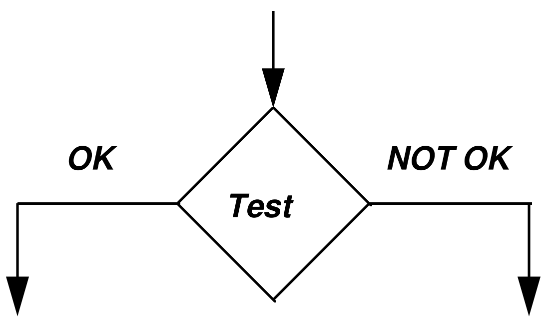
✓ **Function Diagrams tell us:**

- **Functions in the System**
- **Sequence of Function Performance**

✓ **State Transition Diagrams (STD's) tell us:**

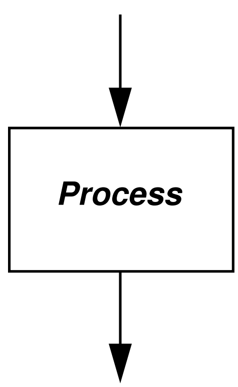
- **States of the System**
- **Relationships between States in the System**
- **Events that Cause State Transitions in the System**
- **Resulting Actions Performed in Response to these Events**





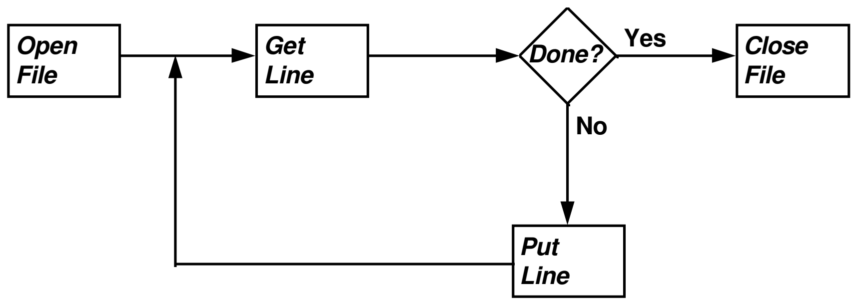
Two common symbols used in Function Diagrams are:

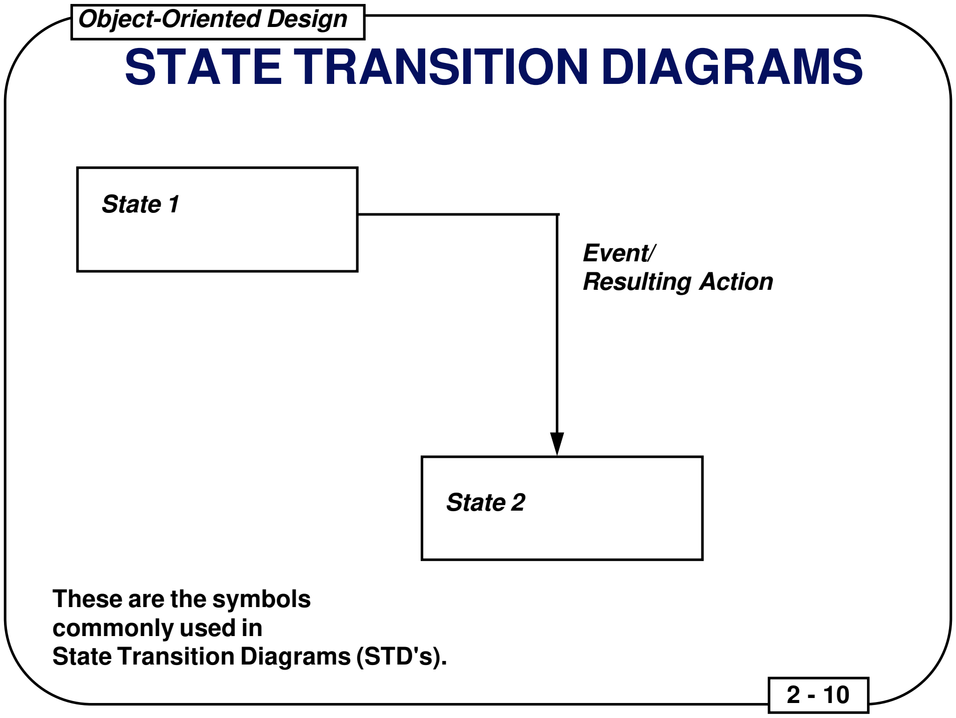
- The decision symbol, which shows a selection of one of two alternate flows based upon a condition being met or not
- The process symbol, which shows a sequence of events being performed

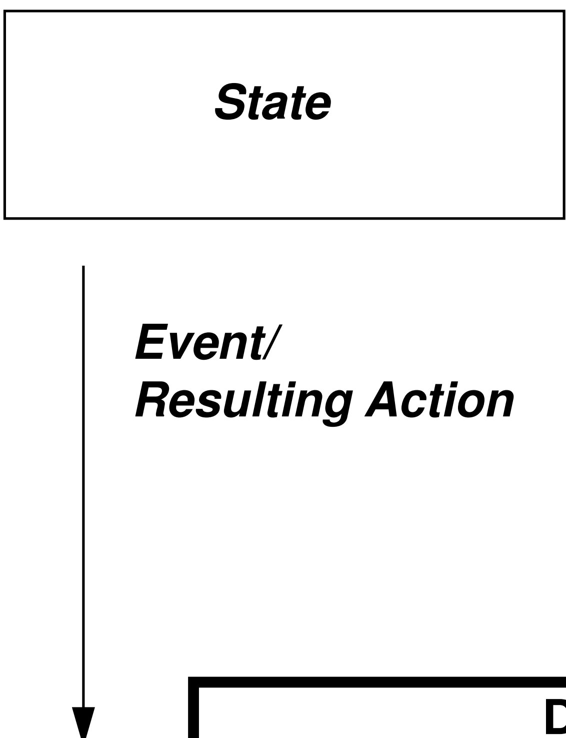


The combinations shown in the transparency are common organizations of these symbols found in structured code.

FUNCTION DIAGRAMS EXAMPLE







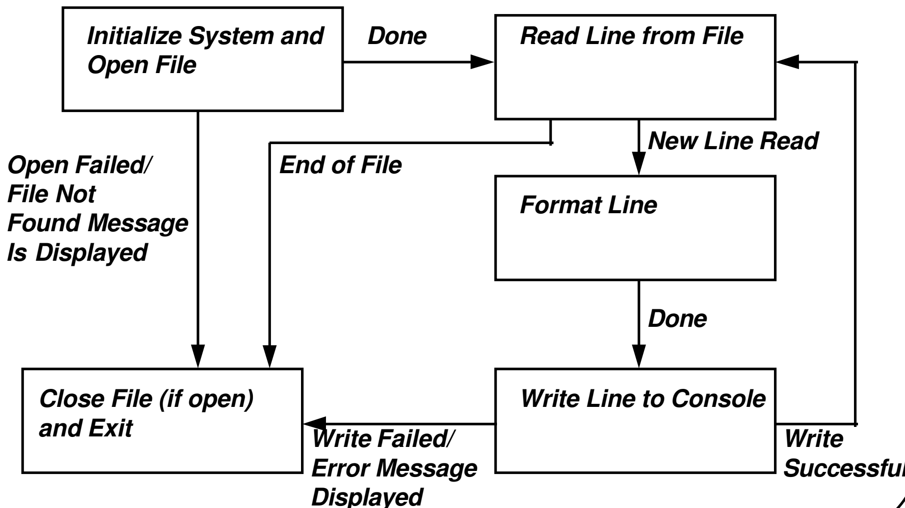
Two common symbols used in STD's are:

- The state symbol, which simply names a state
- The transition symbol, which has text associated with it that names the event which causes the transition and, optionally, an action which results when the event occurs (this action is performed during the transition)

Data Dictionary

The Data Dictionary may play a role in these diagrams, particularly when CASE tools are used, by providing much more information on the details of a state, an event, or a resulting action performed in response to an event.

STATE TRANSITION DIAGRAMS EXAMPLE



OBJECT DIAGRAM CONVENTIONS

✓ **Entity Relationship Diagrams (ERD's) tell us:**

- Entities in the System
- Relationships between these Entities

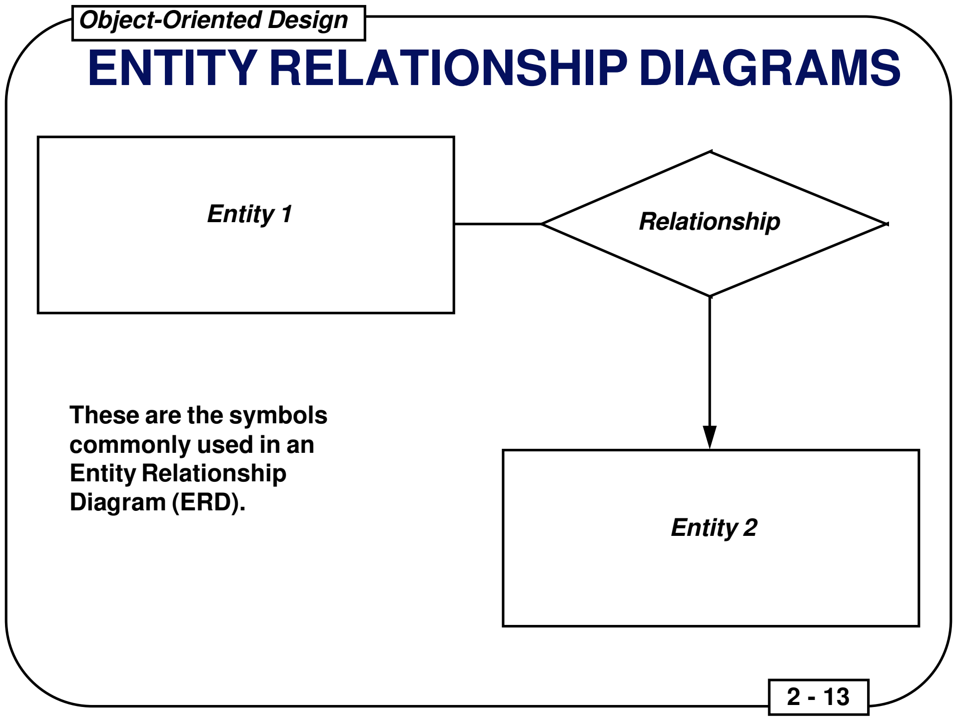
✓ **Object Interaction Diagrams tell us:**

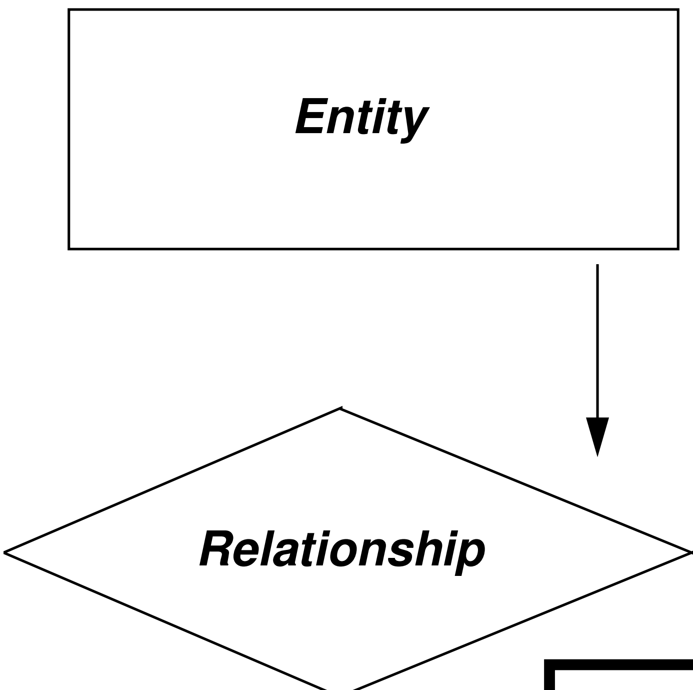
- Objects and Classes in the System
- Relationships between Objects
- Object Interfaces
- Data Flow between Objects
- Method Invocation
- Sequencing of Invocations (optional)

✓ **Booch Diagrams tell us:**

- Dependency Relationships between Classes

- *Diagramming Notations*
- *Data Flow Analysis Methods*
- *Data Flow Diagrams*
- *Data Dictionary and Its Content*
- *Functional Analysis Methods*
- *Function Diagrams*
- *State Transition Diagrams*
- **Object Diagram Conventions**
- **Entity Relationship Diagrams**
- **Object Interaction Diagrams**
- **Booch Diagrams**
- *Design Methodologies*





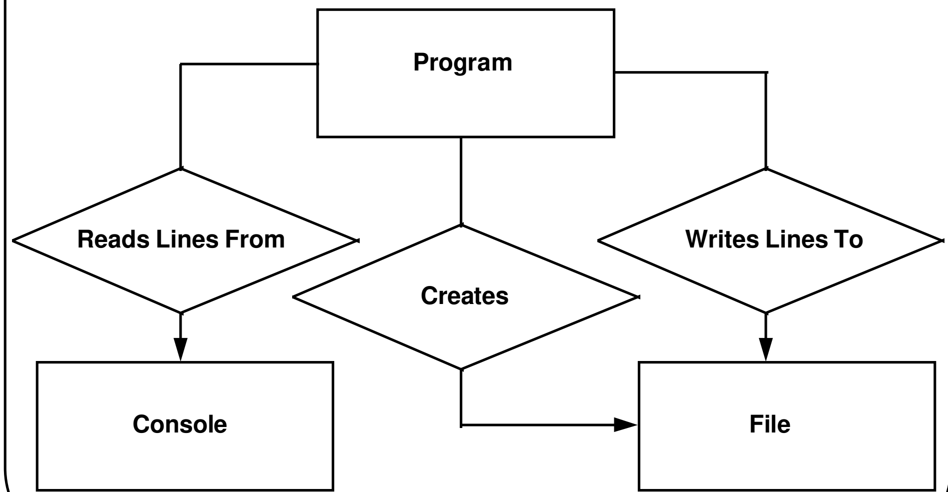
Three common symbols used in ERD's are:

- An entity
- A relation line, showing the direction in which to read the relationship
- A relationship

Data Dictionary

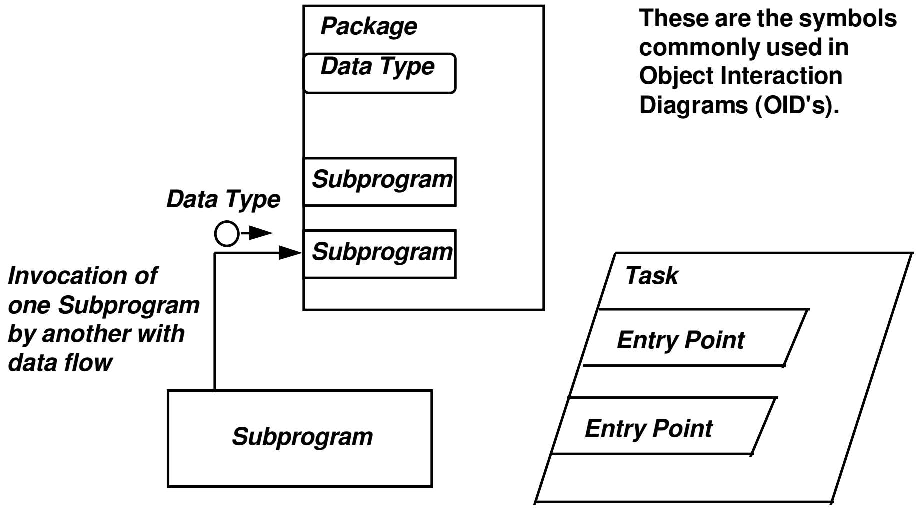
The Data Dictionary may play a role in these diagrams, particularly when CASE tools are used, by providing much more information on the details of a state, an event, or a resulting action performed in response to an event.

ENTITY RELATIONSHIP DIAGRAMS EXAMPLE

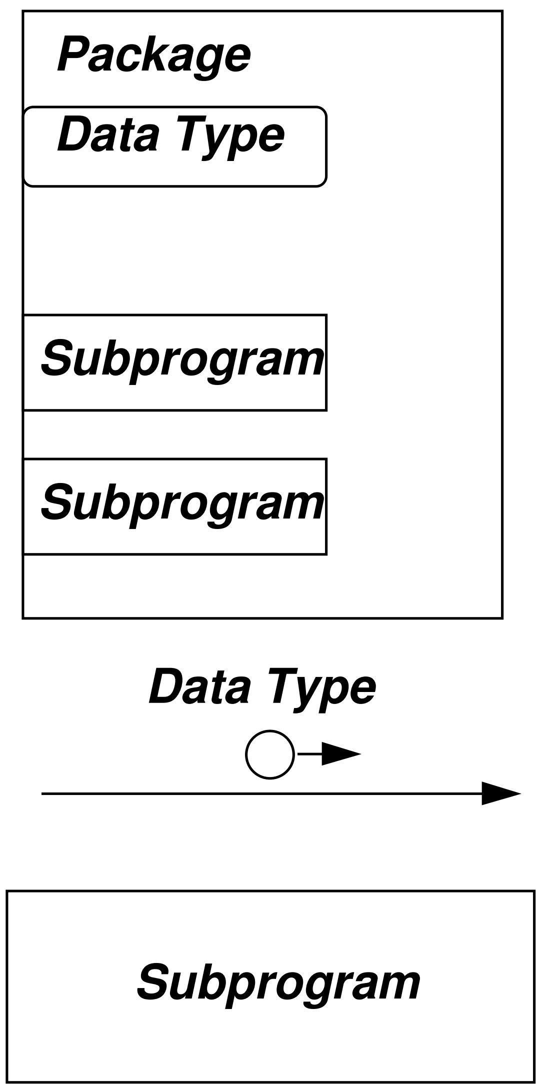


Object-Oriented Design

OBJECT INTERACTION DIAGRAMS

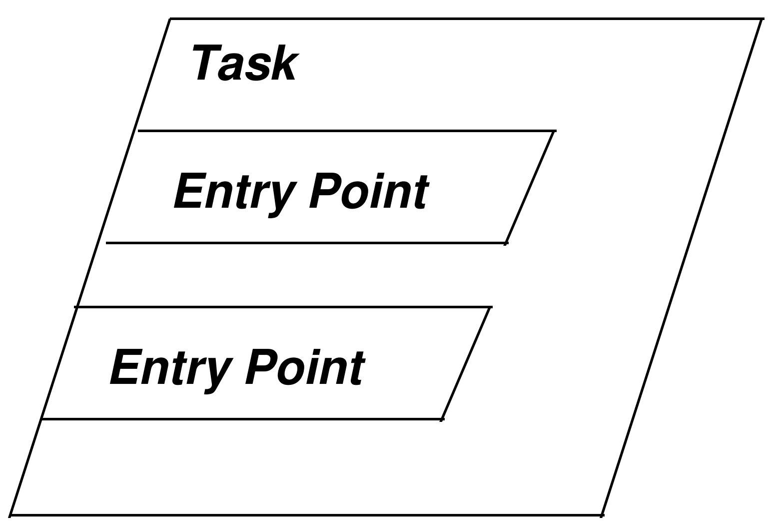


These are the symbols commonly used in Object Interaction Diagrams (OID's).

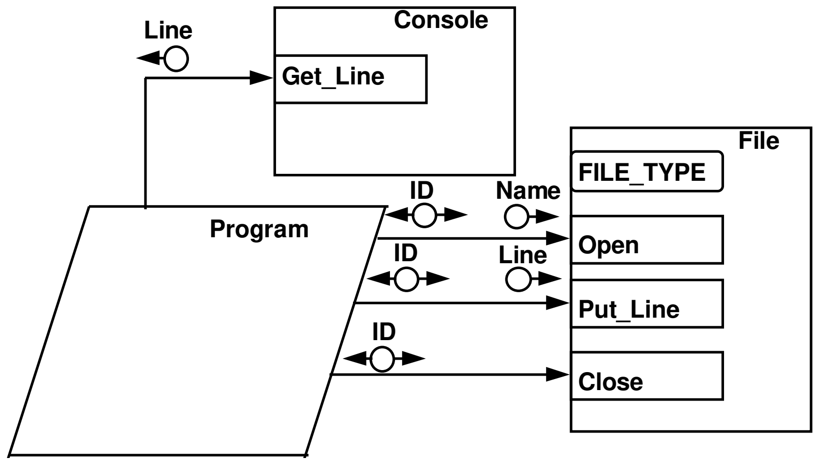


Four common symbols used in OID's are:

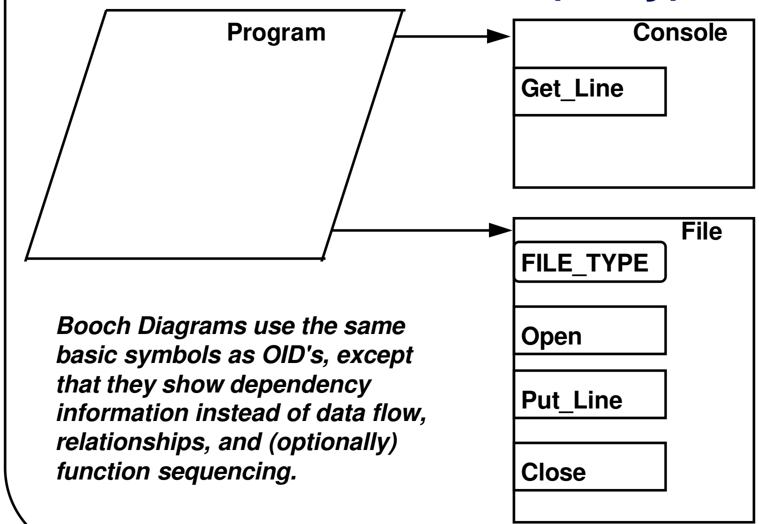
- The package, which usually represents a server, or passive, object
- The task, which usually represents a master, or active, or an agent, or active -passive, object
- The invocation flow with the passed data
- The subprogram, which usually represents a functional interface



OBJECT INTERACTION DIAGRAMS EXAMPLE



BOOCH DIAGRAMS EXAMPLE (Only)



Booch Diagrams use the same basic symbols as OID's, except that they show dependency information instead of data flow, relationships, and (optionally) function sequencing.

DESIGN METHODOLOGIES

- ✓ Data Flow-Oriented Design
- ✓ Data Structure-Oriented Design
- ✓ Object-Oriented Design
- ✓ Real-Time Design

Note

The first three classes are heavily driven by the *Information Domain.*

- *Diagramming Notations*
- *Data Flow Analysis Methods*
- *Data Flow Diagrams*
- *Data Dictionary and Its Content*
- *Functional Analysis Methods*
- *Function Diagrams*
- *State Transition Diagrams*
- *Object Diagram Conventions*
- *Entity Relationship Diagrams*
- *Object Interaction Diagrams*
- *Booch Diagrams*
- **Design Methodologies**

Data Flow-Oriented Design

- Uses information flow characteristics to derive the program structure
- There are two design analysis techniques:
  - ❍ *Transform Analysis and Design* - the information flow exhibits distinct boundaries between incoming and outgoing data (i.e., input, processing, and output are the three key elements of the data flow)
  - ❍ *Transaction Analysis and Design* - an information item causes the flow to branch along a choice of paths
- Data Flow Diagrams (DFD's) are the common graphical means to represent the flow of data

Data Flow-Oriented Design
Transform Analysis and Design

Design Steps:

- Review the fundamental system model
- Review and refine the DFD's for the software
- Determine the transform and transaction characteristics of the DFD's
- Isolate the transform center by specifying incoming and outgoing flows
- Perform "first-level factoring" - derive the mapping from the major parts of the DFD to a program structure
- Perform "second-level factoring" - map individual bubbles in the DFD into modules in the program structure
- Refine the above "first-cut" program structure - maximize cohesion, minimize coupling, and build a structure hierarchy

Data Flow-Oriented Design
Transaction Analysis and Design

Design Steps:

- **Review the fundamental system model**
- **Review and refine the DFD's for the software**
- **Determine the transform and transaction characteristics of the DFD's**
- **Isolate the transaction center and the flow characteristics of each action path**
- **Map the DFD into a software structure amenable to transaction processing**
- **Factor and refine the transaction structure and the structure of each action path**
- **Refine the above "first-cut" program structure - maximize cohesion, minimize coupling, and build a structure hierarchy**

Data Flow-Oriented Design
Design Heuristics

- Minimize coupling and maximize cohesion
- Minimize fan-out and strive for fan-in as the depth increases
- Minimize side-effects; keep the scope of the effect of a module within the scope of control of that module
- Evaluate module interfaces to reduce complexity and redundancy; improve consistency of the module
- Define modules whose function is predictable and testable
- Strive for single-entry, single-exit modules
- Package softwawre based on design constraints and portability requirements

Data Structure-Oriented Design

- **Three key methods:**
  - ***Jackson System Development*** **- concentrates on process modeling and control**
  - ***Logical Construction of Programs (Warnier)*** **- rigorous view of data structure and focus on detailed procedural design**
  - ***Data Structured System Development (Orr)*** **- incorporates data flow analysis with the Logical Construction of Programs and Jackson System Development (JSD to a lesser extent)**
- **This is 1970's technology and is not covered in detail**

Object-Oriented Design (OOD)

- Concerns itself with creating a model of the real world
- Objects represent the information domain, and the operations associated with that information are grouped with the objects
- Messages (interfaces) provide a means by which operations are invoked
- Packaging of objects with their associated operations takes place - data and procedural abstractions are combined in a single program component called an *object* or a *package*
- OOD representations are more prone than others to programming language dependency

Object-Oriented Design Definitions

- ***Object*** **- a component of the real world that is mapped into the software domain or an information item**
- ***Operations*** **or** ***Methods*** **- processes which act on objects to transform their internal data structure or provide information on their internal data structures**
- ***Message*** **- a request to an object to perform one of its operations**
- ***Class*** **- a set of objects which share common characteristics**
- ***Instance*** **- an individual object of a class**

Object-Oriented Design Steps

- **Identify the objects**
- **Identfy the attributes of the objects**
- **Identify the operations that may be applied to the objects**
- **Establish the interfaces of the objects to the outside world (Ada package specifications may be used if Ada is the implementation language)**
- **Implement the objects (Ada package bodies may be used if Ada is the implementation language)**
- **Graphical representation may be employed; Booch Diagrams and Object Interaction Diagrams are the recommended diagramming notations**

Real-Time Design

- Encompasses all aspects of conventional software design while simultaneously introducing timing and sizing constraints; these constraints must be satisfied by the code
- All classes of design (architectural, procedural, and data) become more complex due to the response time required by the real-world constraints
- Mathematical modeling and simulation are common tools used for real-time design

Real-Time System Concerns

- Interrupt handling and context switching
- Response time
- Data transfer rate
- CPU and system throughput
- Resource allocation and priority handling
- Task synchronization and intertask communication