SOFTWARE REQUIREMENTS

- **Fundamentals**
  - Overview
  - Analysis Principles
- **Structured Analysis**
  - Notation
  - Extensions for Real Time
  - Mechanics
  - Requirements Dictionary
- **Object-oriented Analysis**
  - Basic Concepts
  - OO Analysis Modeling
  - OO Data Modeling
- **Formal Techniques**
  - Background
  - The Z Spec Language
- **Automated Techniques**

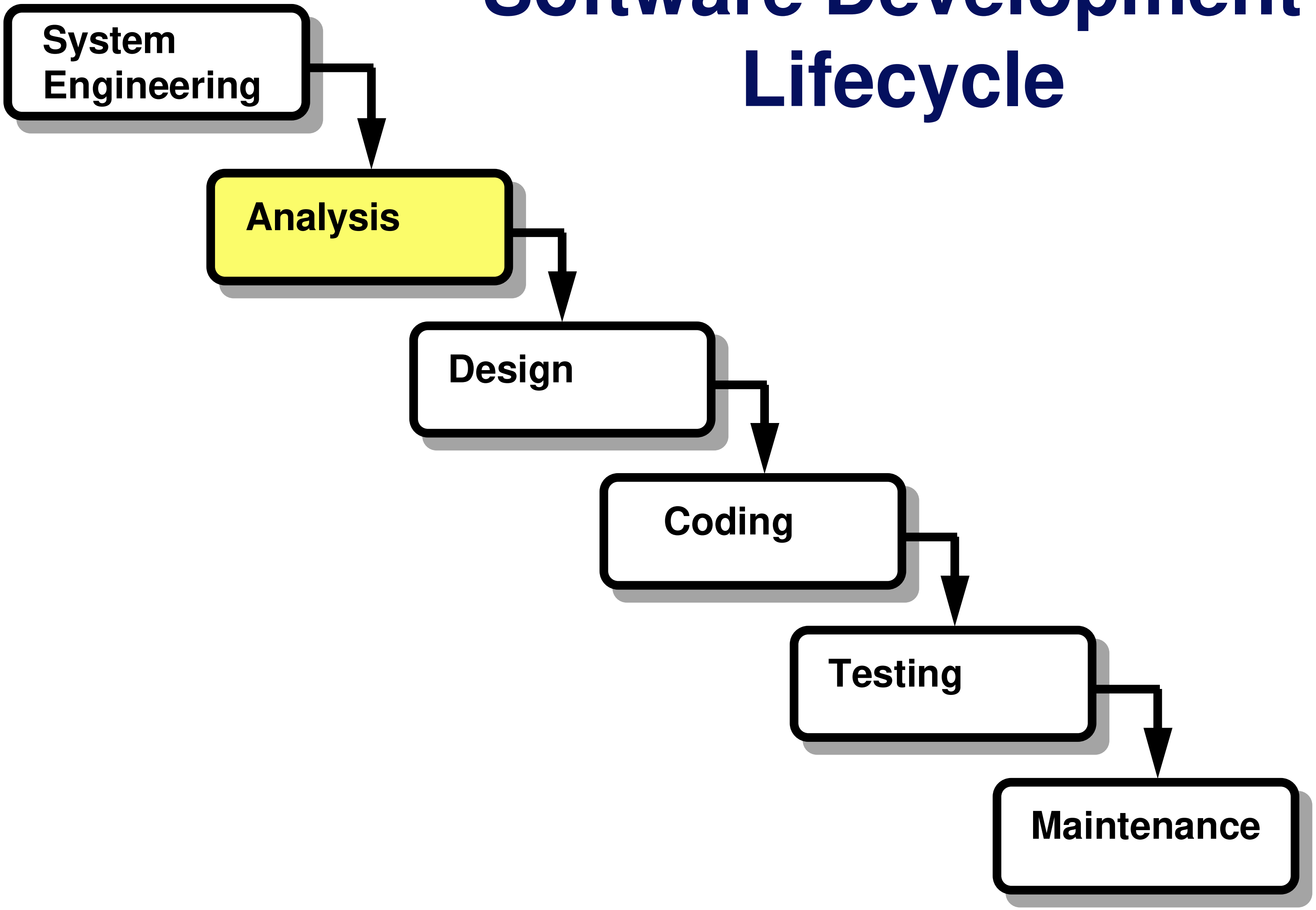
TOPICS

Fundamentals

Structured and Object-Oriented Analysis

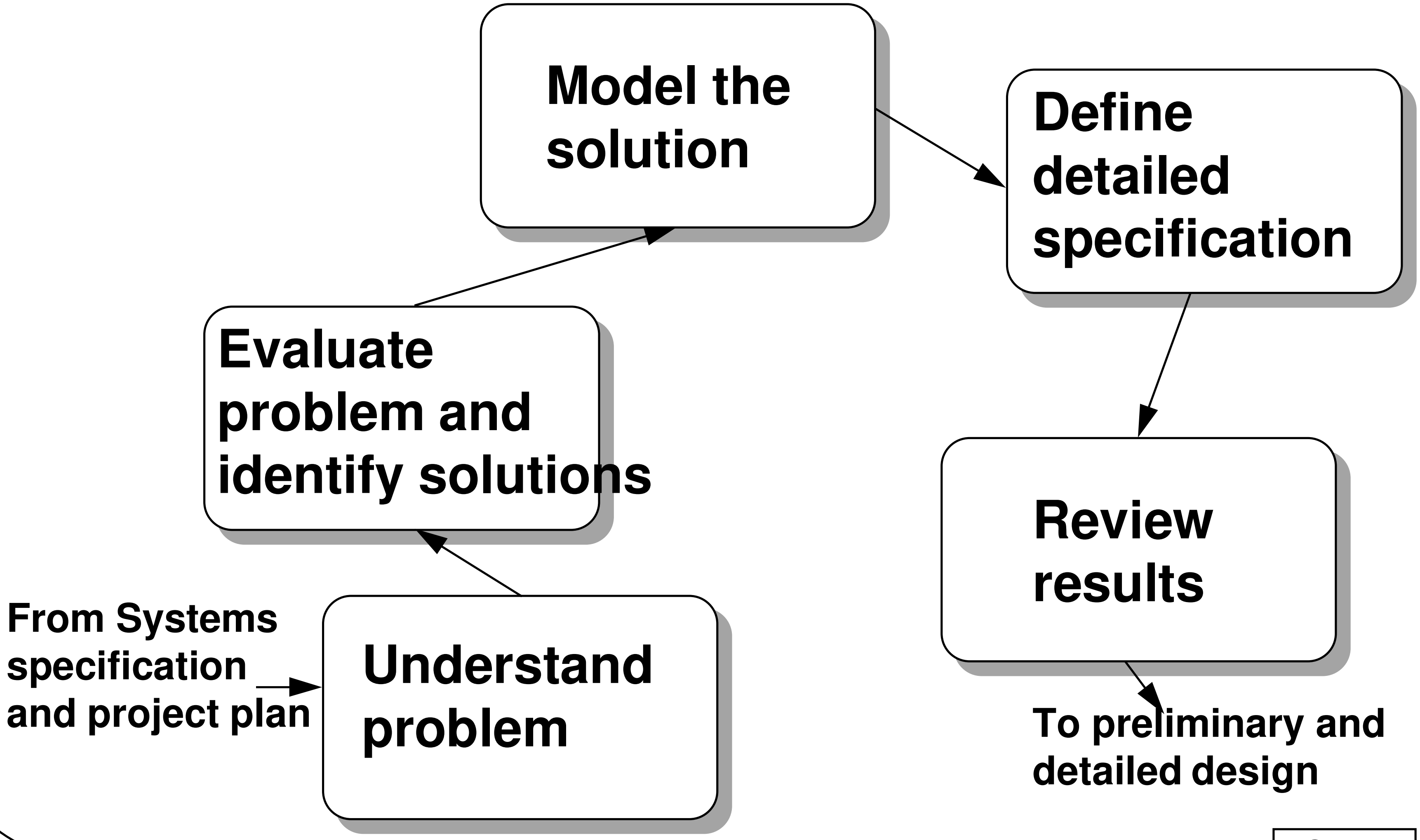
Formal and Automated Techniques

Software Development Lifecycle



Requirements Analysis - Overview

Tasks



From Systems specification and project plan

Understand problem

Evaluate problem and identify solutions

Model the solution

Define detailed specification

Review results

To preliminary and detailed design

Basic Activities of Software Requirements Analysis

- Define the functional domain - what functions are to be performed?
- Define the information domain - what is the flow of information in the system, what is the structure of that information, and what is the content of that information?
- Partition the problem - what is the hierarchy of the problem?
- Develop the logical view of the requirements - detail the functions and data
- Develop the physical view of the requirements - detail the real-world forms of the functions and data

Common Problems Encountered During Requirements Analysis

- **general communications problems, including not understanding the problem, misinterpreting information, and missing information**
- **acquiring pertinent information**
- **handling problem complexity**
- **accommodating changes that will occur during and after analysis**

Beginning the Process

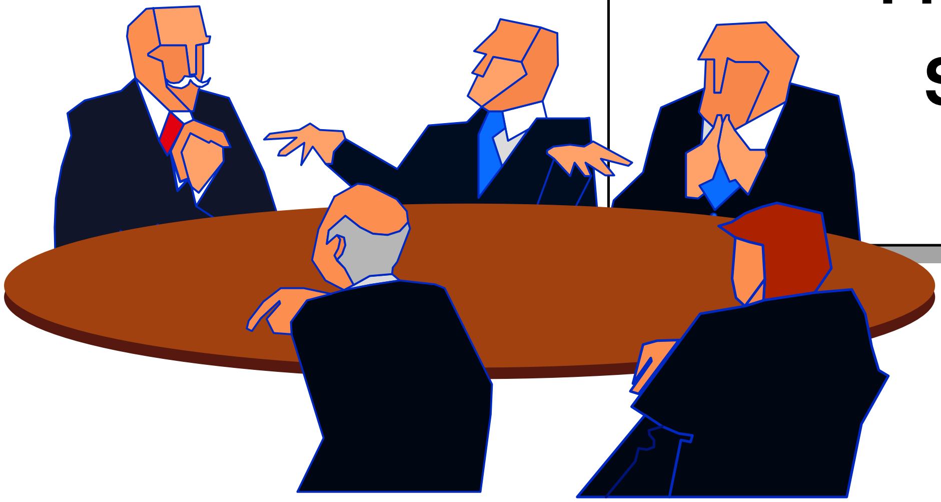
Hold a meeting!

The Facilitated

Application

Specification

Technique (FAST)



Example: The SafeHome System

A microprocessor-based home security system that protects against a number of undesireable events such as illegal entry, fire, flood, etc.

SafeHome will use sensors to detect each situation, can be programmed by the homeowner.

SafeHome will automatically telephone a monitoring agency when a situation is detected.

Problem Understanding

Step 1. Identify objects, operations, constraints, and performance criteria:

Objects

Smoke detectors

Window/door sensors

Motion detectors

Alarm

Control panel

Telephone numbers

Operations

Set/reset alarm

Monitor sensors

Dial phone

Program control panel

Constraints

Cost less than $200

Easy to use

Direct dial to telephone

Performance Criteria

Display within 1 s of event

Prioritize event processing

Delay at least 1 min before dialing phone

Problem Understanding, Continued

Step 2. Develop "mini"-specification for each entry on each list

Object: Control Panel

Mounted on wall

Size 9x5 inches

Contains 12 key-pad and special keys

Diagram of panel

All user interaction through control panel

Used to enable and disable system

Software to provide interaction guidance, echo responses, etc.

Connected to all sensors

Problem Understanding, Continued

Step 3. After much debate and list modifications, create list of validation criteria

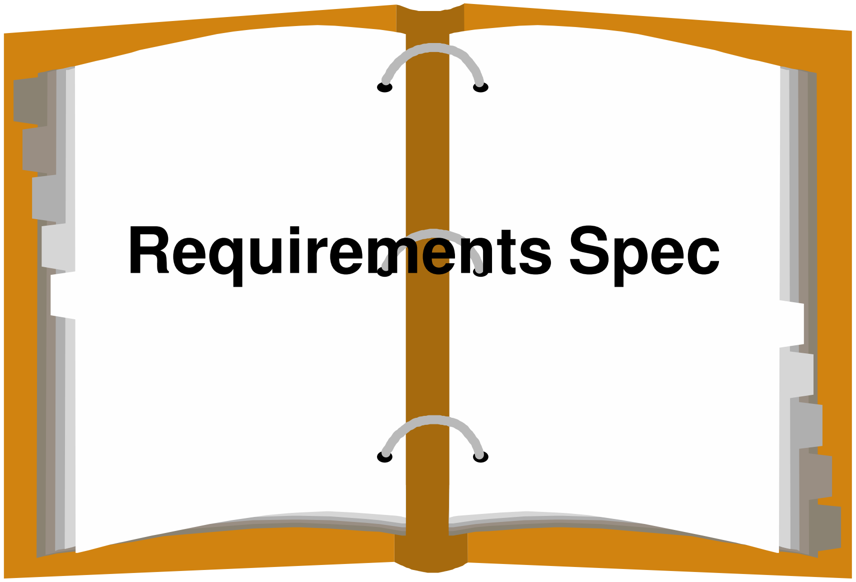
Enter 200 random events and observe alarm responses

Ensure display resets on power up

When phone numbers are entered with 555- prefix, ensure telephone is *not* dialed

Problem Definition

Step 4. Write complete draft specification using results of steps 1-3



Concepts of Analysis

Information Domain:

1. Information flow
2. Information content
3. Information structure

Modeling: Pictorial representation of problem solution

Aids analyst in understanding problem

Focal point of review

Foundation for design

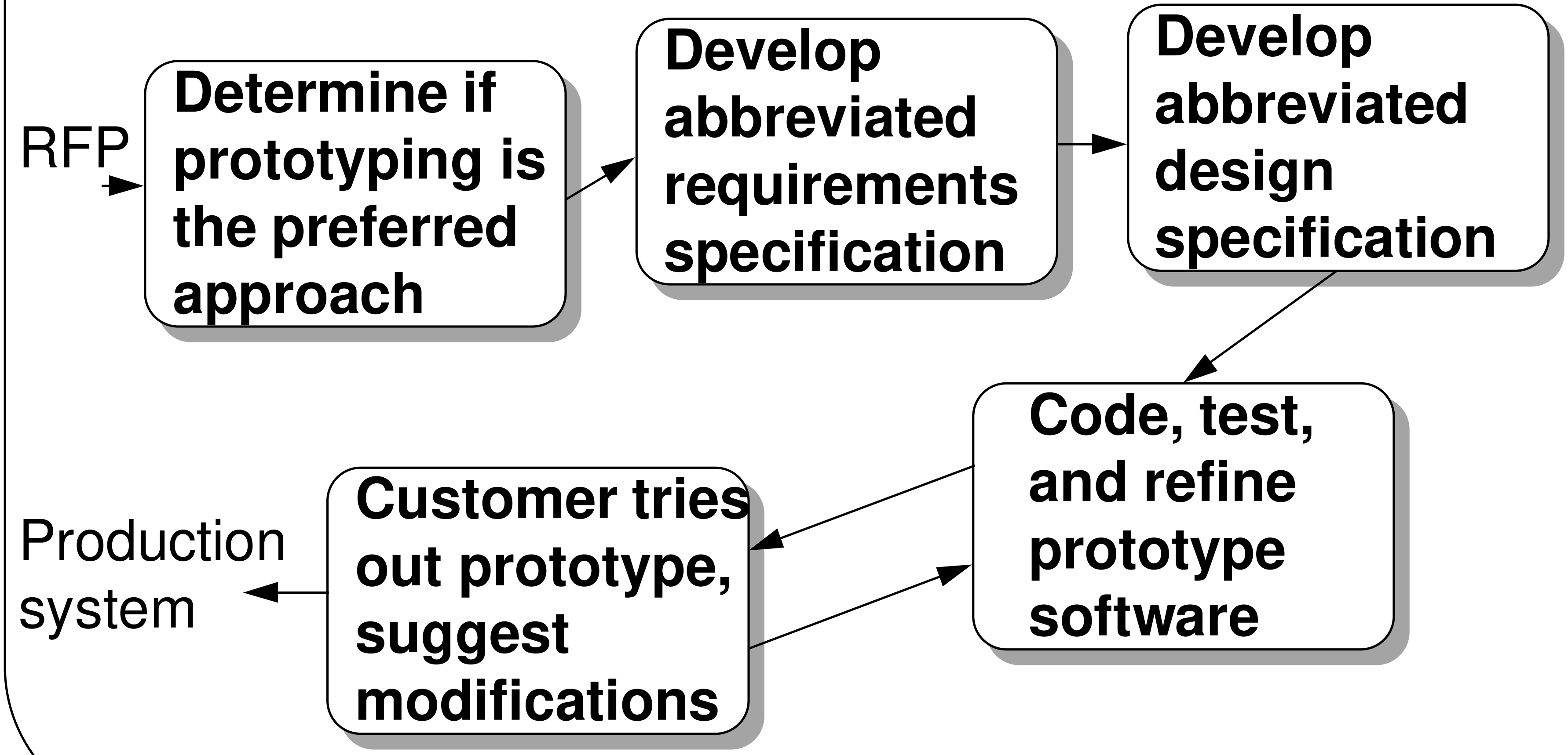
Partitioning: Break big problems into little ones

Software Views

<u>View</u>	<u>Focus</u>
Informational	**Data**
Functional	**Functions**
Behavioral	**Execution process**

Software Prototyping

Assume a request for proposal (RFP) or system spec defines the problem.



RFP

Determine if prototyping is the preferred approach

Develop abbreviated requirements specification

Develop abbreviated design specification

Code, test, and refine prototype software

Customer tries out prototype, suggest modifications

Production system

Specification Principles

- Separate functionality from implementation - describe what is desired, not how
- Understand the system of which the software is a part and the environment in which the system resides
- Develop a cognitive model rather than a design or implementation model, and keep the perspective of the user
- View the specification as a model, see if it is adequate to determine if a proposed implementation is satisfactory, and tolerate imcompleteness
- Localize and loosely couple the specification

Software Requirements Analysis (SRA)

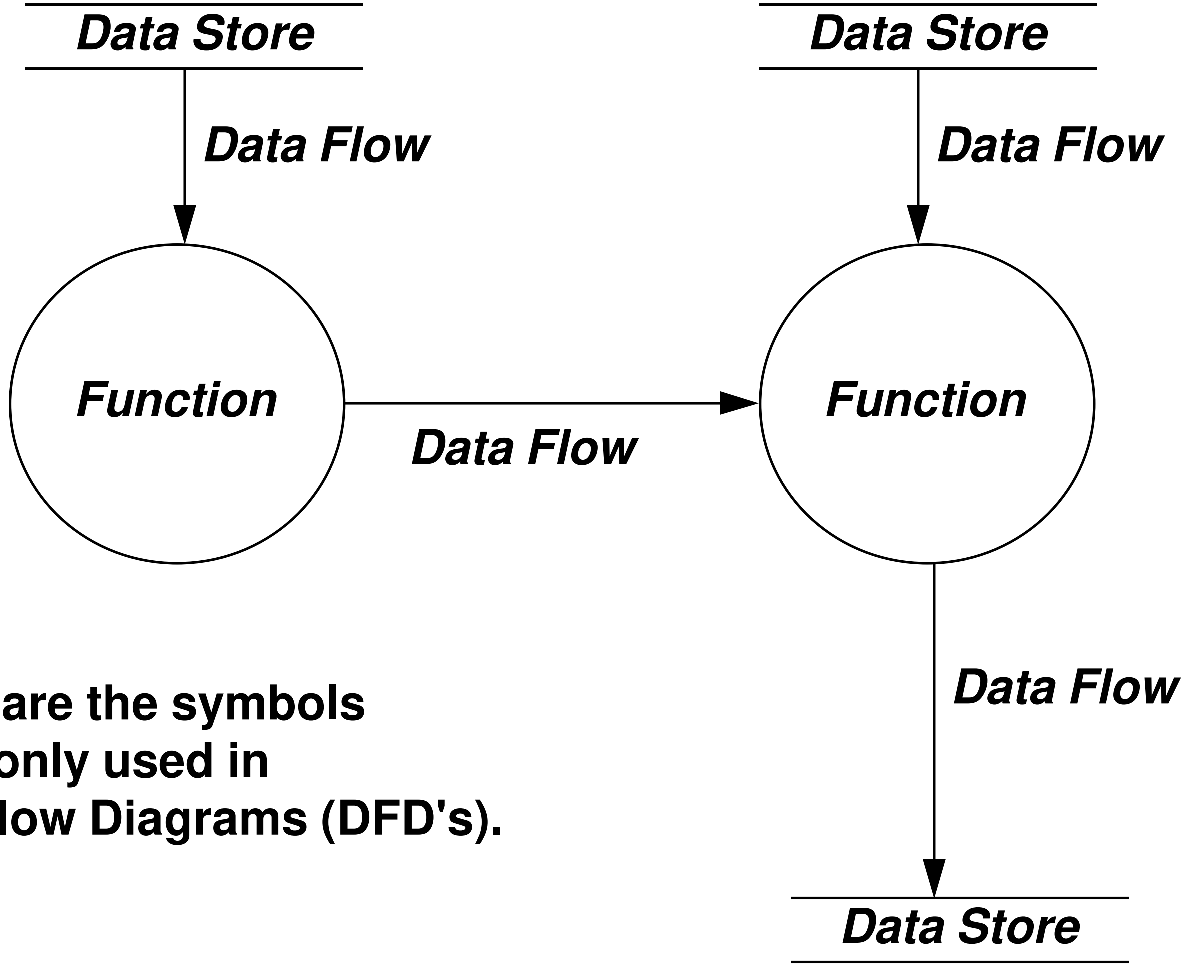
Common Characteristics of the Methodologies

- They perform information domain analysis
- They have a means to represent functions
- They can define interfaces
- They support partitioning of the problem
- They support abstraction
- They can represent both the physical and logical views of the problem

Data Flow Analysis Methods

- **Data Flow Diagrams**
- **Data Dictionary**

Data Flow Diagrams



These are the symbols
commonly used in
Data Flow Diagrams (DFD's).

DFD Example

Simple Printed Circuit Board Layout Program

Given two data files: a list of nets and initial board description,

1. Determine and display the best route for interconnecting each net on the board.

2. Permit user to:

   a. add new nets to list

   b. delete nets from list

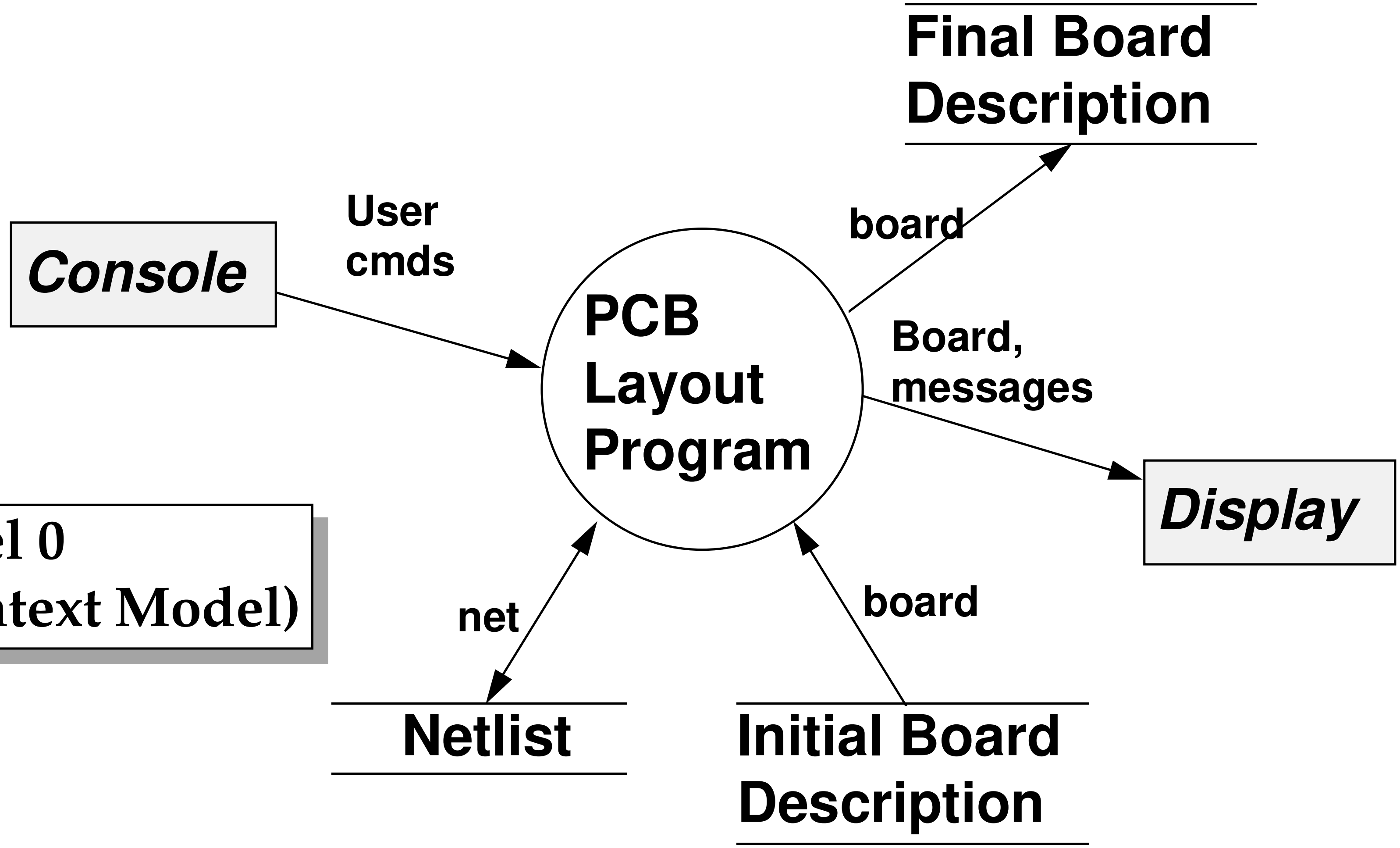
   c. select any or all nets to be routed

   d. request status info about nets or routed board

   e. define style of routing (steiner points, chain, or tree)

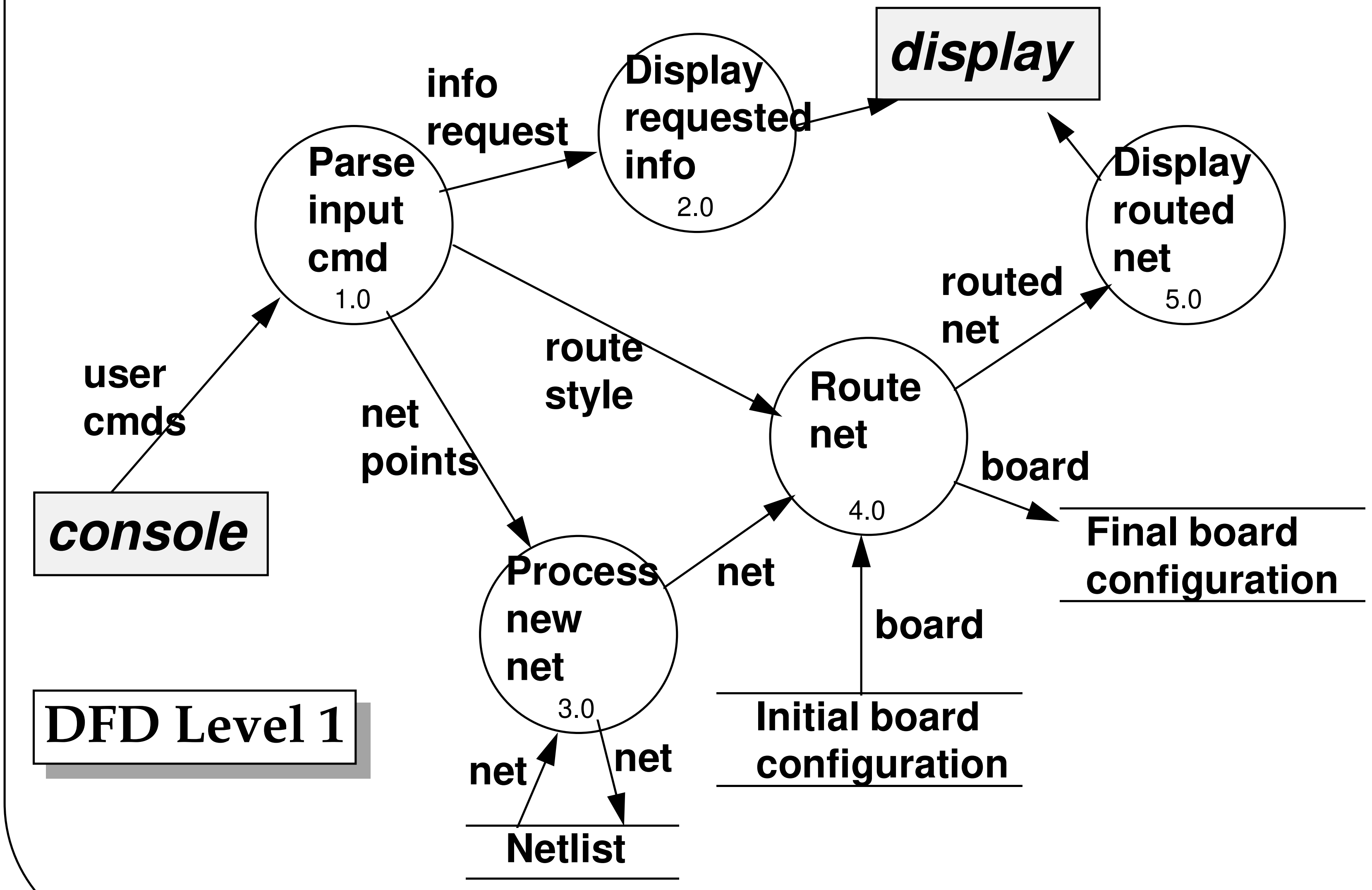
   f. save final routed board in a file

DFD Example, Continued



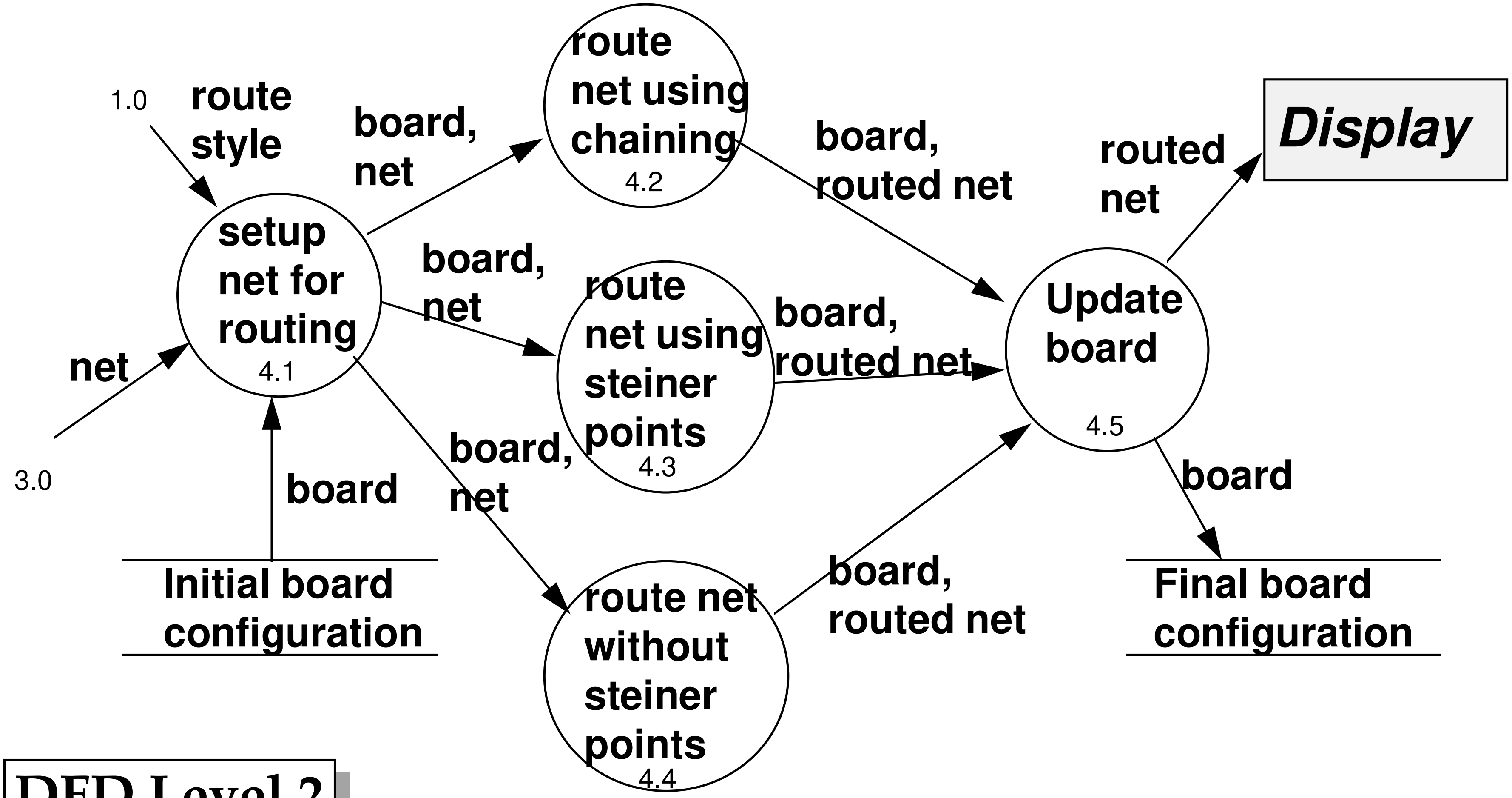
Level 0
(Context Model)

DFD Example, Continued



DFD Level 1

DFD Example, Continued



DFD Level 2

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

Data Dictionary Entry (Example)

Name: **net**

Alias: **net_graph, point_list**

Used:

process	in	out	file	buffer	external
4.1	3.0	4.2,4.3,4.4			
4.2	4.1				
4.3	4.1				
4.4	4.1				

Description: **List of no more than 20 points (x,y) where x and y are vertical and horizontal grid locations on the board. x and y are 16-bit unsigned integer values each greater than 0 and less than the max size of the board.**

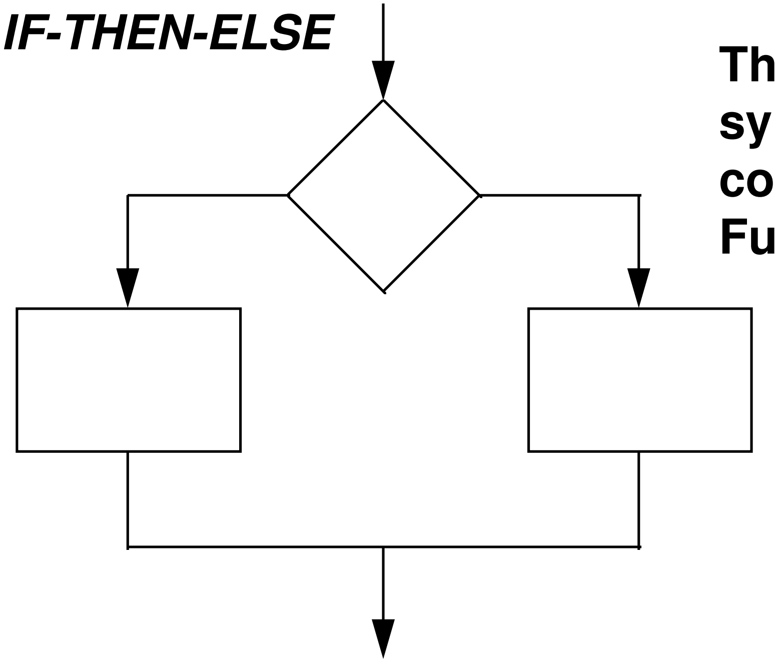
Supplementary Information: **-- none --**

Functional Analysis Methods

- Function Diagrams
- State Transition Diagrams (STD's)
- Entity-Relationship Diagrams (ERD's)

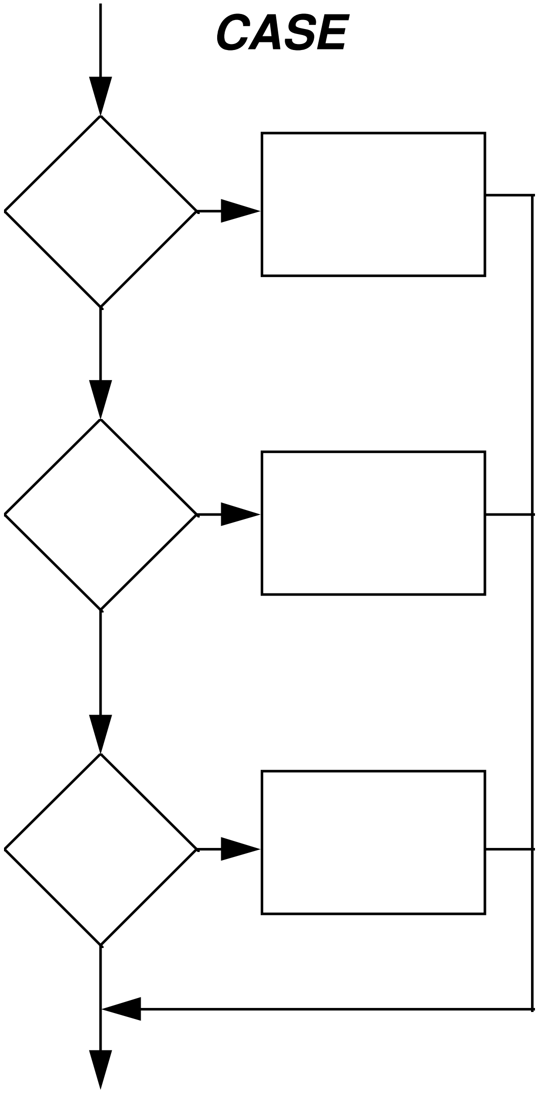
Function Diagrams

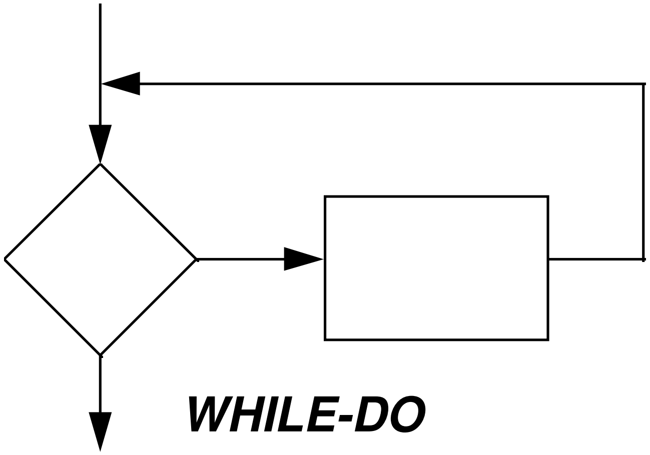
IF-THEN-ELSE



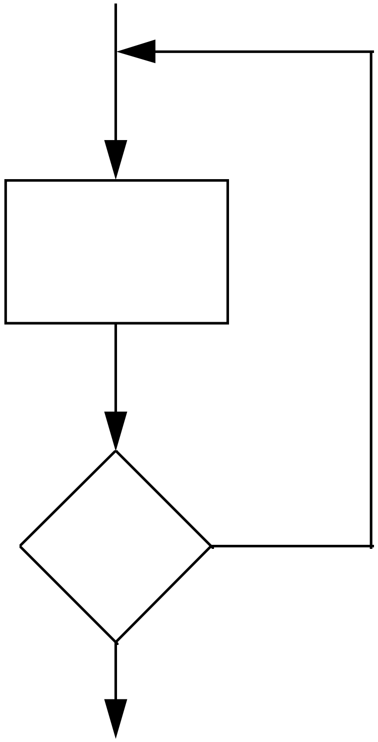
These are the symbol combinations commonly used in Function Diagrams.

CASE



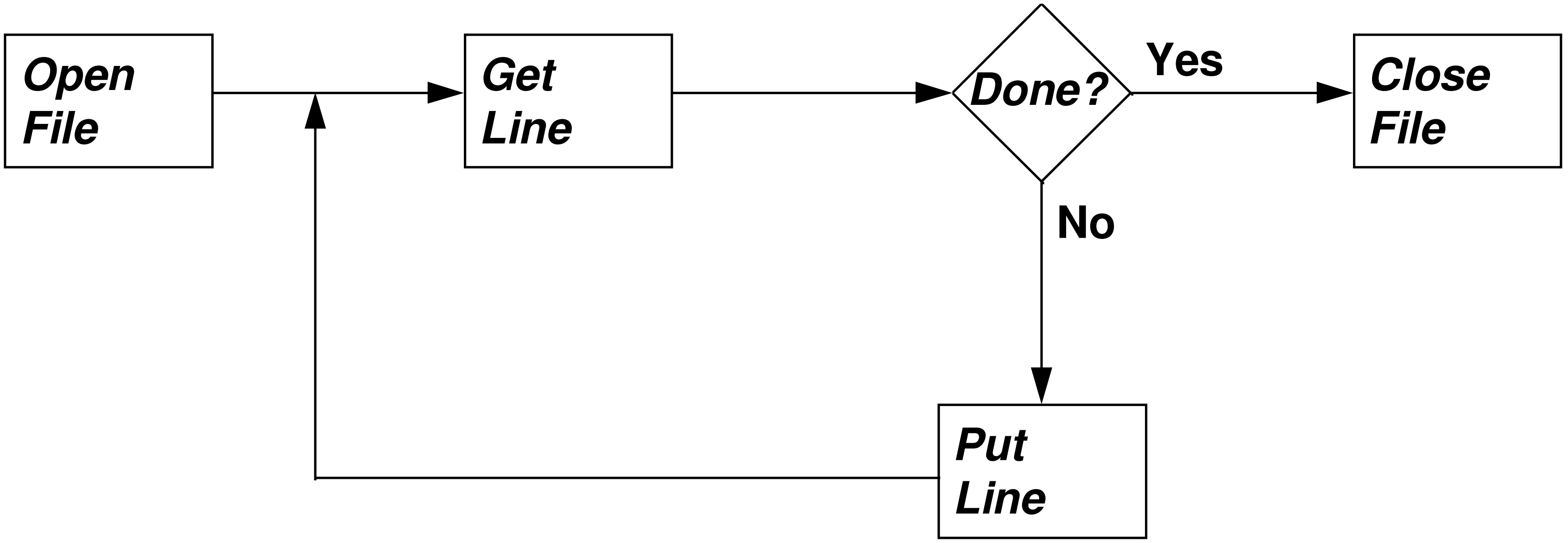


WHILE-DO



REPEAT-UNTIL

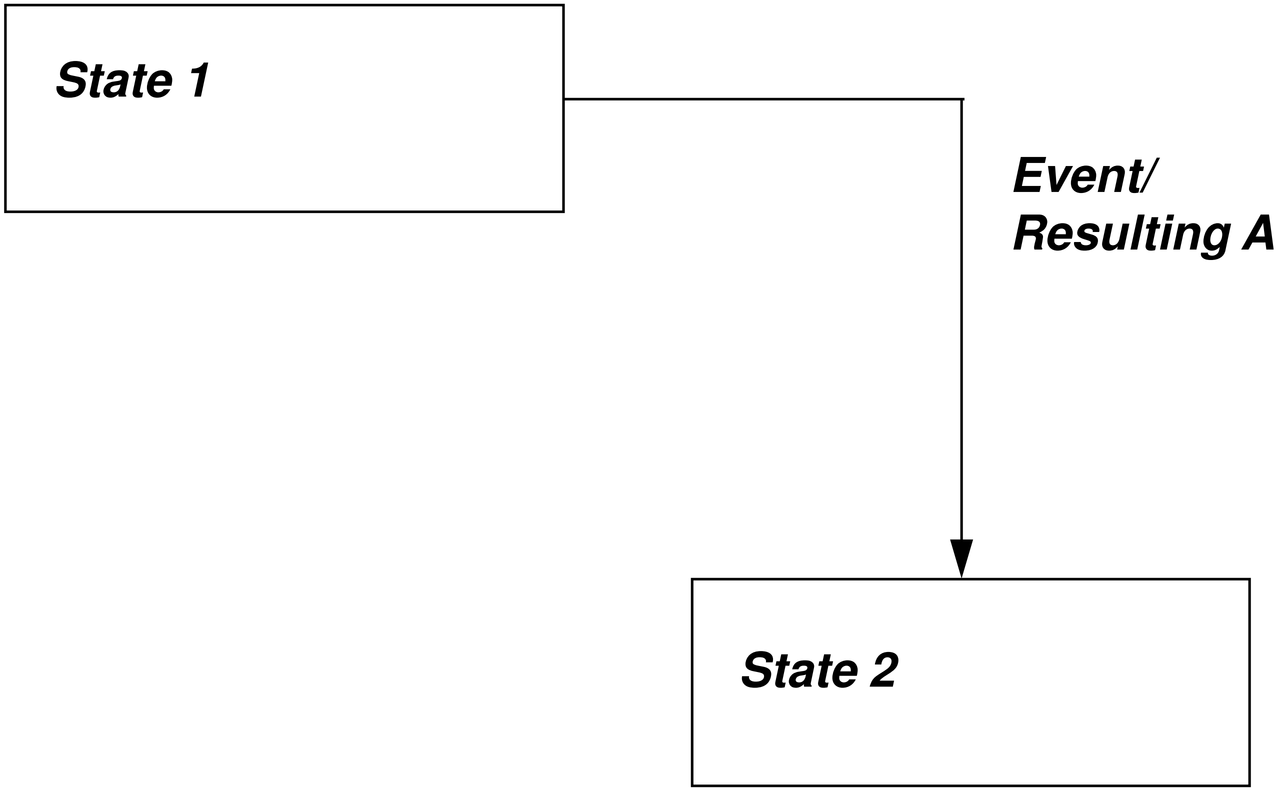
Function Diagrams - Example



Behavioral Modeling

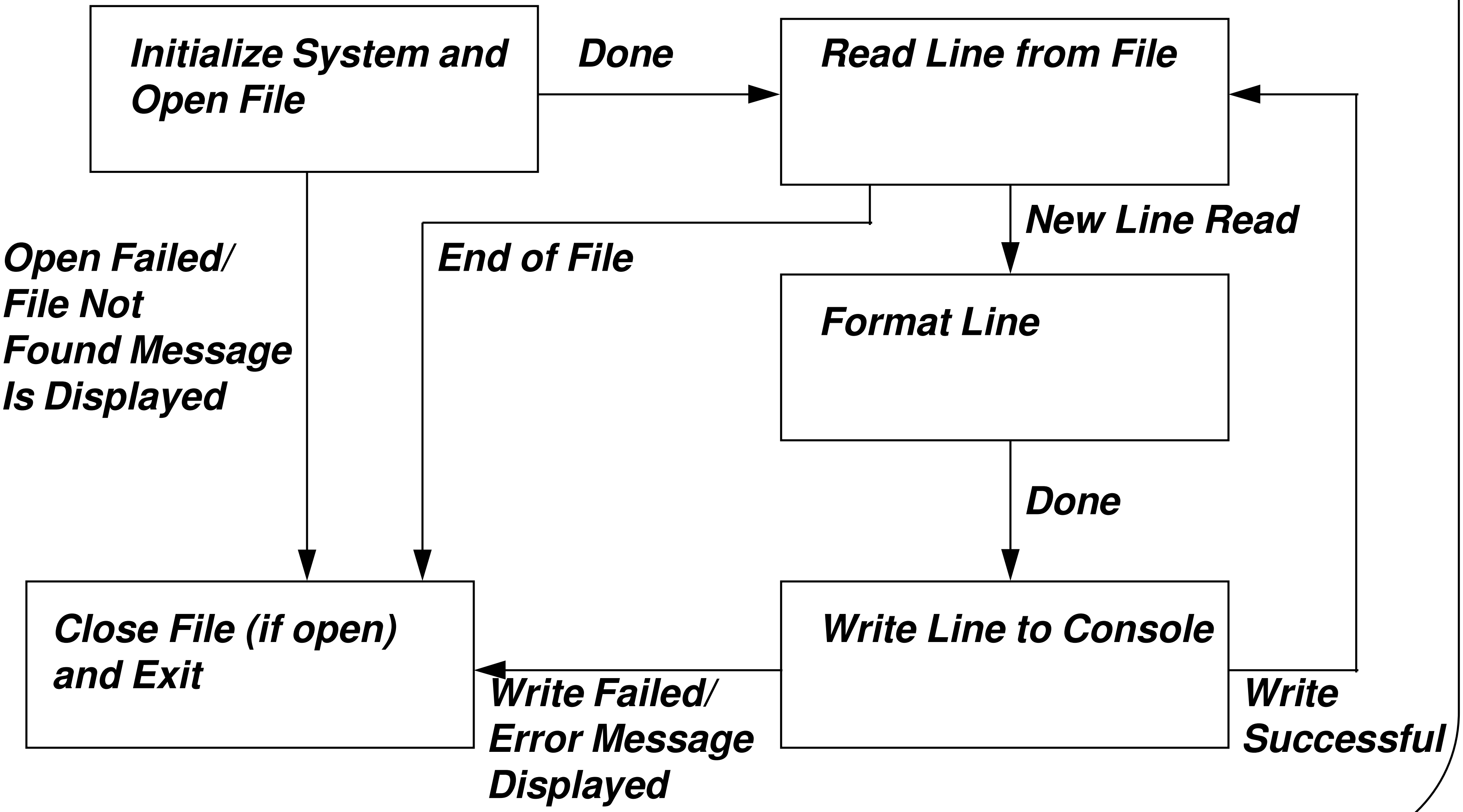
- **Helpful for control-dominated systems**
- **State Transition Diagrams**
  - **Like Finite State machines**
  - **Depicts states and events causing change of state**
  - **Depicts actions to be taken when events received**

State Transition Diagrams



These are the symbols commonly used in State Transition Diagrams (STD's).

State Transition Diagrams - Example



Initialize System and Open File

Done

Read Line from File

Open Failed/ File Not Found Message Is Displayed

End of File

New Line Read

Format Line

Done

Close File (if open) and Exit

Write Line to Console

Write Failed/ Error Message Displayed

Write Successful

SRA for Real Time Systems

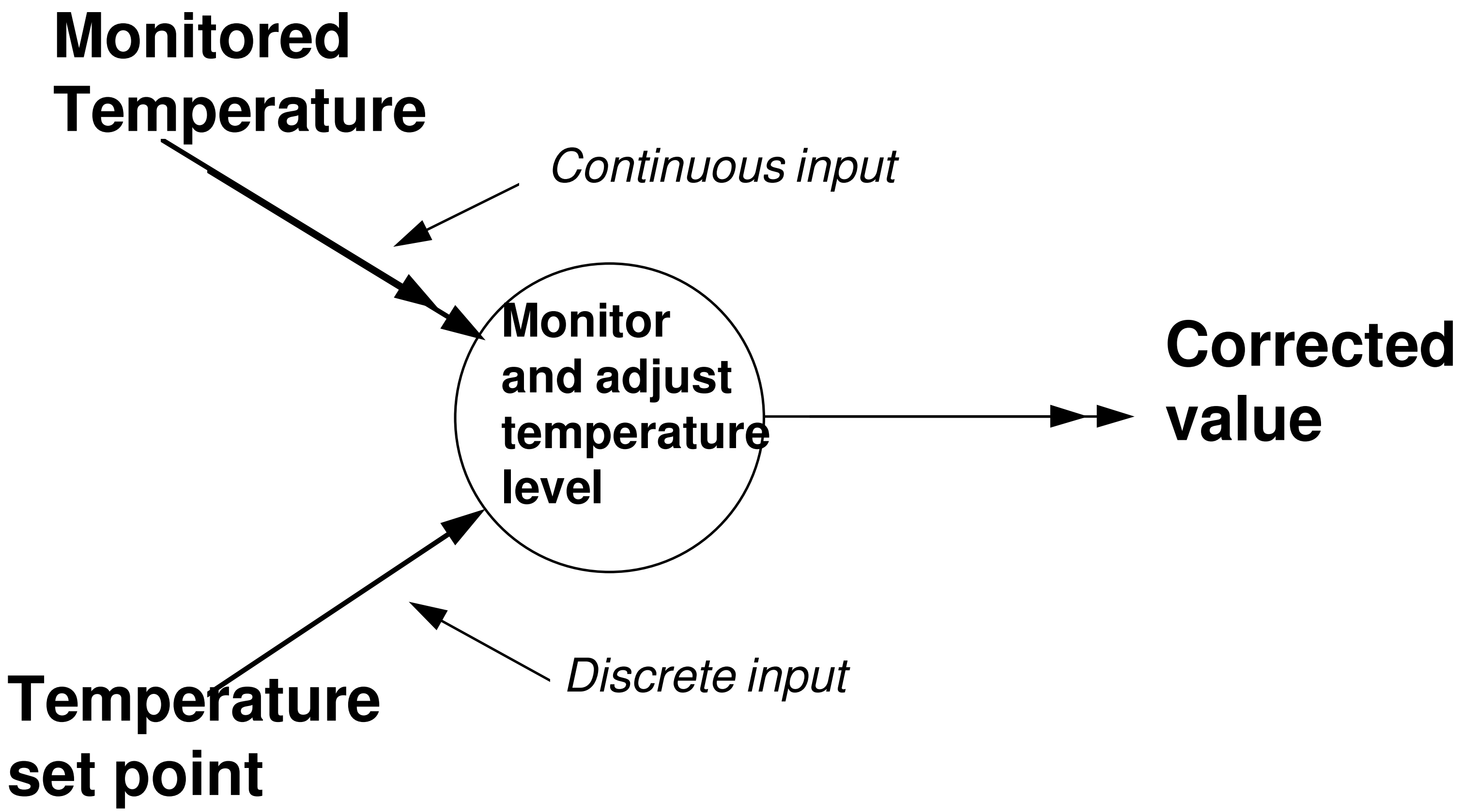
- ***Real Time Systems:***

  1. Time dependent
  2. Control oriented
  3. Driven by events more than data
  4. Some activities execute asynchronously

- Use control flow models to specify such systems
- Approach: Extend DFD model

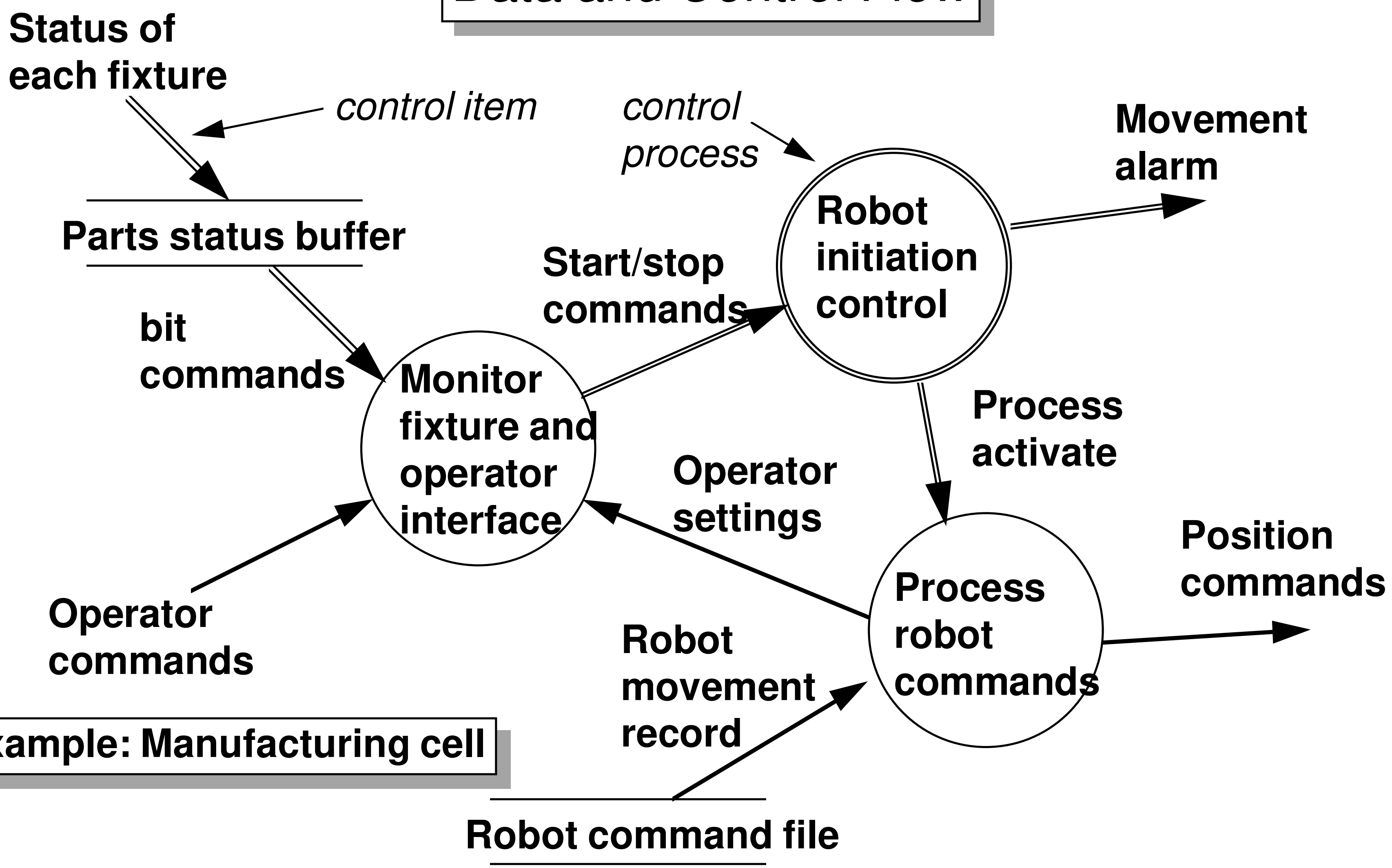
Ward-Mellor Extensions

Time Continuous Data Flows



Ward-Mellor Extensions, Continued

Data and Control Flow



Status of each fixture

control item

control process

Movement alarm

Parts status buffer

Start/stop commands

Robot initiation control

bit commands

Monitor fixture and operator interface

Process activate

Operator settings

Position commands

Process robot commands

Operator commands

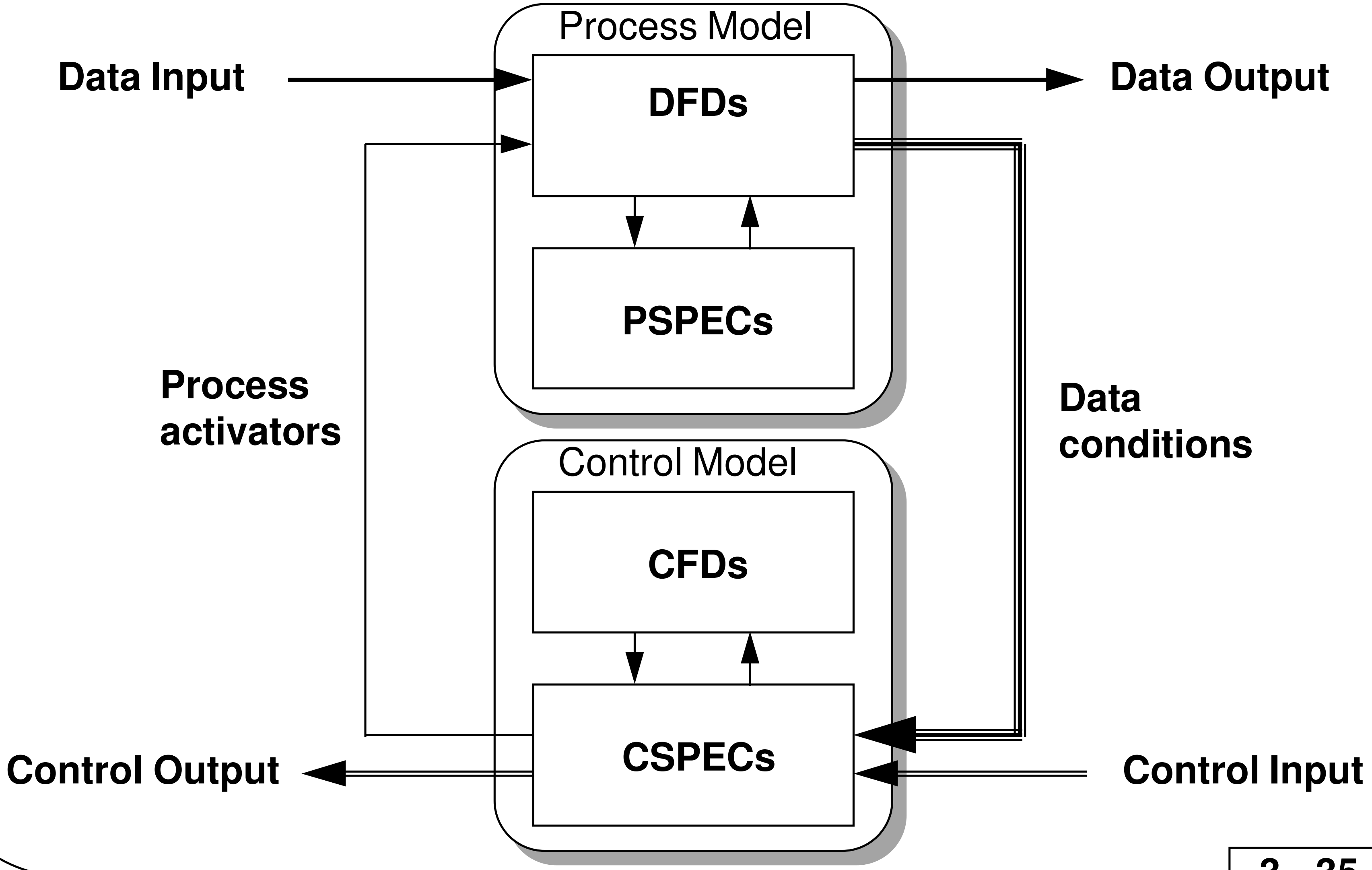
Robot movement record

Robot command file

Example: Manufacturing cell

Hatley-Pirbhai Extensions

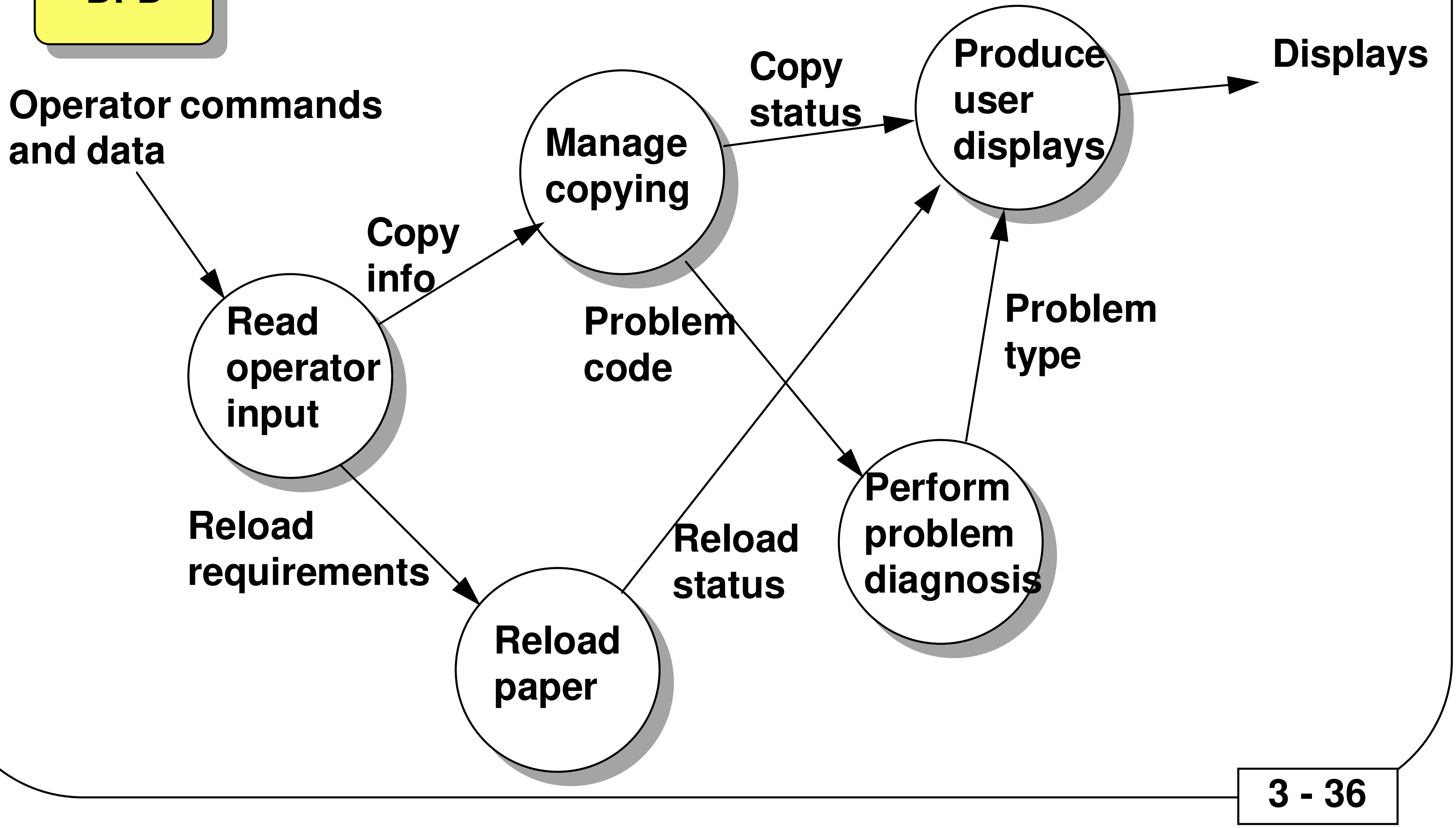
Combined Data Flowand Control Flow



Hatley-Pirbhai Extensions, Continued

Example: Office Photocopier Software

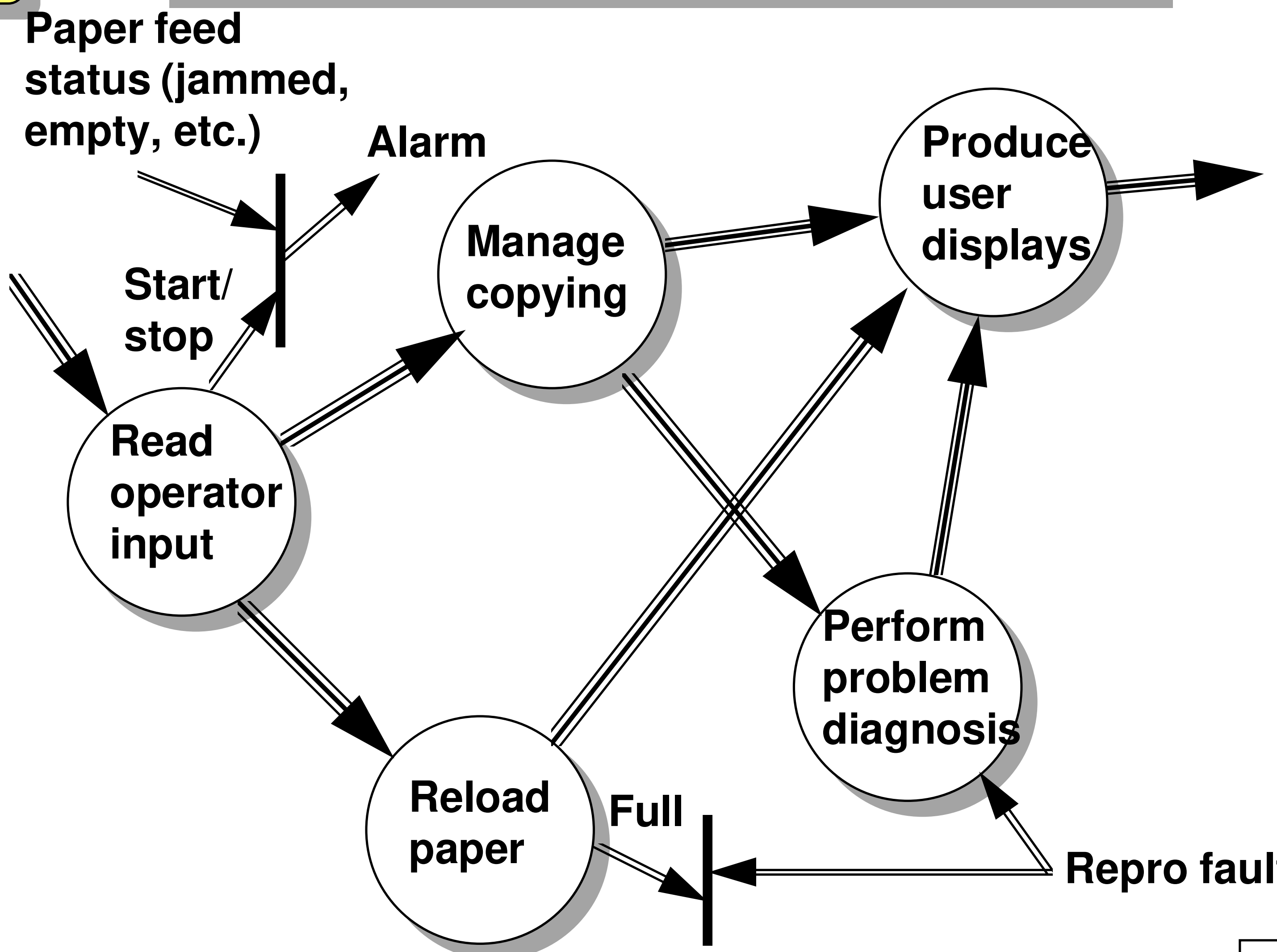
DFD



Hatley-Pirbhai Extensions, Continued

CFD

Example: Office Photocopier Software



Hatley-Pirbhai Extensions, Continued

Example: Office Photocopier Software

PSPEC

```
Read Operator Input:
if op_in = paper11
     then set form=11 inches;
if op_in = paper14
     then set form=14 inches;
if op-in = color
     then set style=colortype;
       .
       .
       .
```

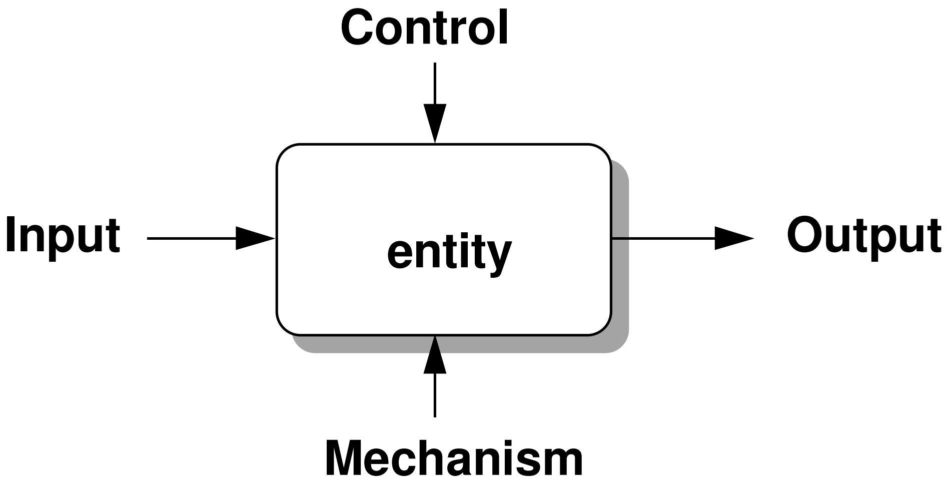
CSPEC

```
Alarm Condition:
if start && (feed_status = NOT
     ok) then set alarm (status);
         .
         .
         .
```

An Alternative: SADT[1]

Structured Analysis and Design Technique
(also known as IDEF 0)

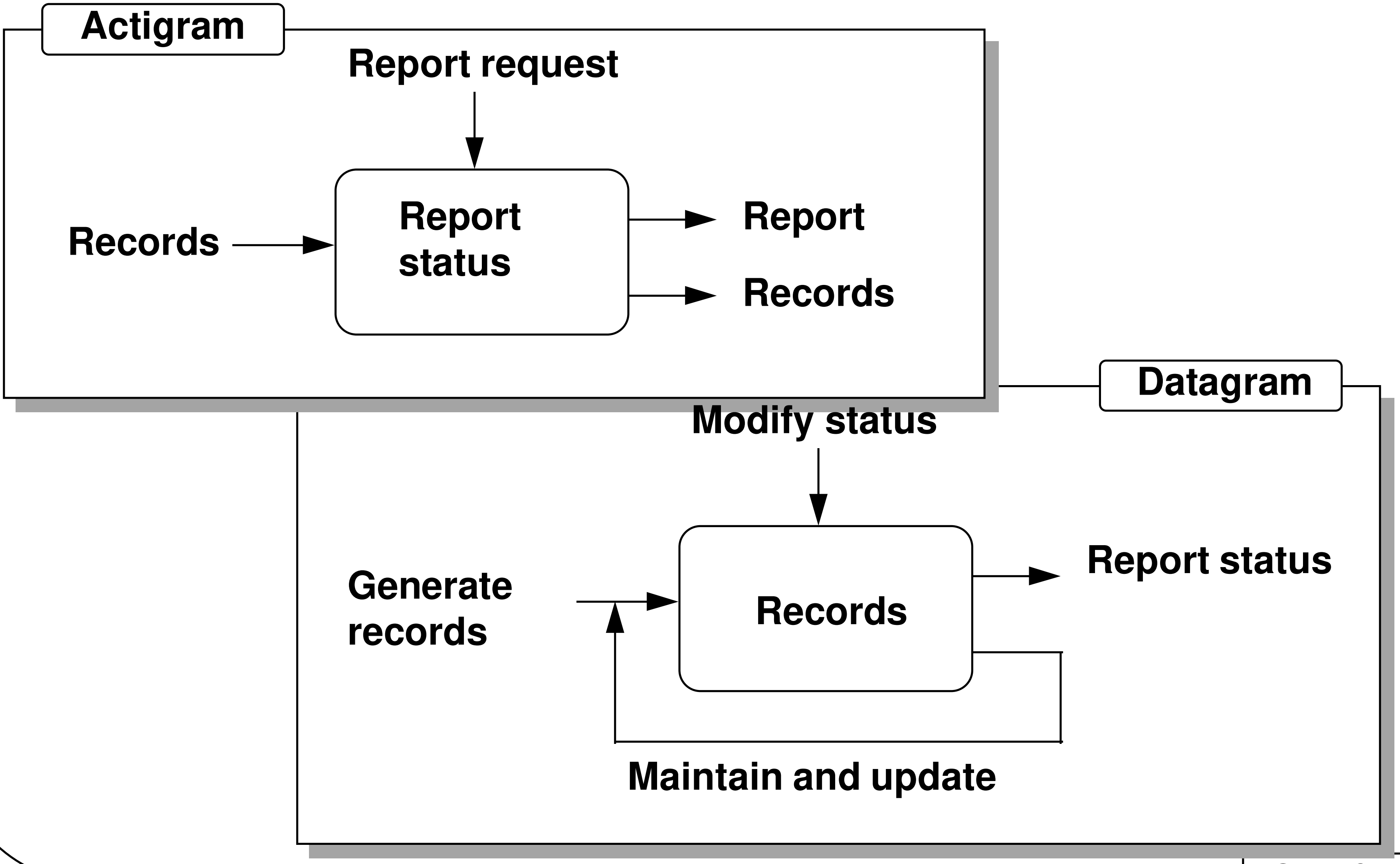
- A graphical notation
- Actigrams and datagrams that omminicate relations of information (data and control) and function within software
- Includes project control guidelines for applying methodology



[1] Trademark of Softech, Inc.

SADT, Continued

Actigrams and Datagrams



Actigram

Report request

Records

Report status

Report

Records

Datagram

Modify status

Generate records

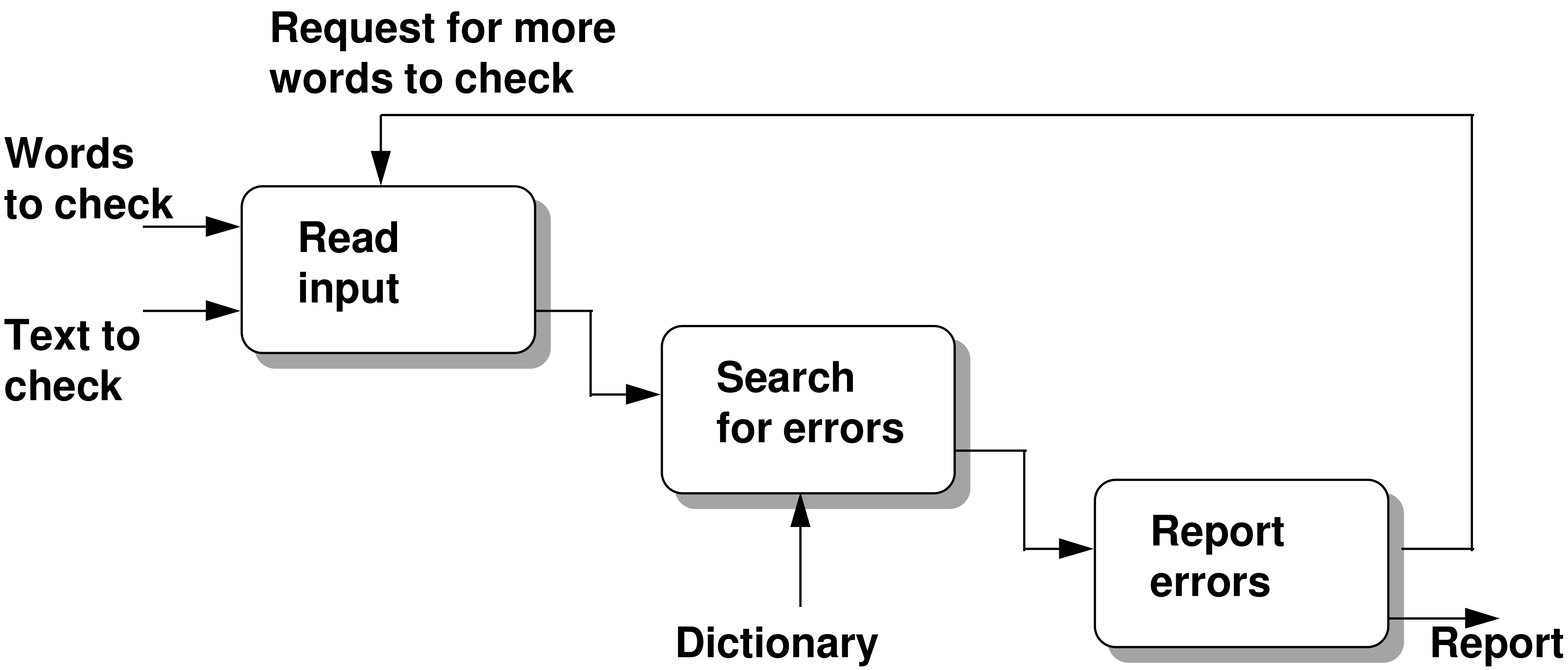
Records

Report status

Maintain and update

SADT, Continued

Example: Spelling Checker



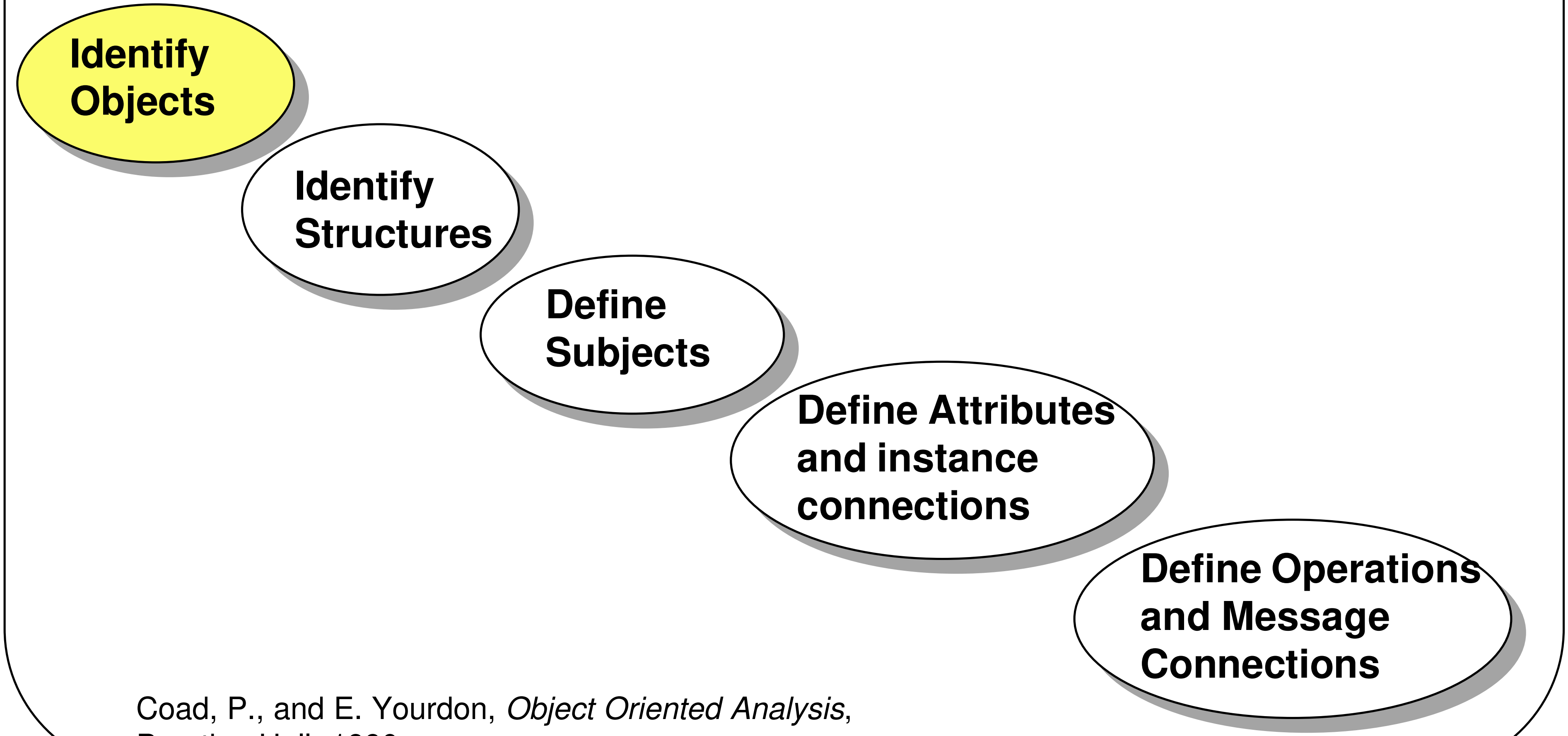
OOA: Object Oriented Analysis

- **Basic concepts**
- **How to identify objects**
  - **Identifying objects**
  - **Specifying attributes**
  - **Defining Operations**
  - **Communication between objects**
- **OOA modeling**
  - **Classification and assembly structures**
  - **Defining subjects**
  - **Instance connections and message paths**
  - **Prototyping**
- **Data Modeling**
  - **Data objects, attributes and relationships**
  - **E-R diagrams**

Basic Concepts

Object Oriented Development Process

Given a clear and complete statement of problem definition:



Identify Objects

Identify Structures

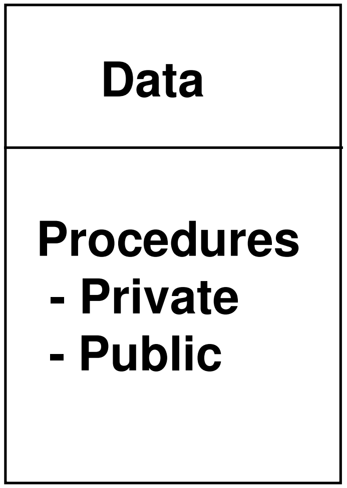
Define Subjects

Define Attributes and instance connections

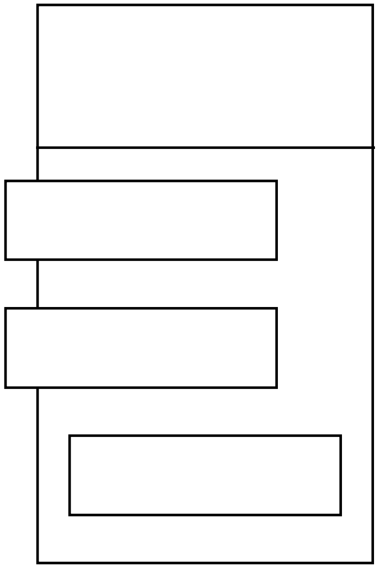
Define Operations and Message Connections

Coad, P., and E. Yourdon, *Object Oriented Analysis*,
Prentice-Hall, 1990.

Basic Concepts, Continued



Object

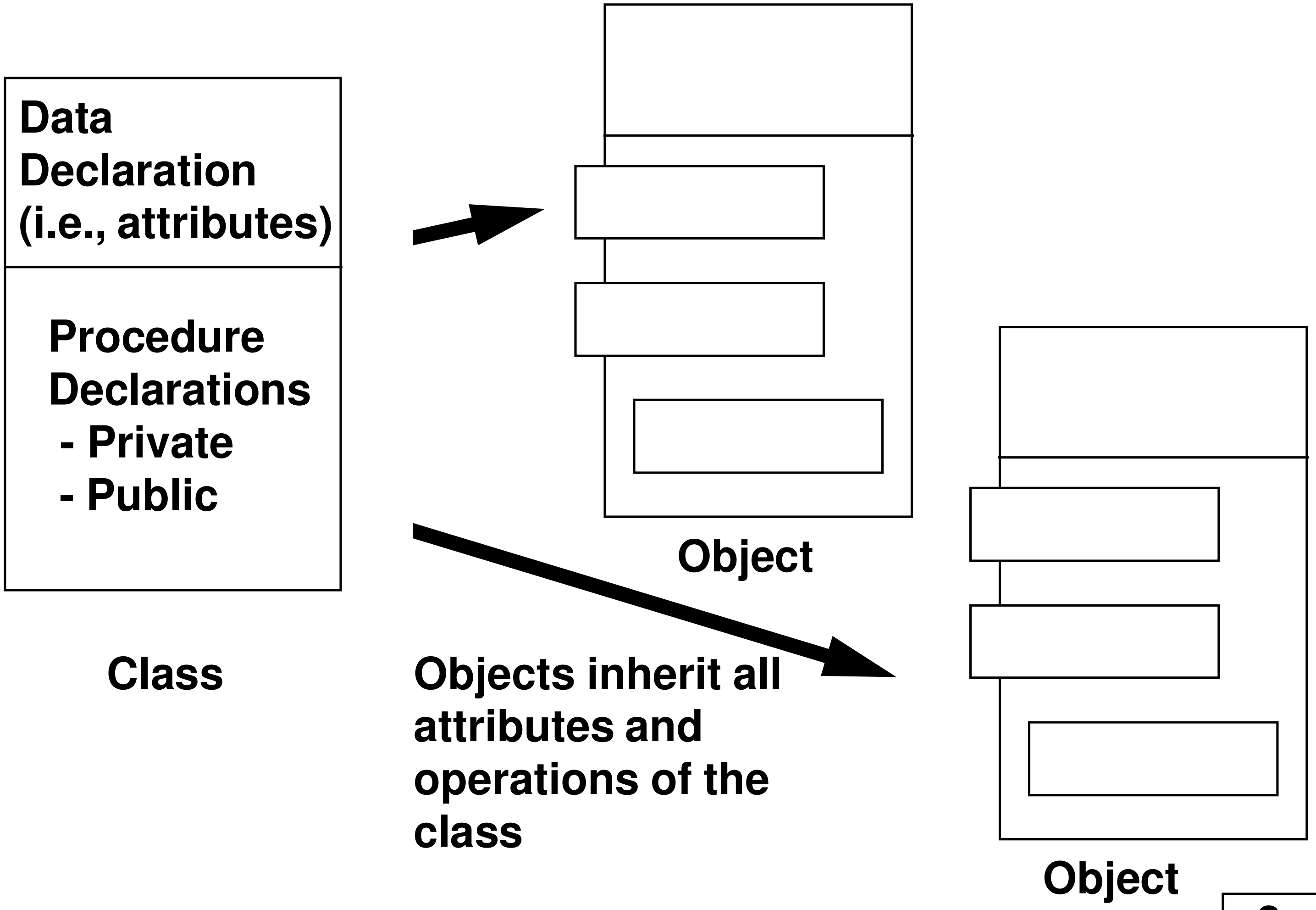


Booch Diagram
of an object

Objects are specific instances of classes (i.e., templates)

Basic Concepts, Continued

Objects are specific instances of classes (i.e., templates)



Data Declaration (i.e., attributes)

Procedure Declarations
- Private
- Public

Class

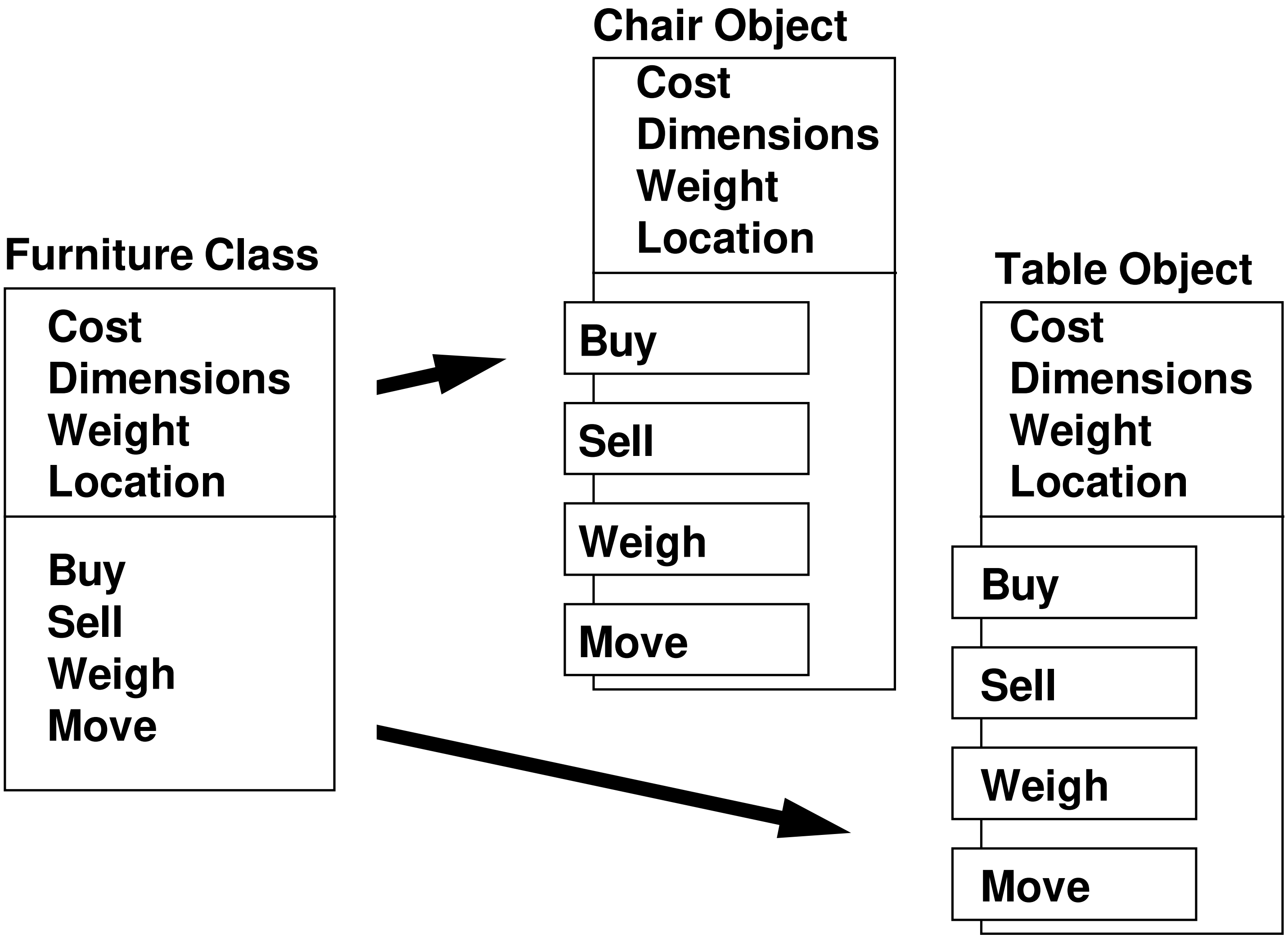
Object

Object

Objects inherit all attributes and operations of the class

Basic Concepts, Continued

Class/object Example



Chair Object

Cost
Dimensions
Weight
Location

Buy
Sell
Weigh
Move

Furniture Class

Cost
Dimensions
Weight
Location

Buy
Sell
Weigh
Move

Table Object

Cost
Dimensions
Weight
Location

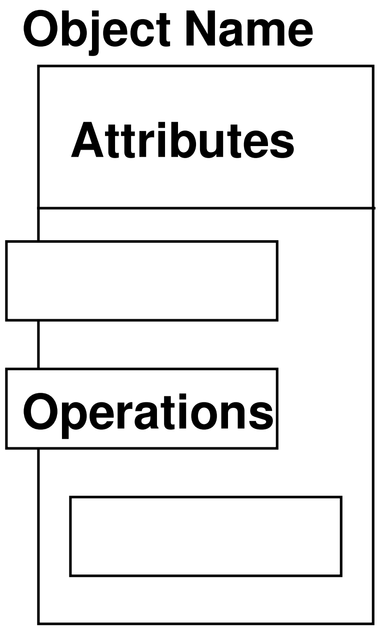
Buy
Sell
Weigh
Move

Basic Concepts, Continued

- Encapsulation - All class information is contained under one name which can be reused as one specification or program component.
- Inheritance - Objects and derived classes inherit all attributes and operations from their class descriptions.
- Polymorphism - Derived classes can add, delete, and redefine inherited attributes and operations.
- Messages - Procedures in separate objects communicate (i.e., call and return) via messages.

How to Identify Objects

Identifying Objects



Object Name

Attributes

Operations

__Potential Objects__ - examples

- External entities - devices, people
- Things - reports, displays, signals
- Occurances or events - interrupts
- Roles - manager, engineer
- Organizational units - division, group
- Places - shop floor, tail section
- Structures - sensors, computers

Identifying Objects - Example

Find the potential objects in the following narrative:

Safehome software enables the homeowner to configure the security system when it is installed, monitors all sensors connected to the security system, and interacts with the homeowner through a key pad and function keys contained in the *SafeHome* control panel.

During installation, the *SafeHome* control panel is used to "program" and configure the system. Each sensor is assigned a number and type, a master password is programmed for arming and disarming the system, and telephone number(s) is (are) input for dialing when a sensor event occurs.

When a sensor event is sensed by the software, it rings an audible alarm attached to the system. After a delay time that is specified by the homeowner during sysem configuration activities, the software dials a telephone number of a monitoring service, provides information about the location, and reports the nature of the event that has been detected. The number will be redialed every 20 secondss until telphone connection is abtained. .....

Identifying Objects - Example

Selection Criteria for classes and objects:

1. Retained information - information that must be remembered for system to function.
2. Needed services - operation are needed to change values of attributes.
3. Multiple attributes - focus on "major" information. Single or minor attributes can be collected together in single object.
4. Common attributes - attributes which apply to all occurances of the object.
5. Common operations - operations which apply to all occurances of the object.
6. Essential requirements - external entities that produce or consume information that is essential to system operation.

How to Identify Objects

Specifying Attributes

1. Scan the problem definition and select those things that belong to an object.
2. For each object, ask "what data items (composite or elementary) fully define this object in the context of the problem?
3. For example, using the *SafeHome* system object:

sensor_info = sensor_type + sensor_number + alarm_threshold

alarm_response = delay_time + telephone_number + alarm_type

activate/deactivate_info = master_password + tries_allowed + temp_password

id_info = system_ID + verification_phone_no. + system_status

How to Identify Objects

Defining Operations

Operations are of three types:

- Manipulation - add, delete, reformat, select, initialize
- Computation - equations, transformations
- Monitoring - occurance of a controlling event

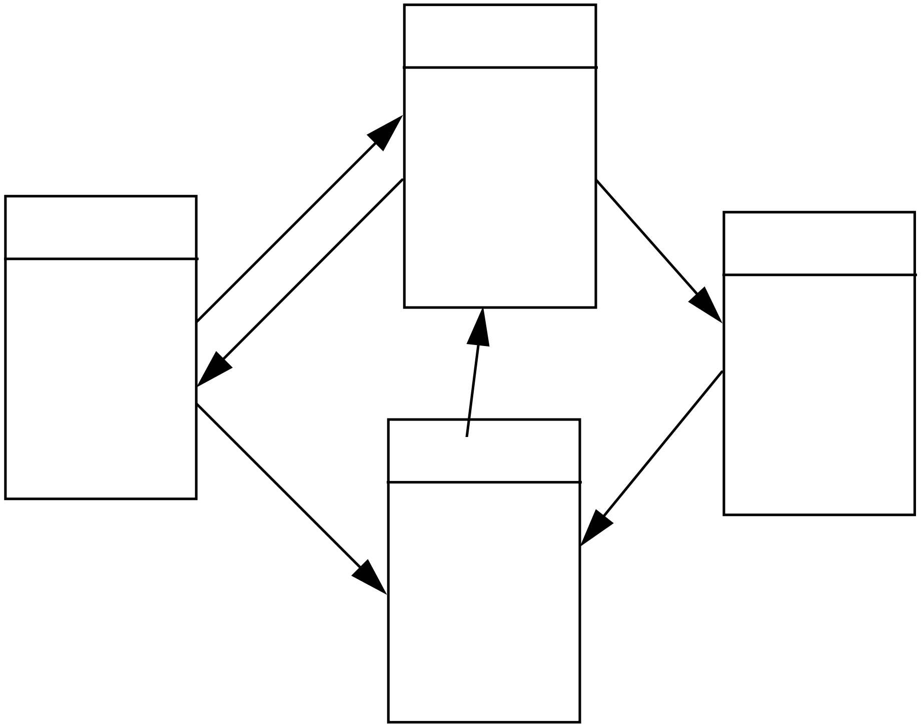
To derive a set of operations for each object:

1. Scan the problem definition and grammatically parse it for verbs to be candidate operations that belong to each object.
2. Try defining the candidate operations for objects defining the SafeHome system (use description in prior slide)

How to Identify Objects

Interobject Communication

During requirements definition, eplicite messages need not be known. Only general object interaction should be defined.

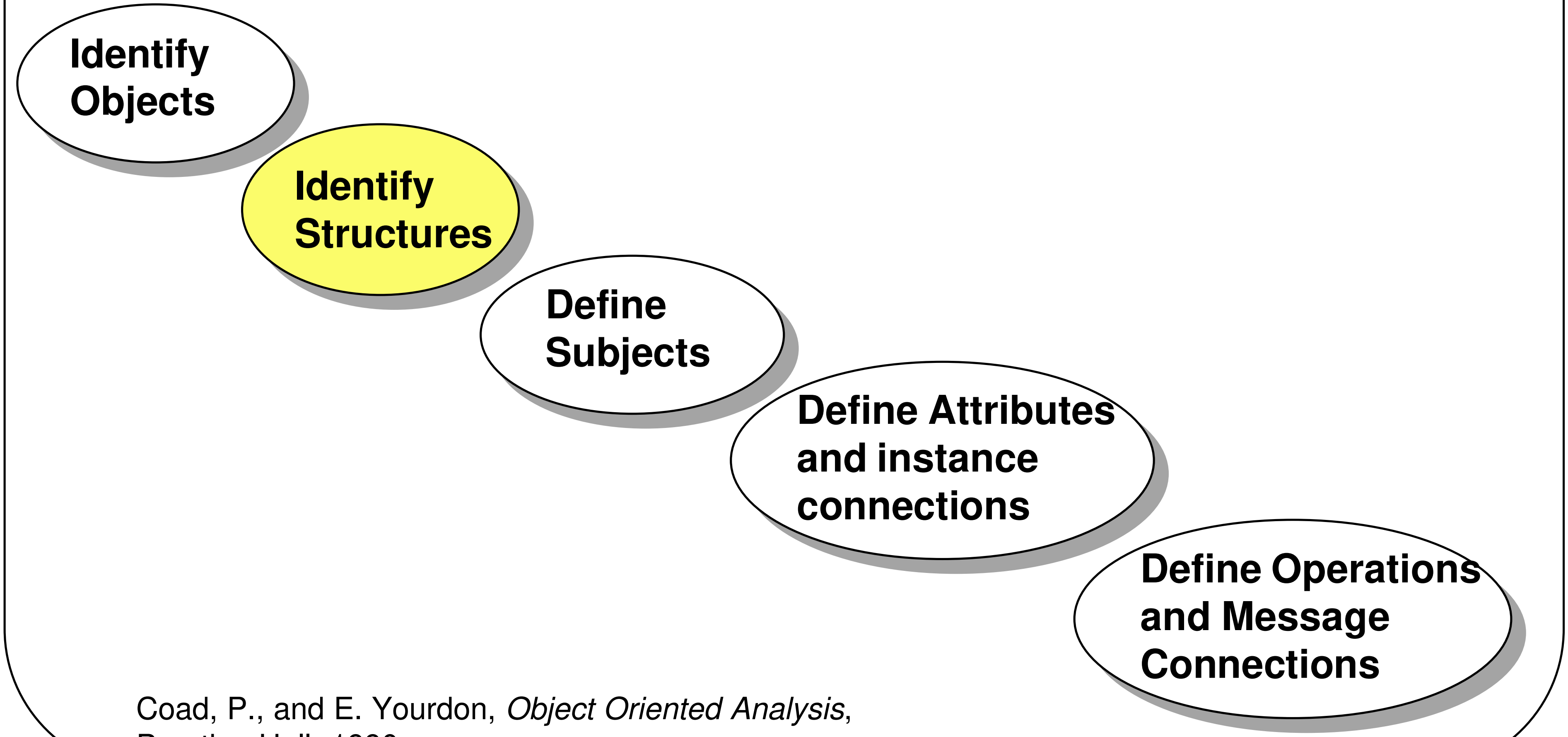


Note: Messages initiated by procedures in objects, and are sent to precedures in objects.

msg: (dest, op, args)

Object-Oriented Development Process

Given a clear and complete statement of problem definition:

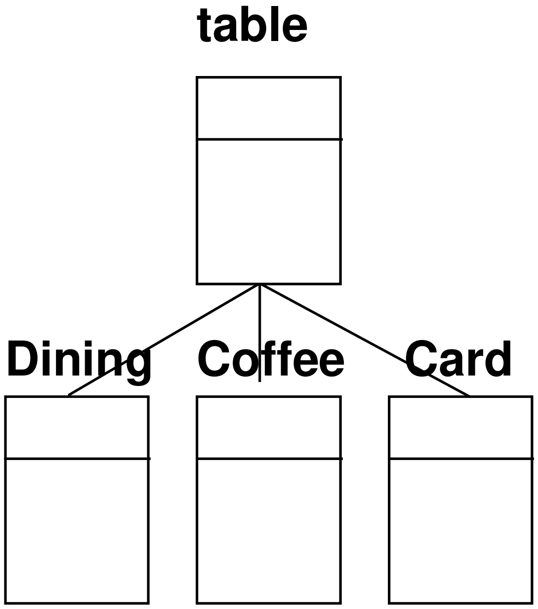

Identify Objects
Identify Structures
Define Subjects
Define Attributes and instance connections
Define Operations and Message Connections

Coad, P., and E. Yourdon, *Object Oriented Analysis*,
Prentice-Hall, 1990.

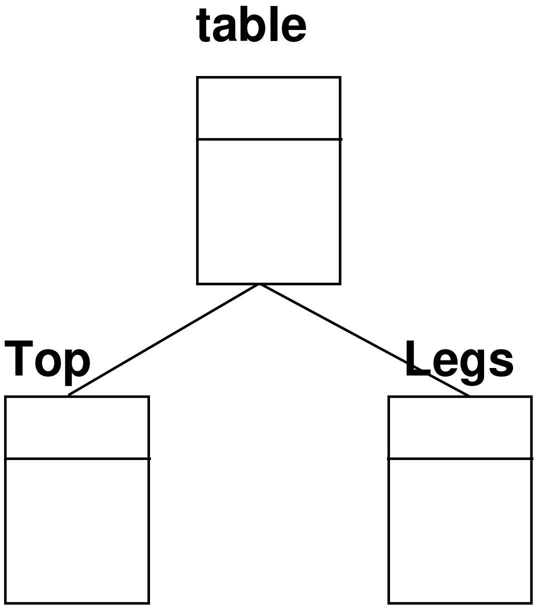
OOA Modeling

Classification and Assembly Structures

Once objects have been defined, structure groups of them into classification trees or assembly trees:



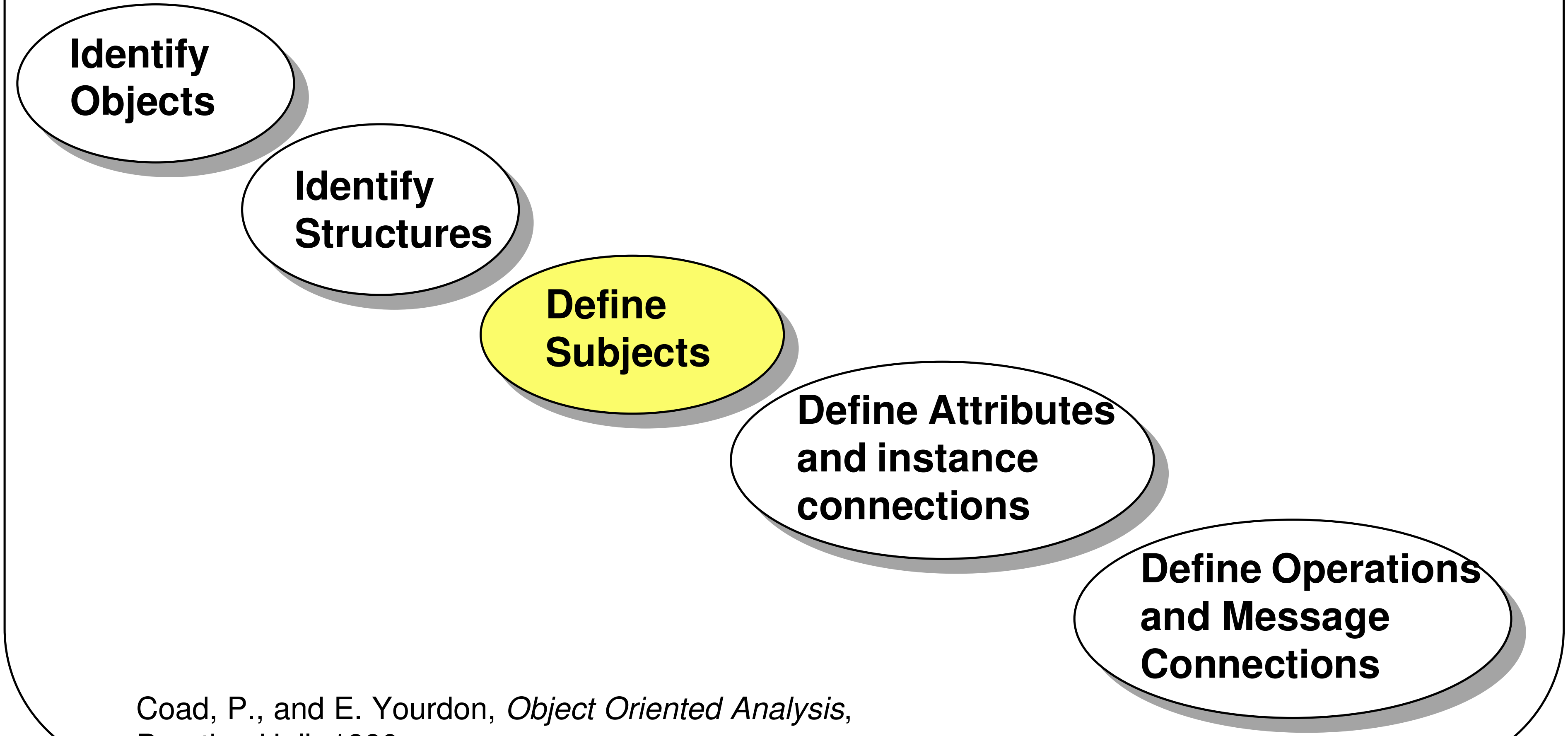
Classification Structure



Assembly Structure

Object-Oriented Development Process

Given a clear and complete statement of problem definition:



Identify Objects

Identify Structures

Define Subjects

Define Attributes and instance connections

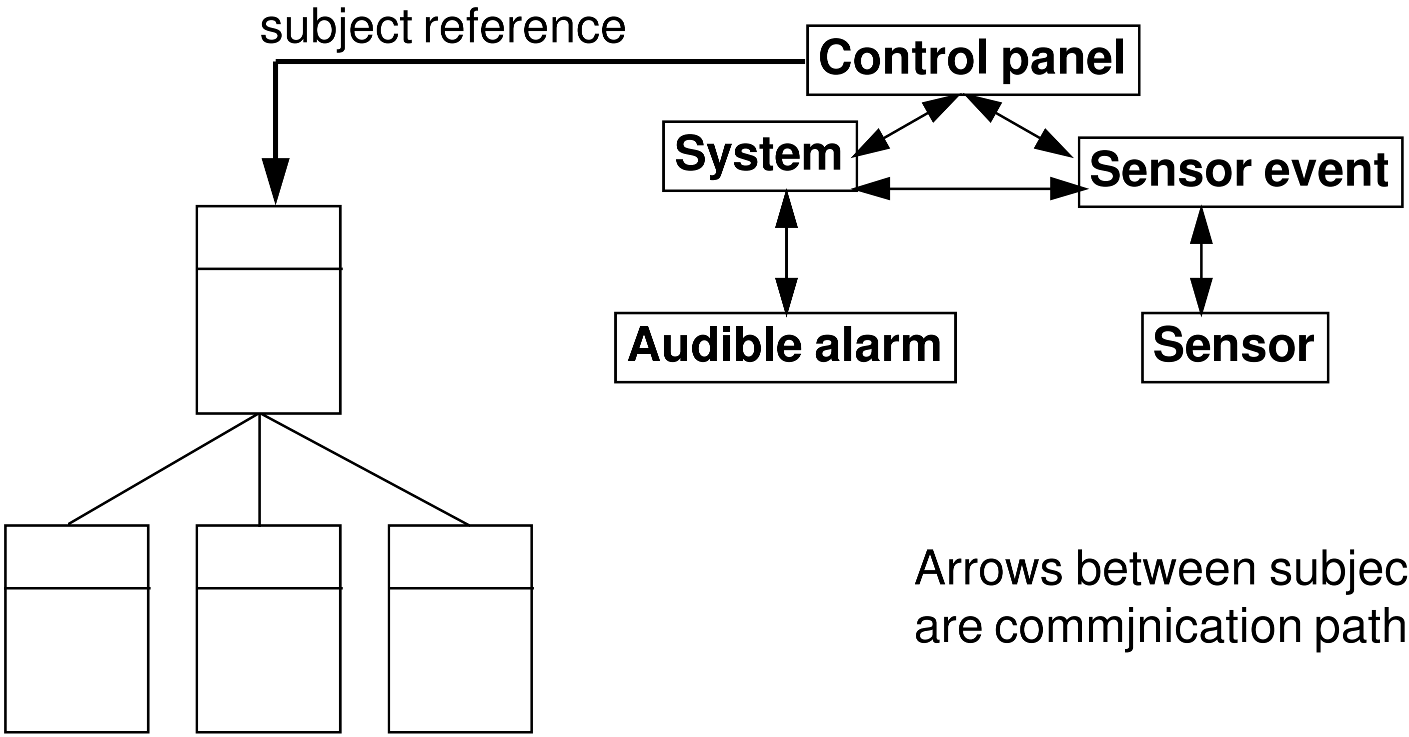
Define Operations and Message Connections

Coad, P., and E. Yourdon, *Object Oriented Analysis*, Prentice-Hall, 1990.

OOA Modeling

Defining Subjects

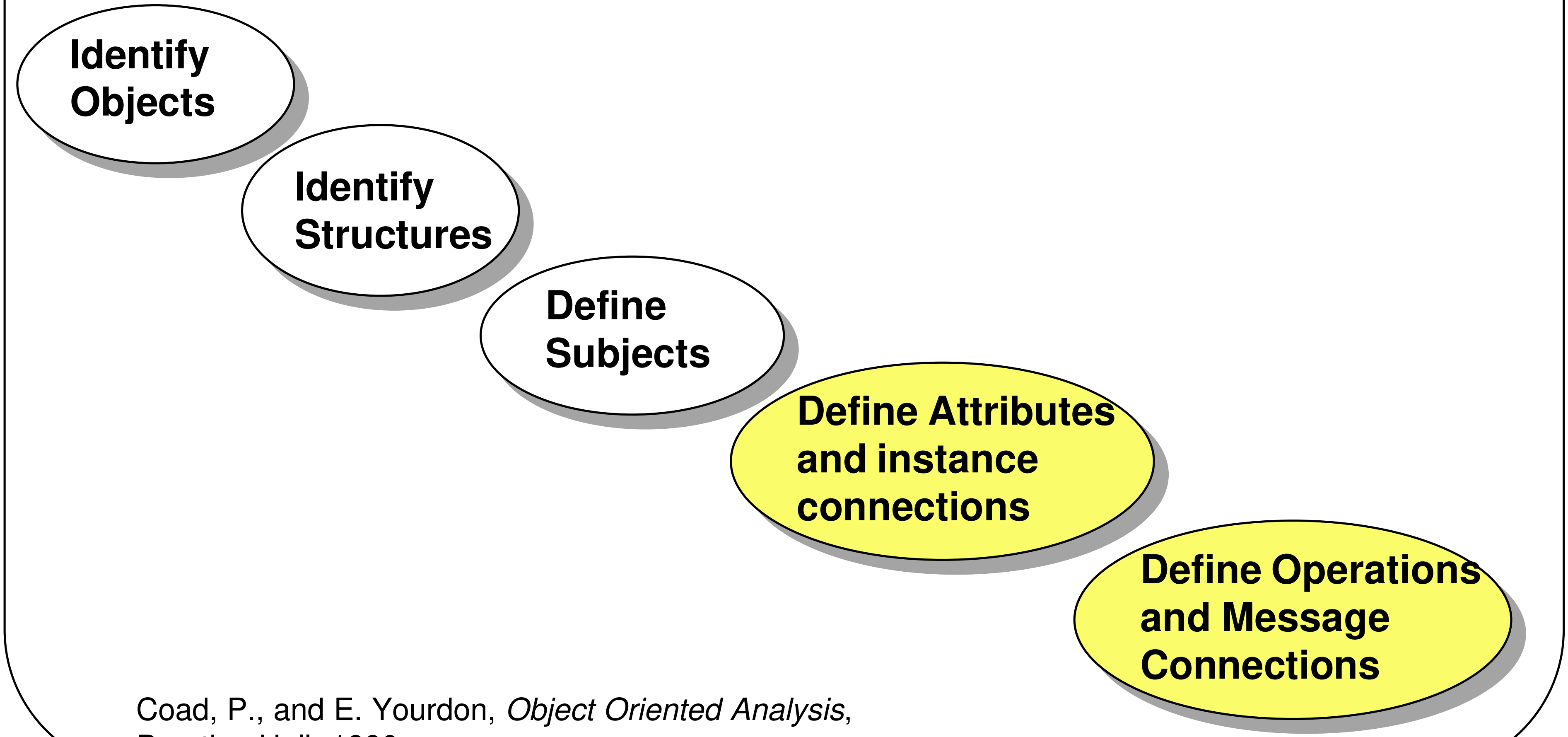
For large OOA models with hundreds of objects and dozens of structures, organize the structures in to subjects which can be referenced by a single name or ID.



Arrows between subjects
are commjnication paths.

Object-Oriented Development Process

Given a clear and complete statement of problem definition:



Identify Objects

Identify Structures

Define Subjects

Define Attributes and instance connections

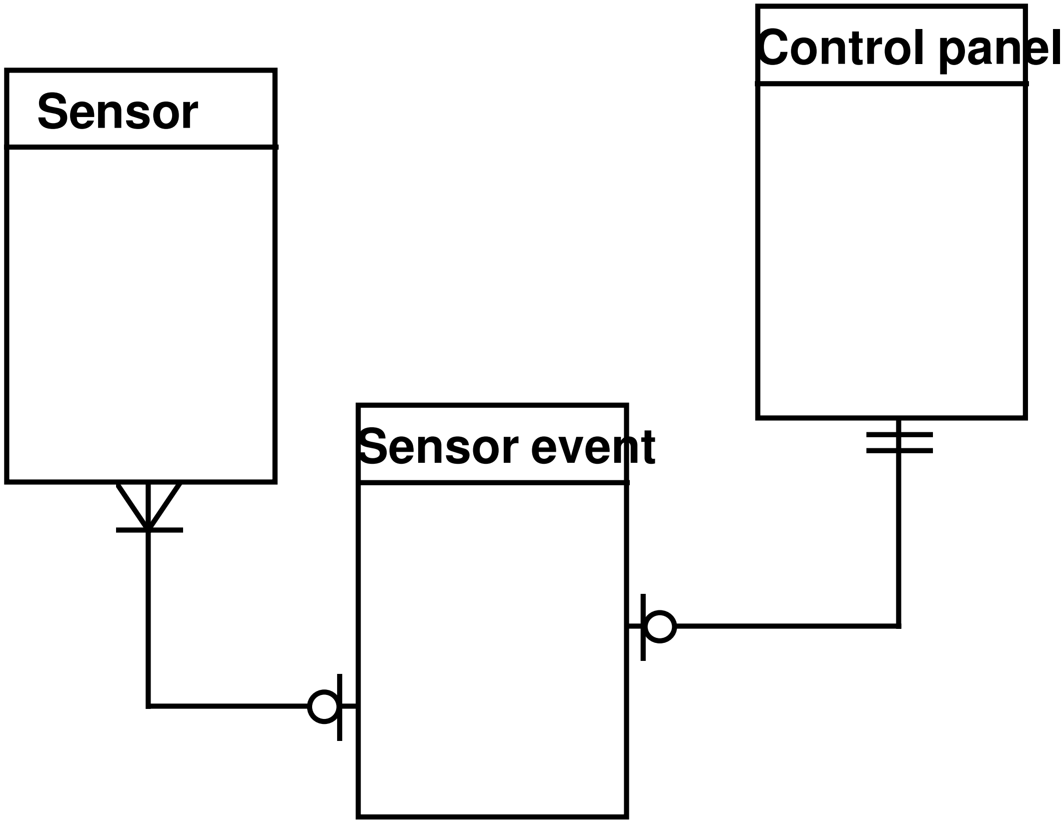
Define Operations and Message Connections

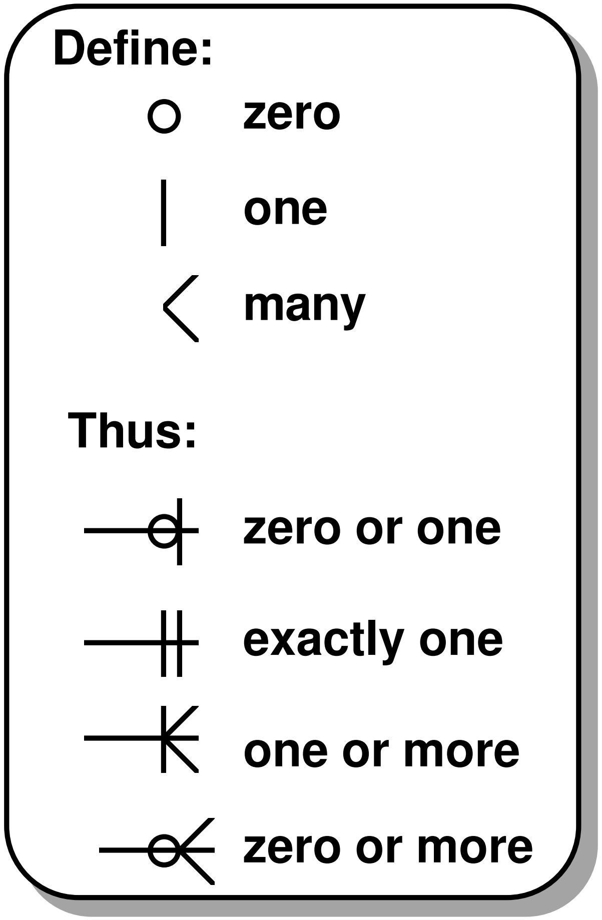
Coad, P., and E. Yourdon, *Object Oriented Analysis*, Prentice-Hall, 1990.

OOA Modeling

Instance Connections and Message Paths

Analyist should define specific relationships between objects:





Define:

○ zero

| one

< many

Thus:

zero or one

exactly one

one or more

zero or more

OOA Modeling

Prototyping

OOA can lead to very effective prototyping techniques

- Reuse defined, coded, and tested objects

- Establish library of quality objects and save analysis info as well as code and tested objects

- Use existing object specifications in the development of new products.

Data Modeling

Data Objects, Attributes and Relationships

OOA concepts arose out of data-intensive analysis techniques (called *data modeling* or *information maodeling*) that have been in existence for years (especially in database systems).

Recent uses of data modeling are seen in defining data formats for interchanging data between CAD systems, computers, and manufacturing organizations.

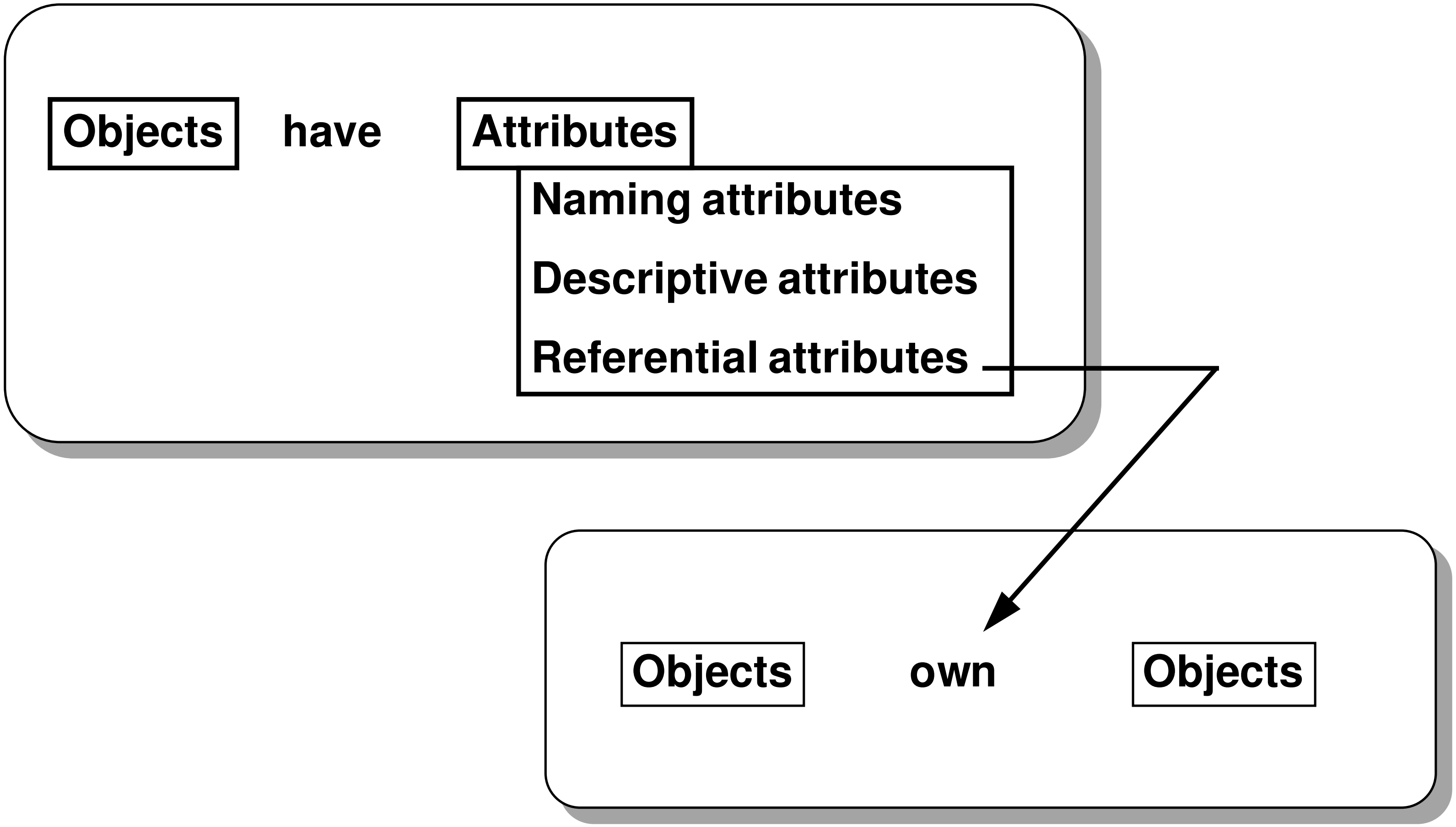
Some terms:

<u>schema</u> - data model used in databases

<u>protocol</u> - data model used in digital communications

<u>framework</u> - data models used to interchange data between CAD systems and manufacturing organizations

Data Objects, Attributes, and Relationships



Objects have Attributes

Naming attributes

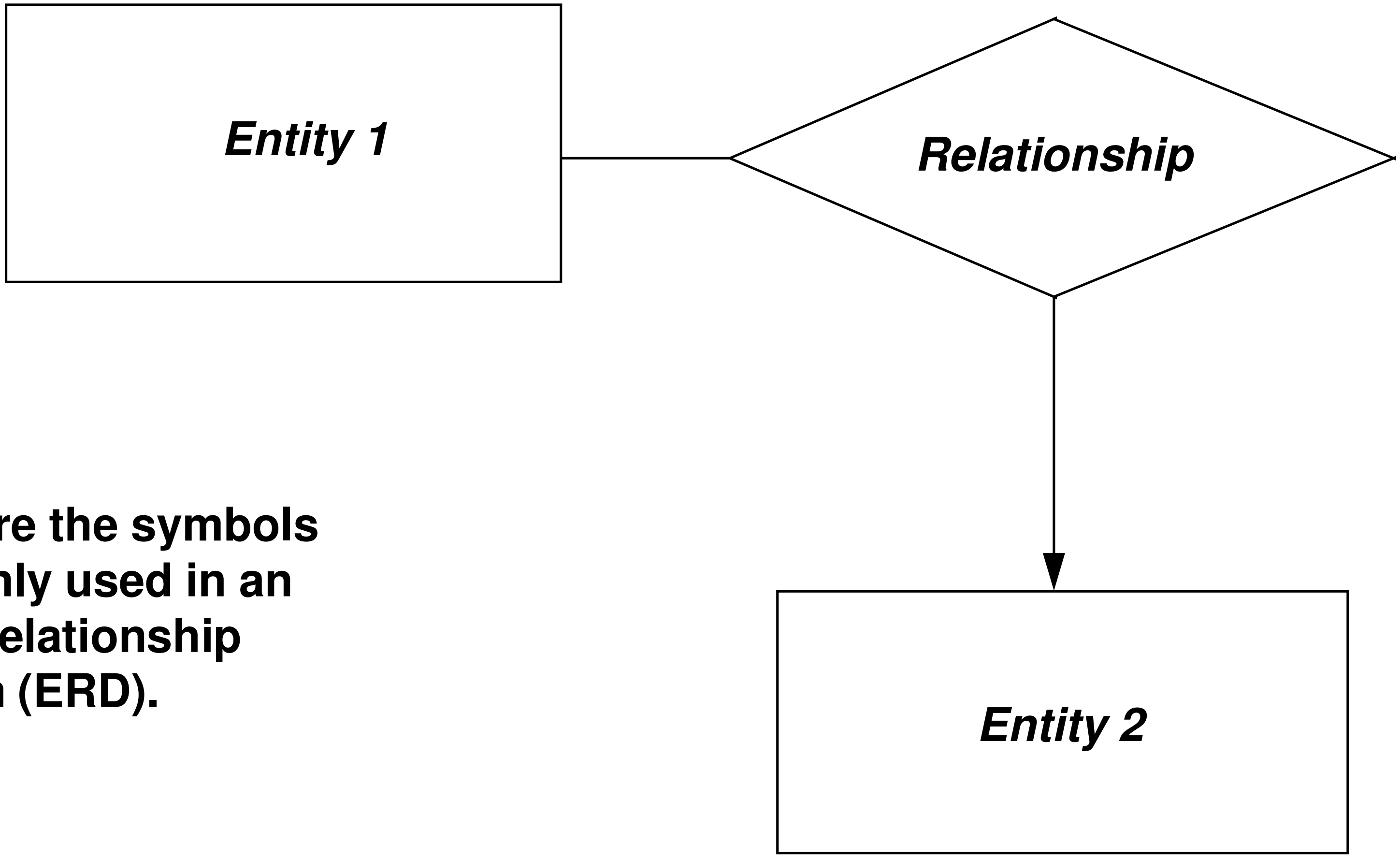
Descriptive attributes

Referential attributes

Objects own Objects

Data Modeling

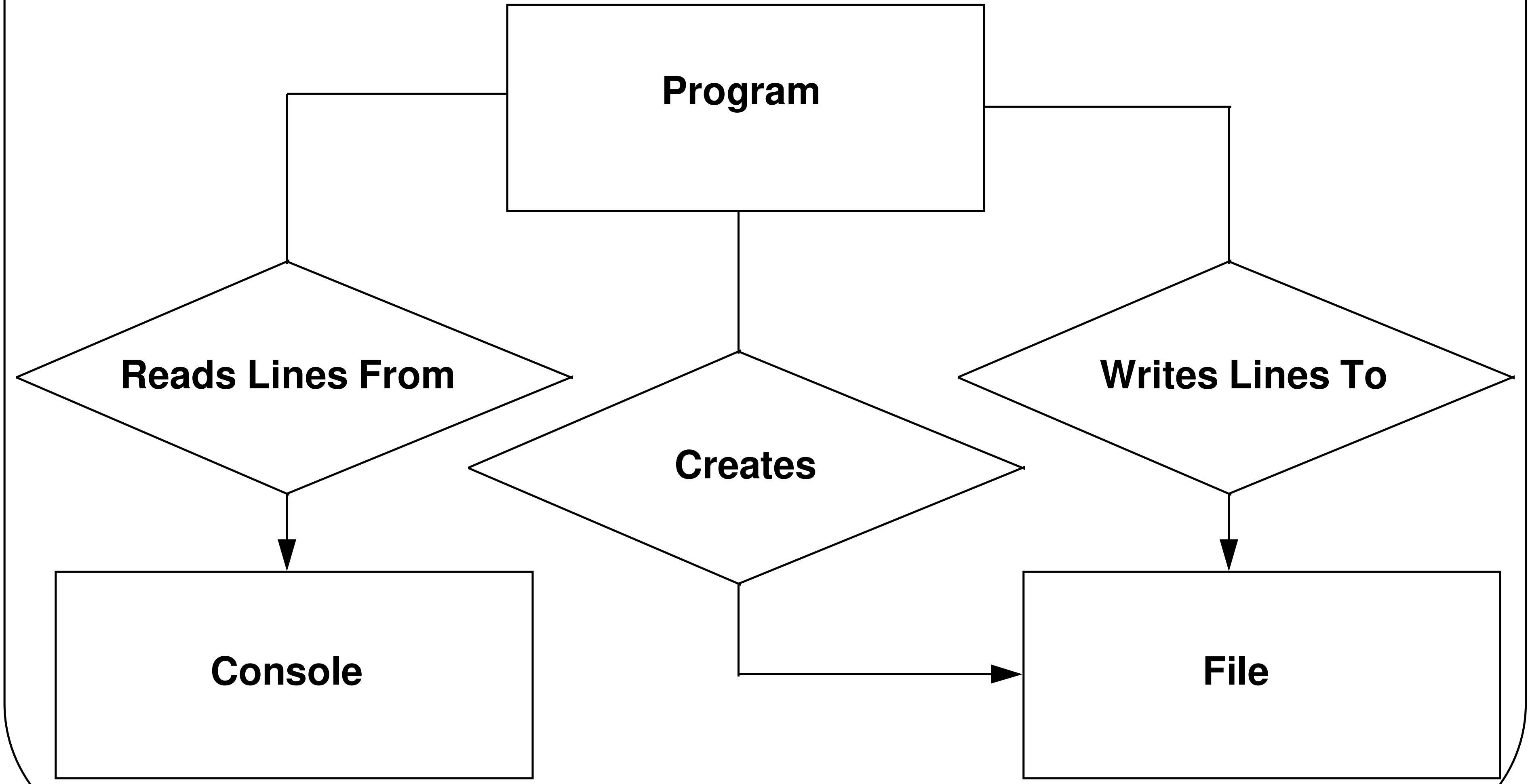
Entity-Relationship Diagrams



These are the symbols commonly used in an Entity-Relationship Diagram (ERD).

Data Modeling, Continued

Entity-Relationship Diagrams - Example



Automated Tools

- are often graphically-oriented
- may provide consistency checking
- support the development of the data dictionary
- usually support the development of DoD-STD-2167A documentation