## Data Flow Diagrams

A structured analysis technique that employs a set of visual representations of the data that moves through the organization, the paths through which the data moves, and the processes that produce, use, and transform data.

## Why Data Flow Diagrams?

- Can diagram the organization or the system
- Can diagram the current or proposed situation
- Can facilitate analysis or design
- Provides a good bridge from analysis to design
- Facilitates communication with the user at all stages

## Types of DFDs

- Current how data flows now
- Proposed how we'd like it to flow
- Logical the "essence" of a process
- Physical the implementation of a process
- Partitioned physical system architecture or high-level design

## Levels of Detail

- Context level diagram shows just the inputs and outputs of the system
- Level 0 diagram decomposes the process into the major subprocesses and identifies what data flows between them
- Child diagrams increasing levels of detail
- Primitive diagrams lowest level of decomposition

## Recommended Progression

- Current logical diagrams
  - start with context level
  - decompose as needed for understanding
- Proposed logical diagrams
  - start at level where change takes place
  - decompose as far as possible
- Current physical diagrams
  - at level of change
- Proposed physical diagrams
  - same levels as proposed logical
  - lower levels become design

# Four Basic Symbols

Source/ Sink Data Flow

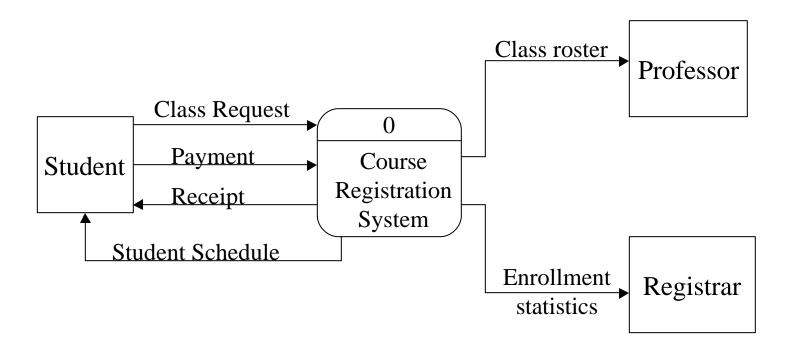
#
Process

# Data Store

# Context Level Diagram

- Just one process
- All sources and sinks that provide data to or receive data from the process
- Major data flows between the process and all sources/sinks
- No data stores

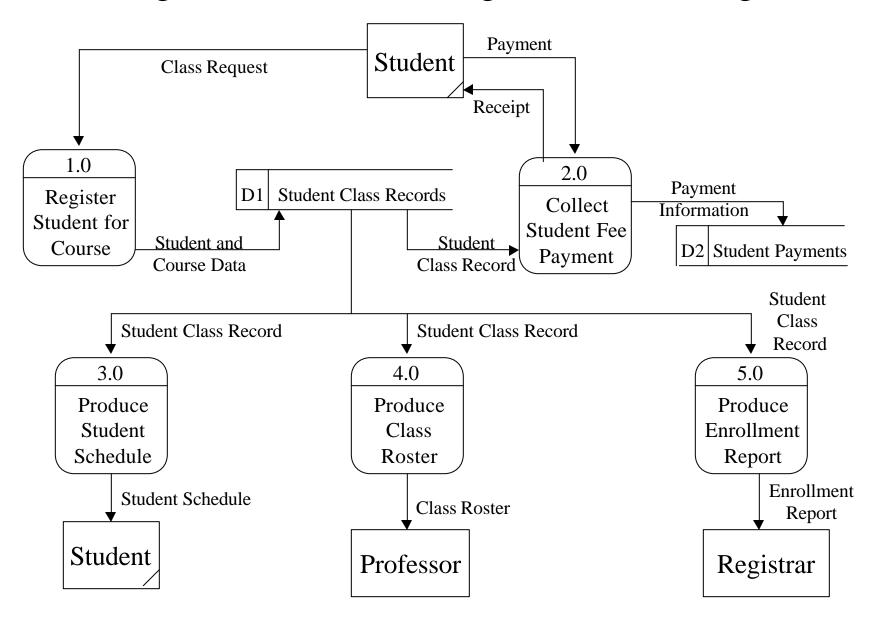
## Course Registration: Context level Diagram



## Level 0 Diagram

- Process is "exploded"
- Sources, sinks, and data flows repeated from context diagram
- Process broken down into subprocesses, numbered sequentially
- Lower-level data flows and data stores added

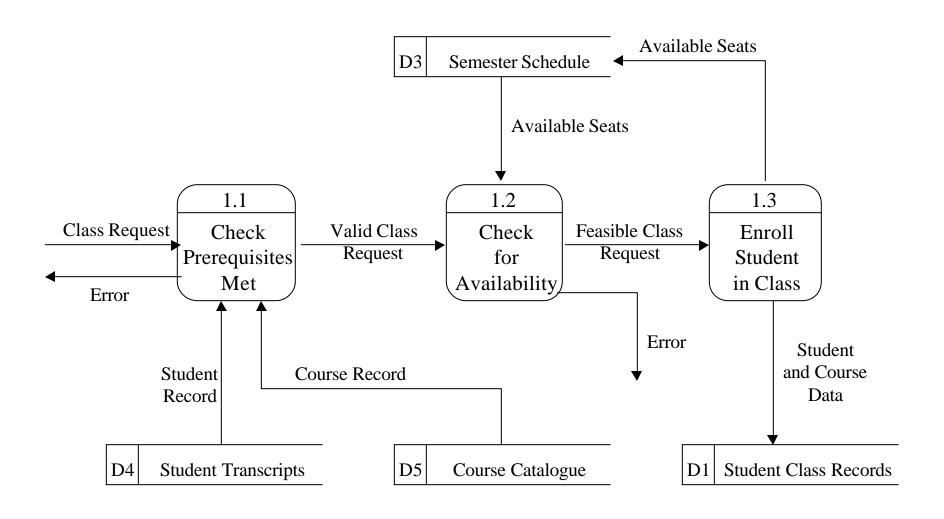
### Course Registration: Current Logical Level 0 Diagram



## Child Diagrams

- "Explode" one process in level 0 diagram
- Break down into lower-level processes, using numbering scheme
- Must include all data flow into and out of "parent" process in level 0 diagram
- Don't include sources and sinks
- May add lower-level data flows and data stores

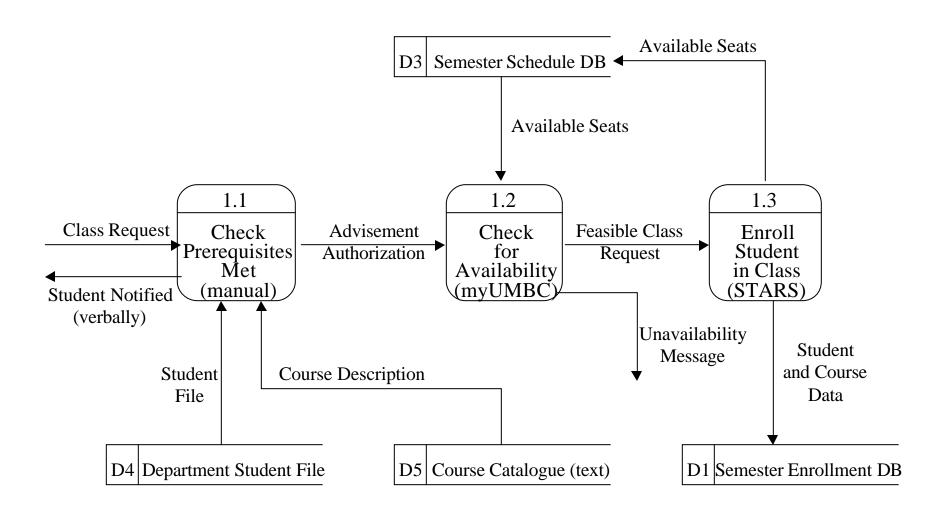
### Course Registration: Current Logical Child Diagram



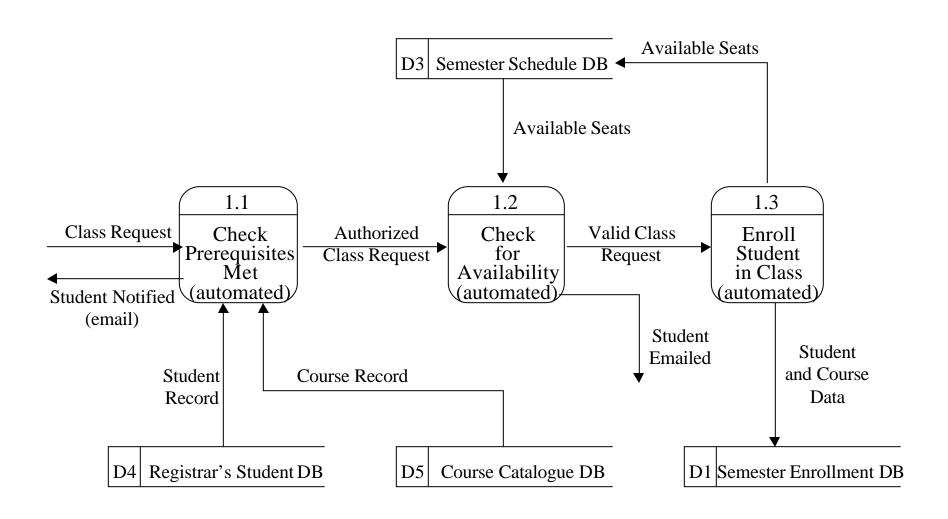
## Physical DFDs

- Model the implementation of the system
- Start with a set of child diagrams or with level 0 diagram
- Add implementation details
  - indicate manual vs. automated processes
  - describe form of data stores and data flows
  - extra processes for maintaining data

### Course Registration: Current Physical Child Diagram



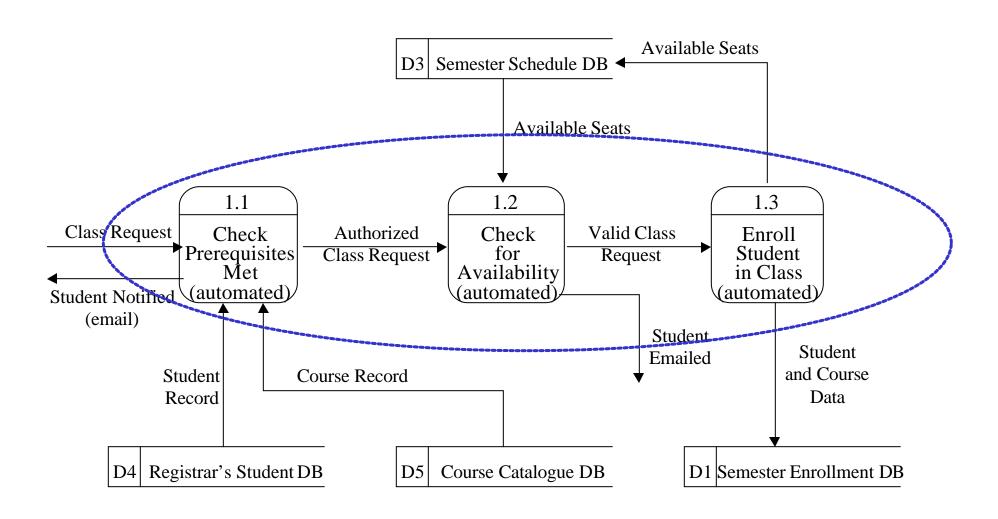
### Course Registration: Proposed Physical Child Diagram



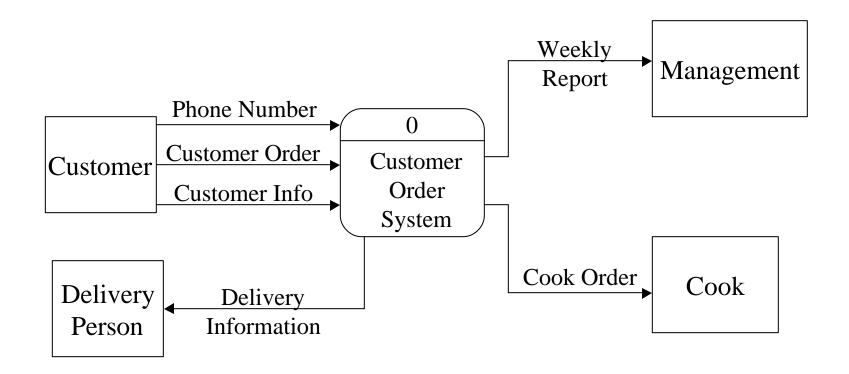
# Partitioning a physical DFD

- Part of system design
- System architecture
  - high-level design
  - overall shape of system
  - some standard architectures
- Decide what processes should be grouped together in the system components

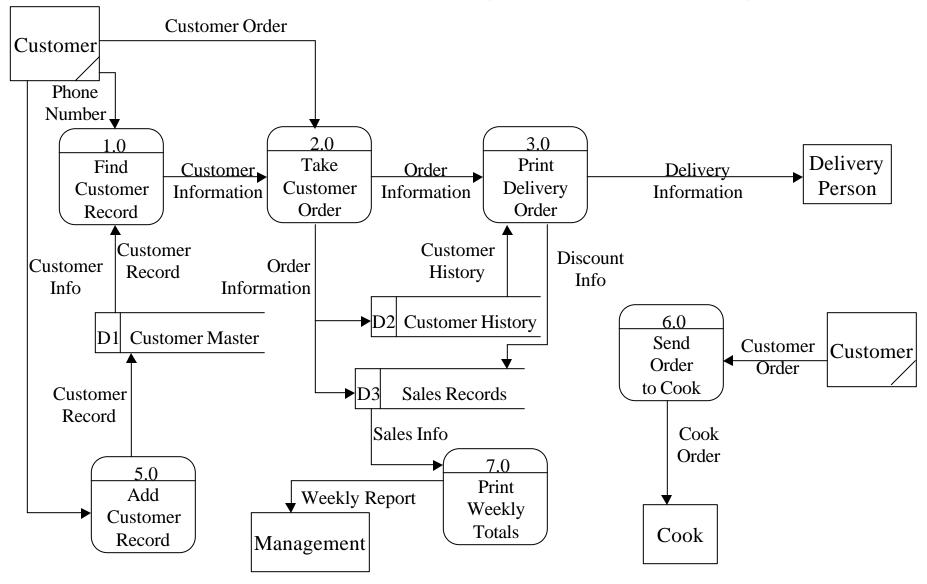
### Course Registration: Physical diagram (partitioned)



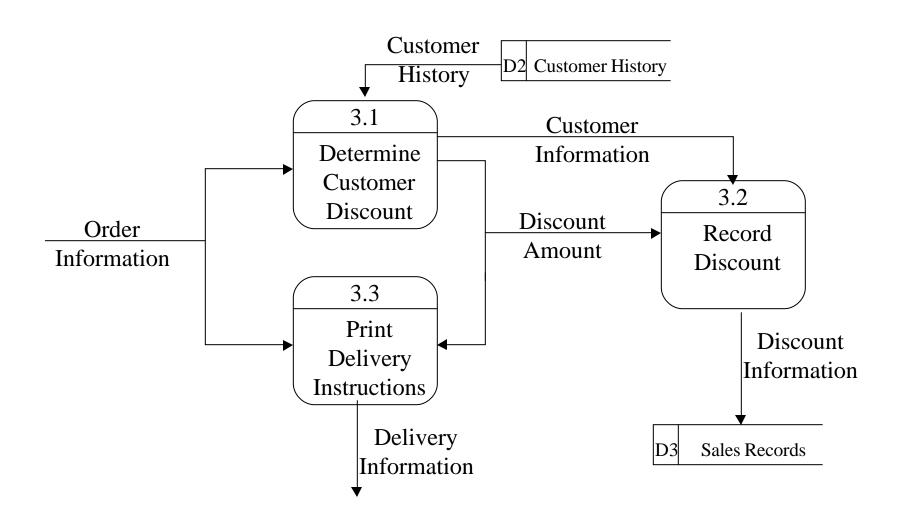
### Perfect Pizza: Context Level Diagram



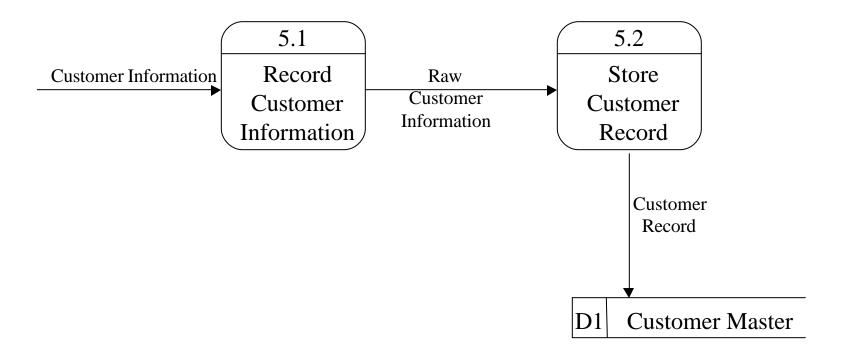
## Perfect Pizza: Current Logical Level 0 Diagram



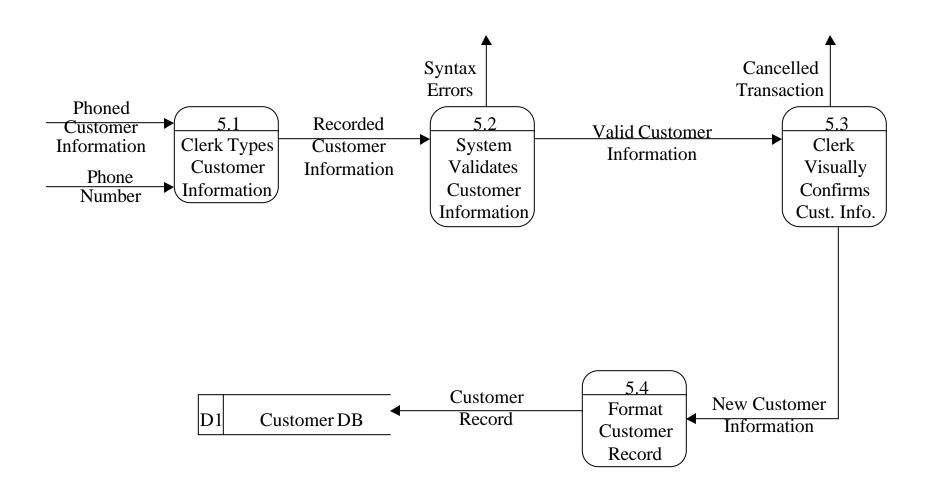
### Perfect Pizza: Current Logical Child Diagram



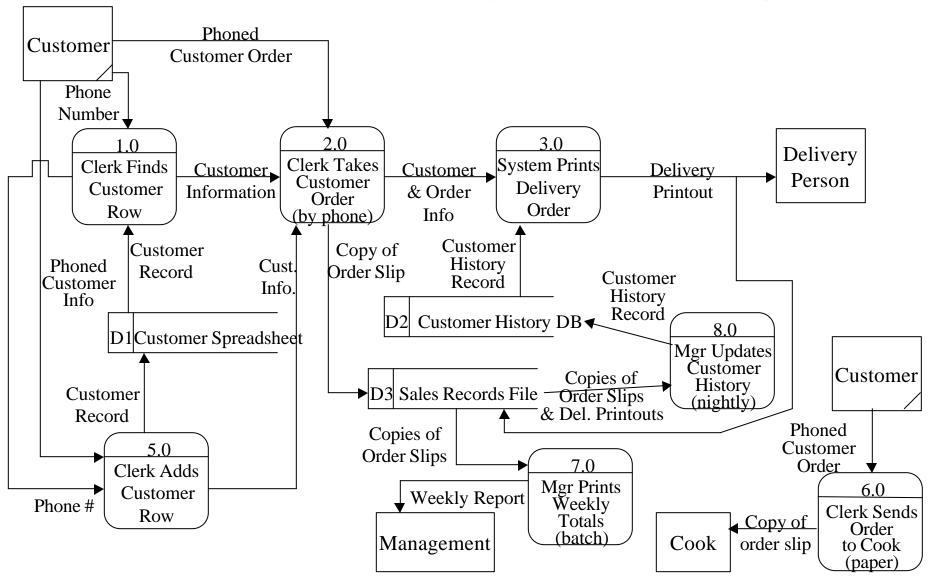
## Perfect Pizza: Current Logical Child Diagram



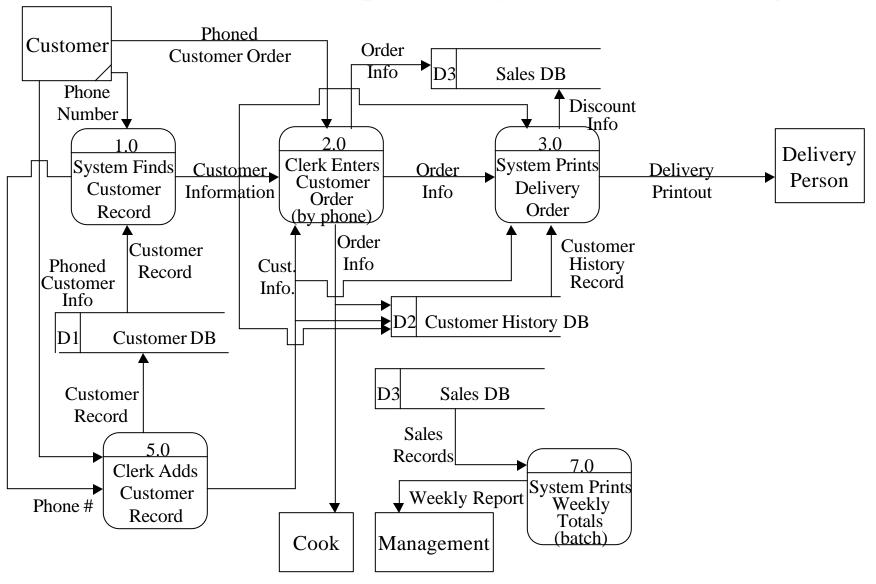
## Perfect Pizza: Physical Child Diagram



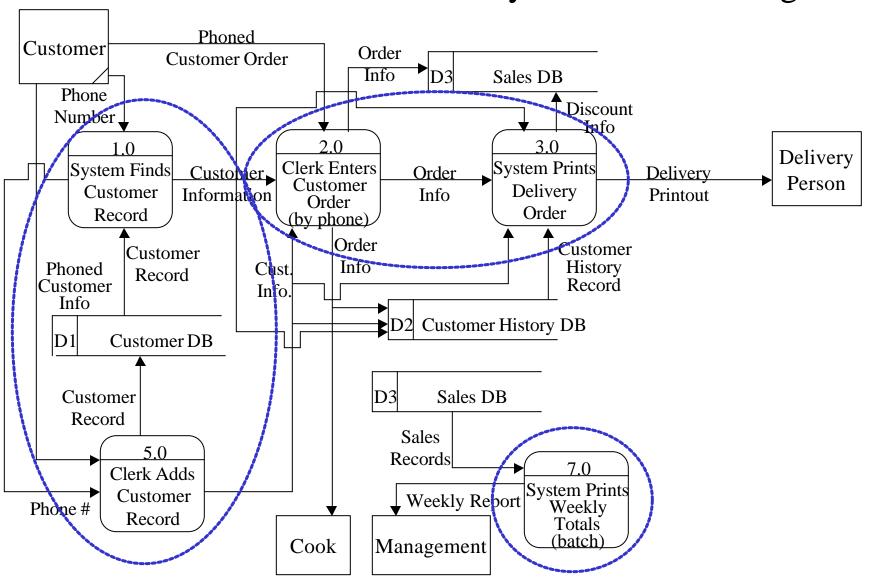
### Perfect Pizza: Current Physical Level 0 Diagram



### Perfect Pizza: Proposed Physical Level 0 Diagram



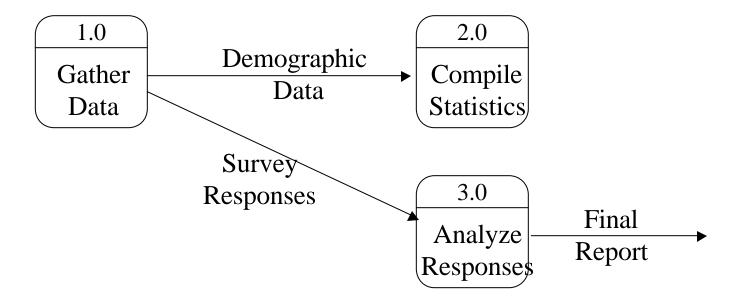
### Perfect Pizza: Partitioned Physical Level 0 Diagram

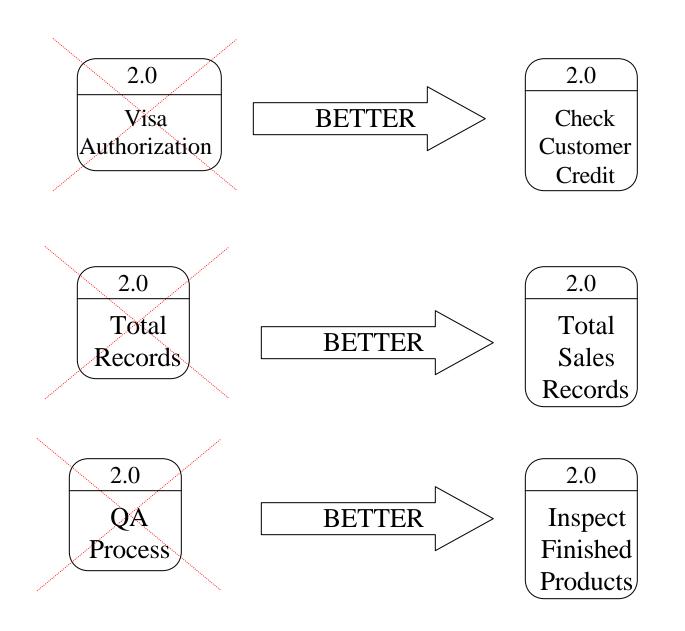


# Data Flow Diagramming Rules

### Processes

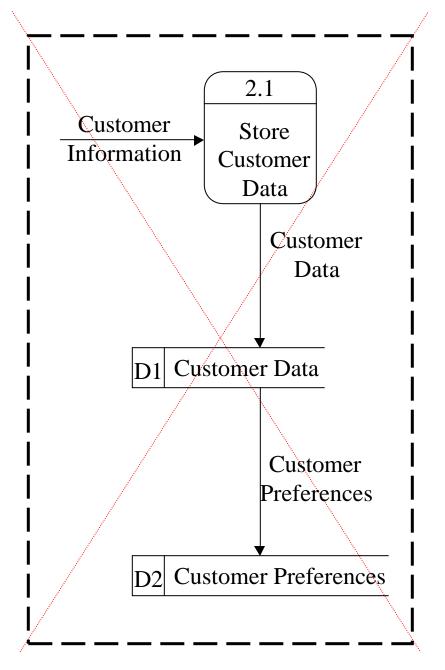
- a process must have at least one input
- a process must have at least one output
- a process name (except for the context level process) should be a verb phrase
  - usually three words: verb, modifier, noun
  - on a physical DFD, could be a complete sentence

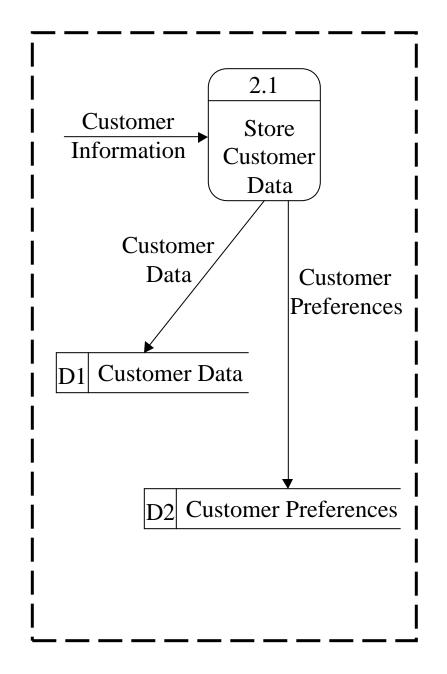


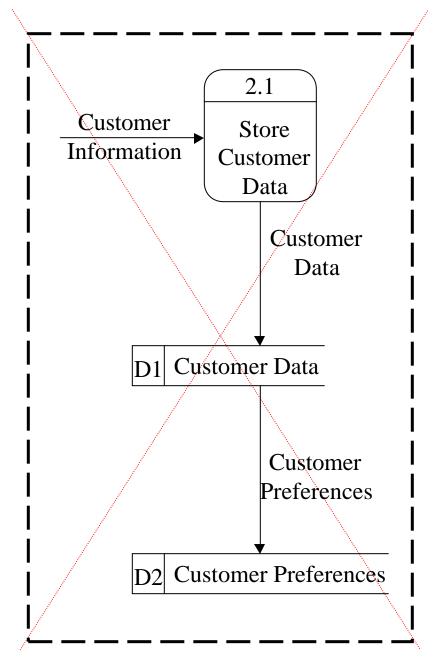


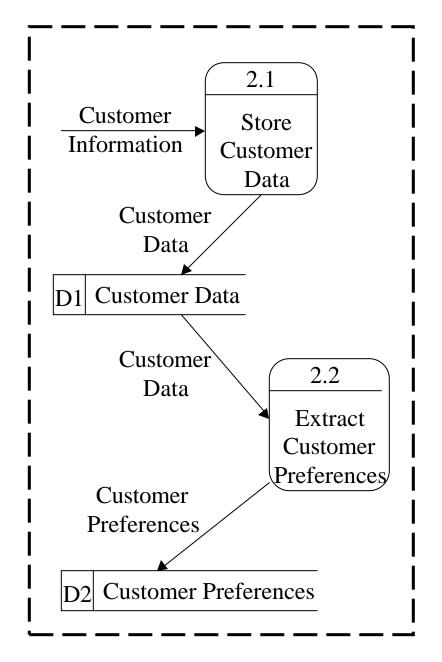
# Data Flow Diagramming Rules

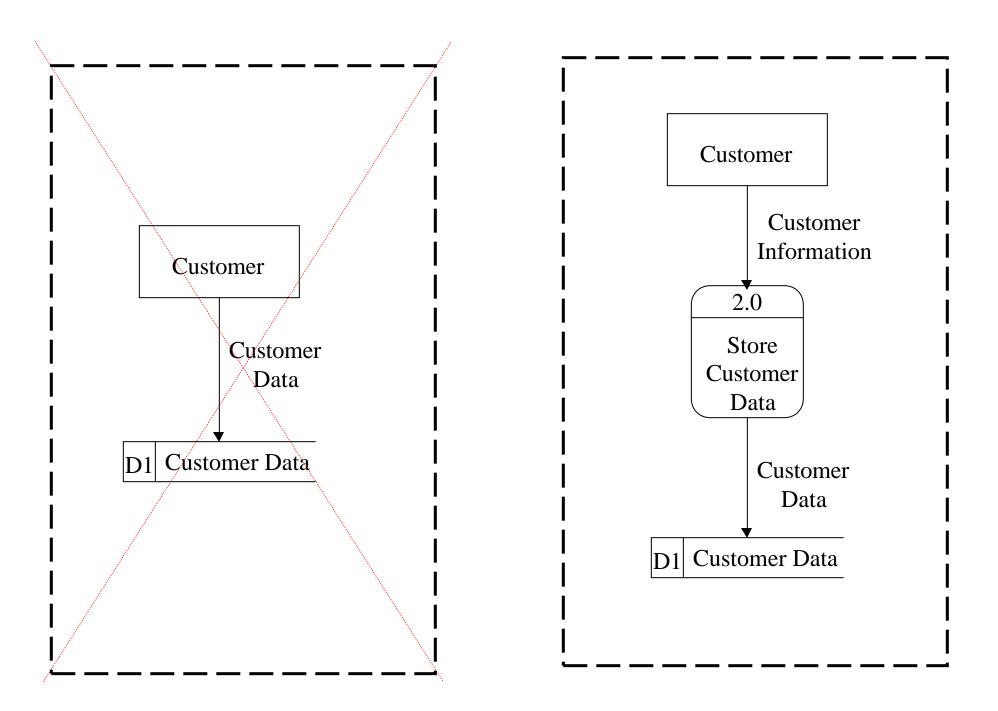
- Data stores and sources/sinks
  - no data flows between two data stores; must be a process in between
  - no data flows between a data store and a source or sink; must be a process in between
  - no data flows between two sources/sinks
    - such a data flow is not of interest, or
    - there is a process that moves that data

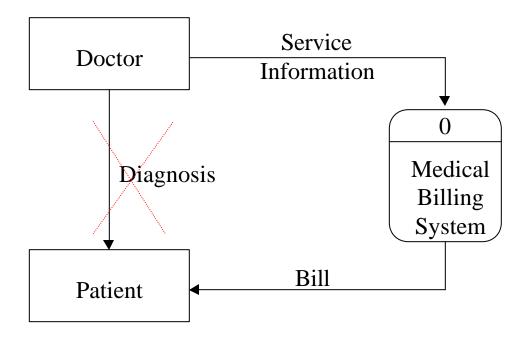








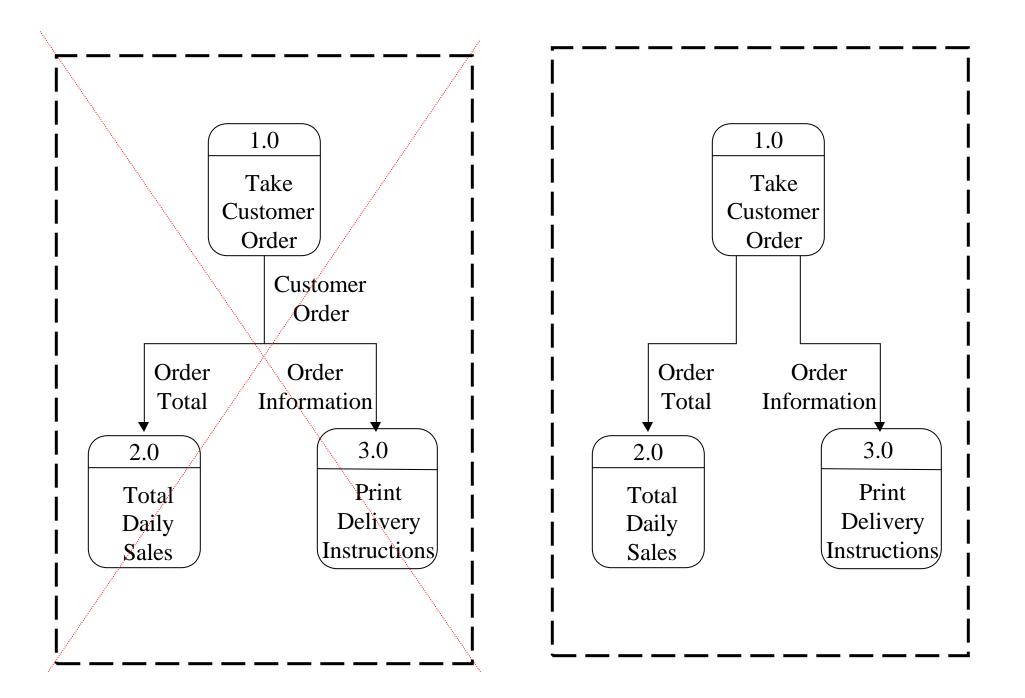


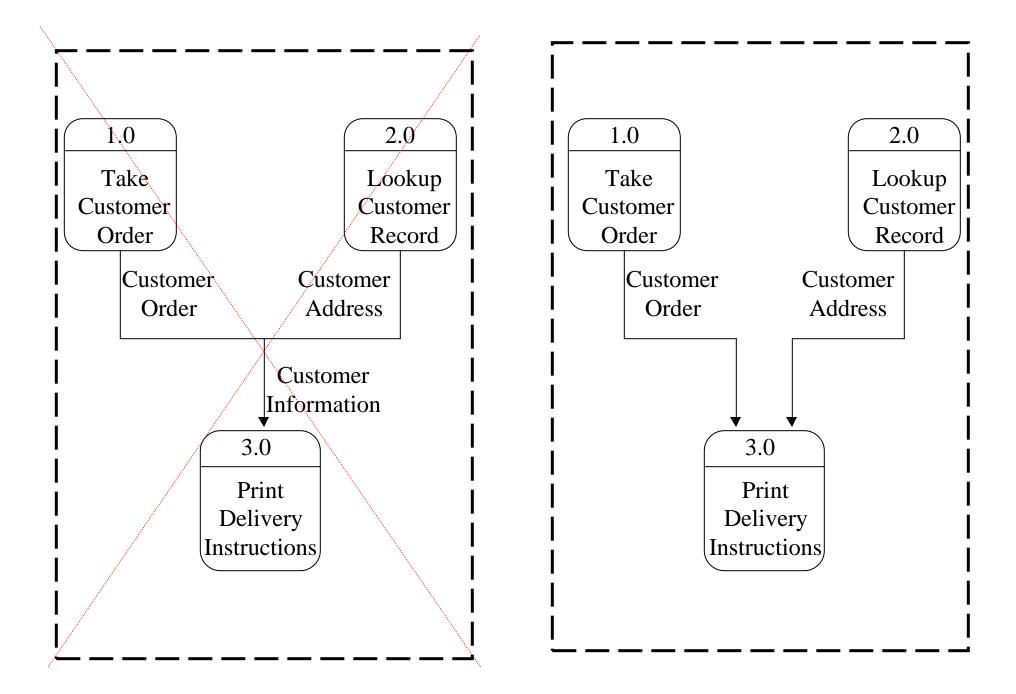


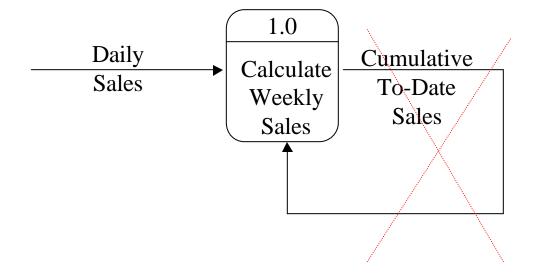
# Data Flow Diagramming Rules

### Data flows

- data flows are unidirectional
- a data flow may fork, delivering exactly the same data to two different destinations
- two data flows may join to form one only if the original two are exactly the same
- no recursive data flows
- data flows (and data stores and sources/sinks) are labelled with noun phrases



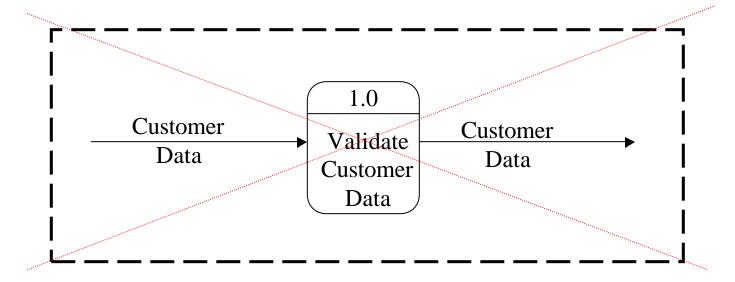


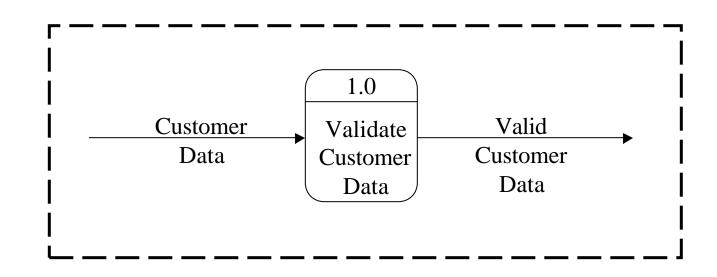


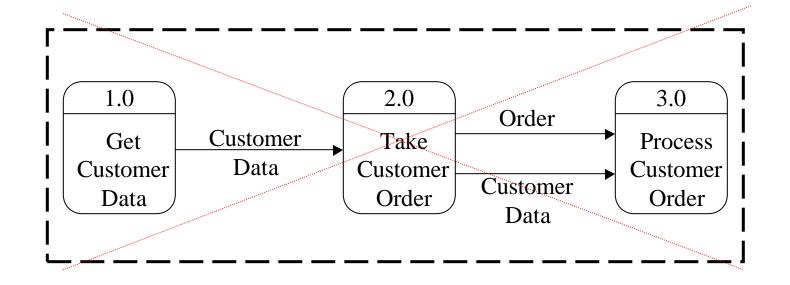
# Data Flow Diagramming Guidelines

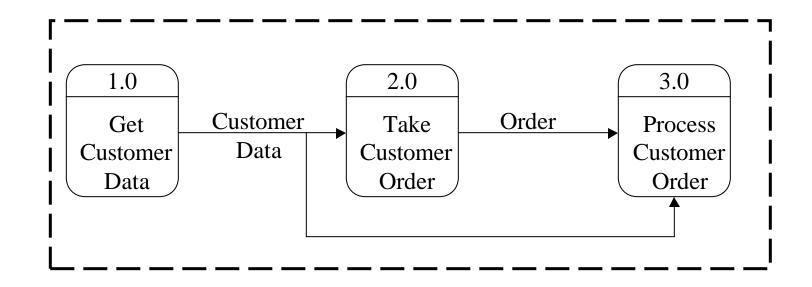
• The inputs to a process are different from the outputs

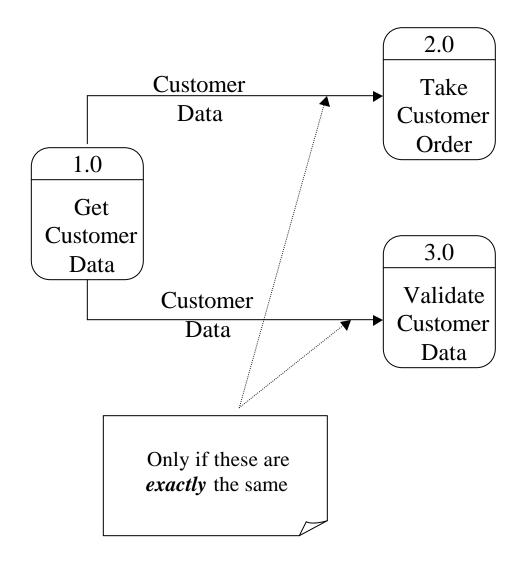
 Every object in a DFD has a unique name





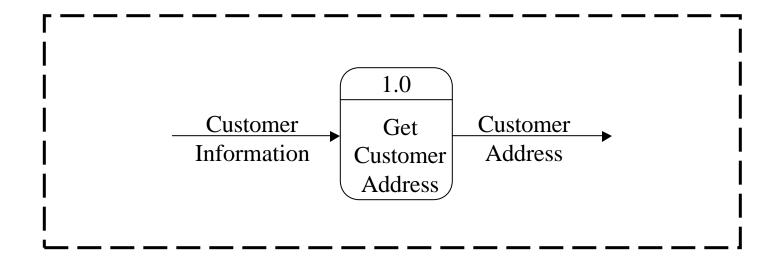


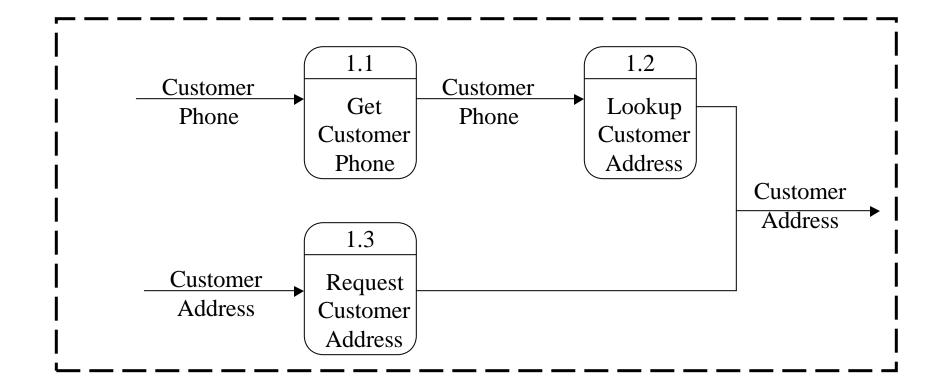


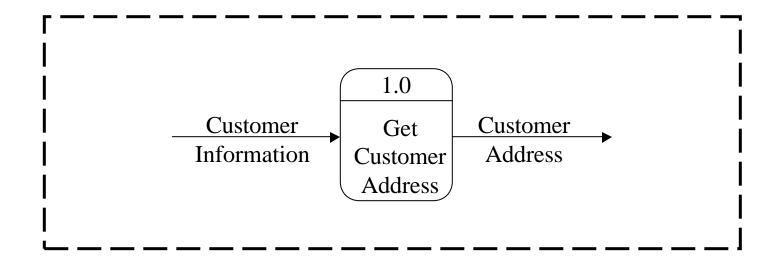


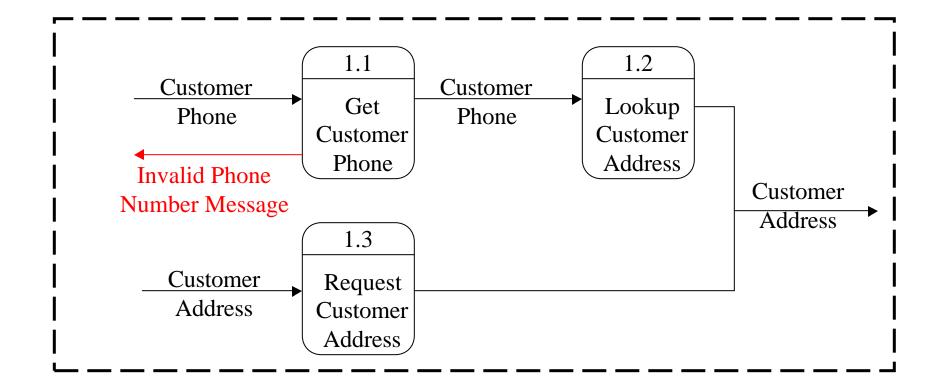
# Data Flow Diagramming Guidelines

- A data flow at one level may be decomposed at a lower level
- All data coming into and out of a process must be accounted for
- On low-level DFDs, new data flows can be added to represent exceptional situations



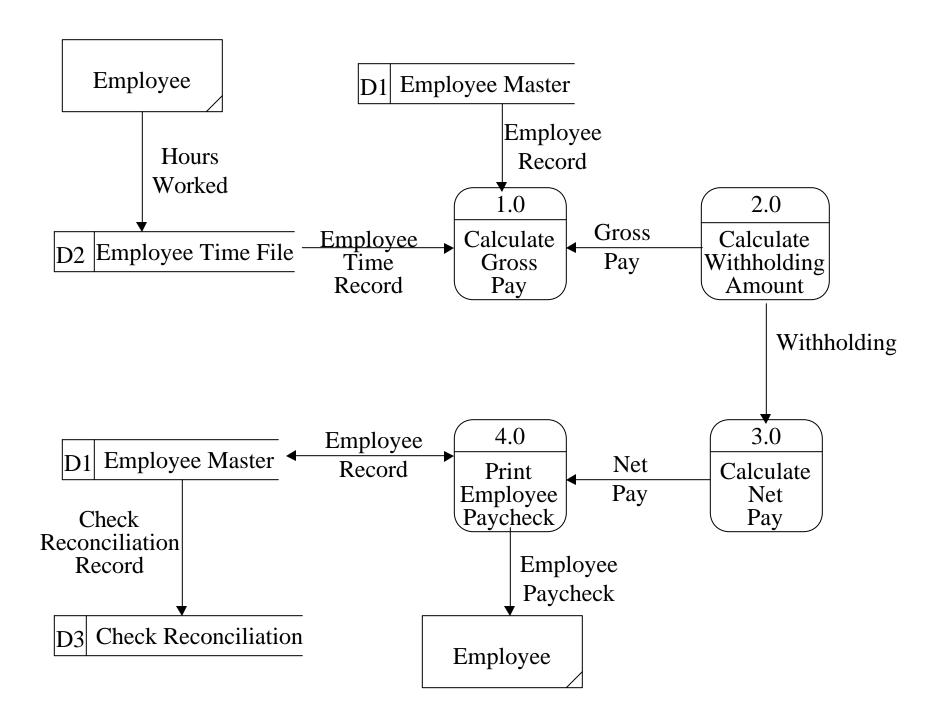


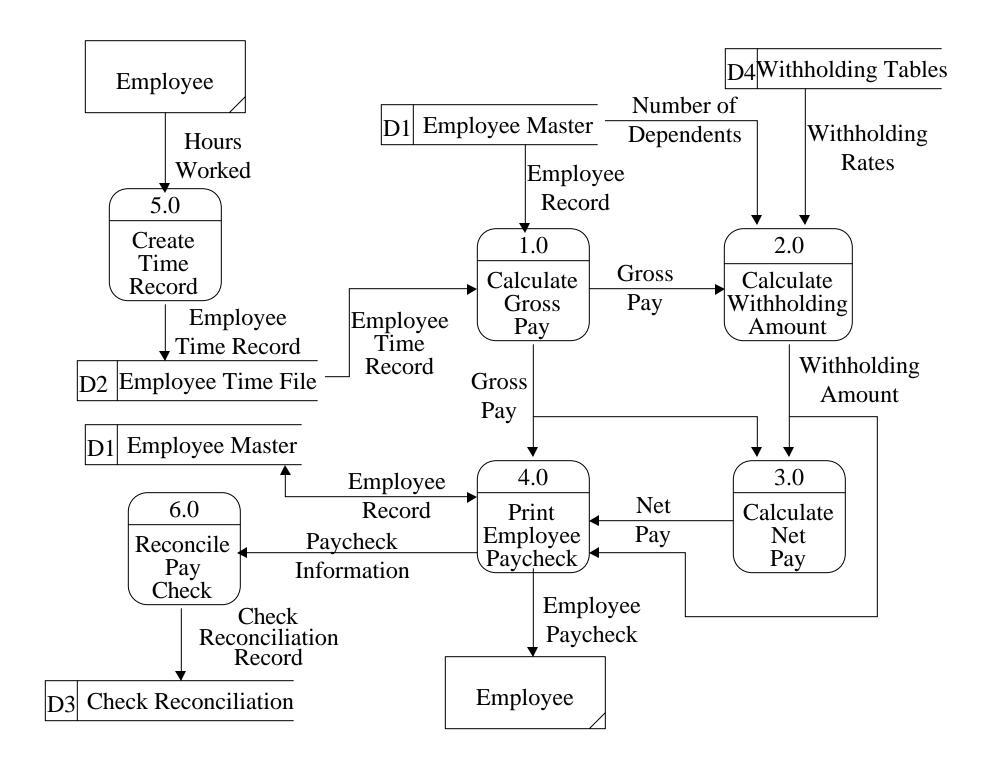




#### **Data Elements**

- Indivisible pieces of data
- Data flows and data stores are made up of data elements
- Like attributes on an ER diagram
- The data elements of a data flow flowing in or out of a data store must be a subset of the data elements in that data store





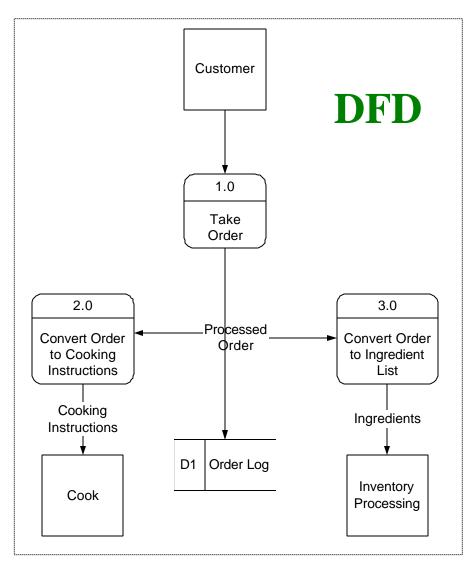
#### DFDs and ERDs

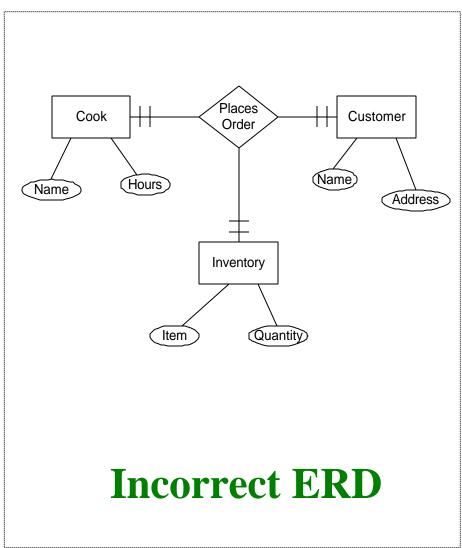
- DFDs and ERDs are both used to model systems, but they show two very different perspectives on the system
- A DFD shows what the system *does* as well as the *data* that the system manipulates
- An ERD shows **only** the *data* that the system manipulates.

### DFDs and ERDs (cont.)

- Entities on an ERD often (but not always) correspond to data stores on a DFD
- Attributes on an ERD usually correspond to data elements (listed in the data dictionary) that make up the data store and data flows on a DFD
- Relationships on an ERD do not correspond to processes on a DFD.
- Sources and sinks on a DFD usually do not show up as entities on an ERD

## Example DFD and ERD





### Example DFD and ERD

