# Analysis Modeling

Peter Lo

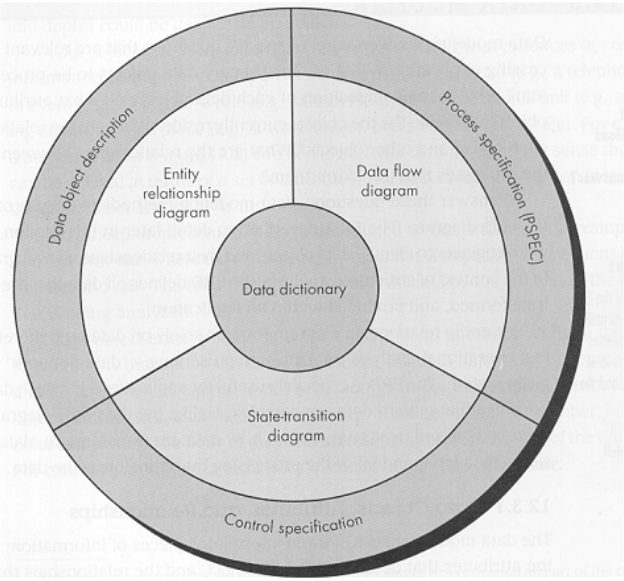


# An Introduction to Analysis Modeling

- Two dominant methods in analysis modeling.
  - Structured Analysis
    - This is a classic modeling method and is the concern of this chapter.
  - Object Oriented Analysis
    - Object Oriented Analysis is concerned with developing software engineering requirements and specifications that expressed as a system's object model (which is composed of a population of interacting objects), as opposed to the traditional data or functional views of systems.



# Basic Elements of the Analysis Model






# Basic Elements of the Analysis Model

- Data Dictionary
  - Repository that contains descriptions of all data objects consumed or produced by the software.
- Entity Relation Diagram (ERD)
  - Depicts relationships between data objects.
  - Data Object Description: Presents the attributes of each object in the ERD.

# Basic Elements of the Analysis Model

- Data Flow Diagram (DFD)
  - Presents how data is transformed as it moves through a system.
  - Process Specification: Presents a description of each function in the DFD.
- State Transition Diagram (STD)
  - Presents how the system behaves as a consequence of external events.
  - Control Specification: Contains information about the control aspects of the software



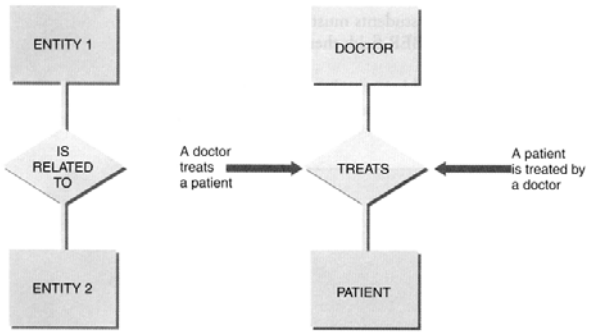
# Data Modeling

- A Data Model consists of three interrelated pieces of information:
  - **Data Object**
    - E.g. a report, an onscreen display, a file structure.
  - **Attributes**: define the properties of a data object.
    - E.g. Attributes for the car would be color, body type etc.
  - **Relationships:** Data objects are connected to one another in different ways.
    - E.g. Relationships between a book and a bookstore could be: “A bookstore orders books”, “A bookstore displays books”



# Entity-Relationship Diagrams

- An Entity-Relationship Diagram (ERD) is a graphical model of the information system that depicts the relationships among system entities.






# Three Main Types of Relationships

- One-to-One relationship (1:1)
- One-to-Many relationship (1:M)
- Many-to-Many relationship (M:N)

# One-to-One Relationship (1:1)

- It exists when exactly one of the second entity occurs for each instance of the first entity.




# One-to-Many Relationship (1:M)

- It exists when one occurrence of the first entity can be related to many occurrences of the second entity, but each occurrence of the second entity can be associated with only one occurrence of the first entity.

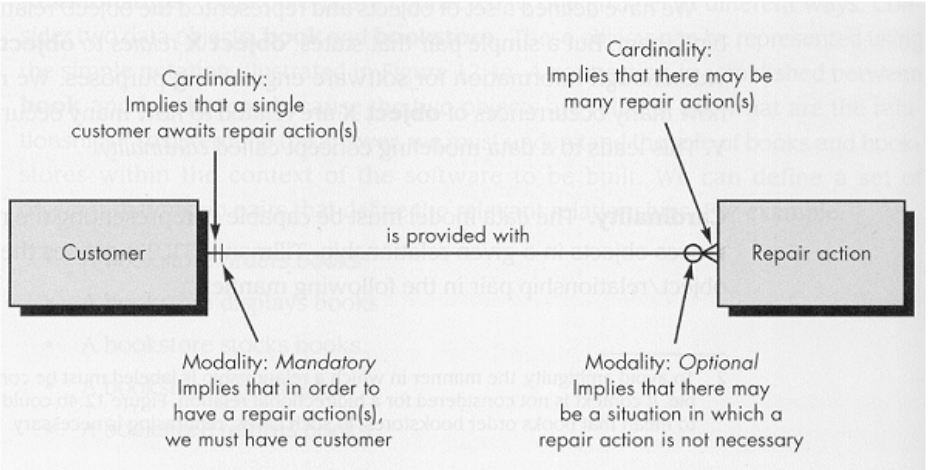



# Many-to-Many Relationship (M:N)

- It exists when one instance of the first entity can be related to many instances of the second entity, and one instance of the second entity can be related to many instances of the first entity.




# Cardinality and Modality

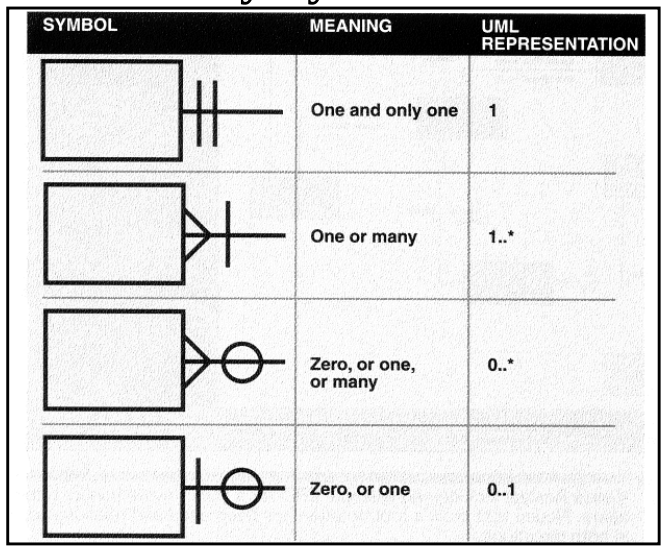


# Cardinality and Modality

- Cardinality
  - Specification of the number of occurrences in a data object that can be related to the number of occurrences of the other object.
  - This can be of a one to one, one to many, or many to many nature.
- Modality
  - Modality is equal to 0 if there is no explicit need for the relationship to occur, i.e. it is optional.
  - Modality is equal to 1 if the occurrence of the relationship is mandatory, i.e. not optional.



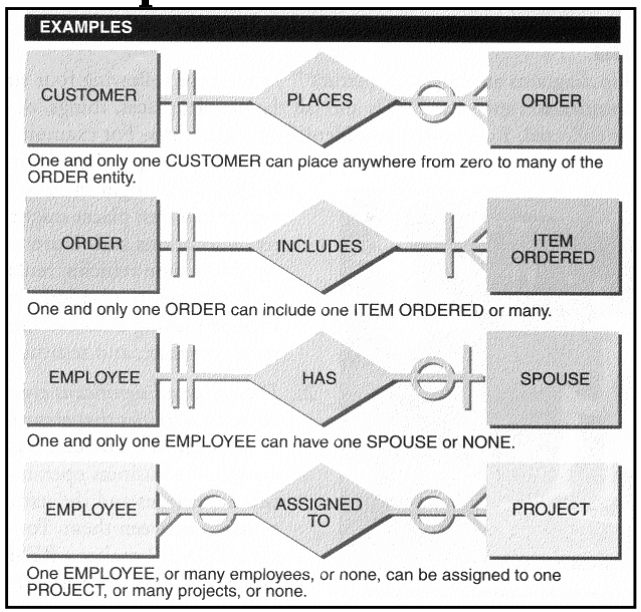
# Cardinality Symbols



| SYMBOL | MEANING | UML REPRESENTATION |
|---|---|---|
| | One and only one | 1 |
| | One or many | 1..* |
| | Zero, or one, or many | 0..* |
| | Zero, or one | 0..1 |



# Example






# Creating an Entity Relationship Diagram (ERD)

1. Identify the entities.
2. Determine all significant events, transactions, or activities that occur between two or more entities.
3. Analyze the nature of the interaction.
4. Draw the Entity Relationship Diagram.

# Data Flow Diagrams (DFD)

- A Data Flow Diagram (DFD) is a graphical tool to depict the flow of data through a system and the work or processing performed by that system.
  - It shows how data moves through an information system but does not show program logic or processing steps.
  - It only represent a logical model that shows what the system does, not how it does it.

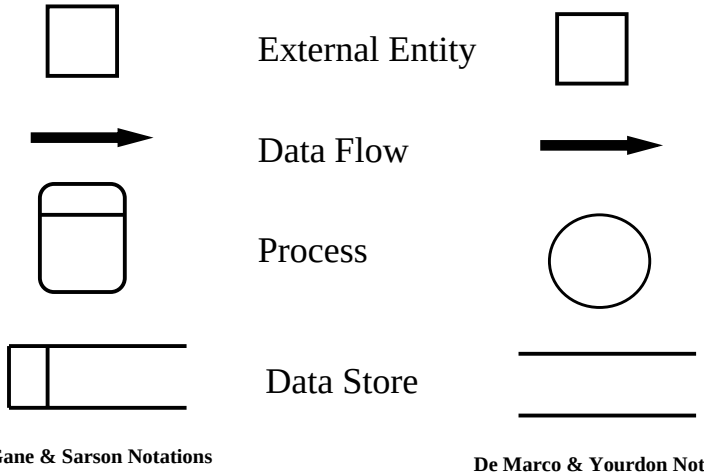


# Why DFD ?

- Language Description is subject to interpretation, it may omit crucial info.
- Graphical Description of the flow of data within an organization with DFD.

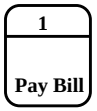


# DFD Symbols/Elements






# DFD Symbols – Process




- A Process is a work or action performed on input data flow to produce an output data flow
- Use a verb to label the action performed by the process (not the name of person or department who does it as in physical DFD)
- A Process must have at least one input data flow and at least one output data flow.
- Examples: Apply rent payment, verify order, pay bill.

# DFD Symbols – Data Flows

DELIVERY SLIP →

- A Data Flow represents a movement of data (information) among processes or data stores
- A Data Flow does not represent a document or a physical good: it represents the exchange of information in the document or about the good
- A Data Flow represents an input of data to a process, or the output of data from a process.
  - A data flow may also be used to represent the creation, reading, deletion, or updating of data in a file or database (data store).
  - A composite data flow (packet) is a data flow that consists of other data flows.
- Examples: Deposit, invoice payment, delivery slip.



# DFD Symbols – Data Store

| D1 | Accounts Receivable |
|---|---|

- A Data Store is a storage of data: it contains information
- Physical storage is immaterial: it can be a filing cabinet, book, computer file
- A data store is an inventory of data.
  - A data store is "data at rest" compared to a data flow that is "data in motion."
  - Almost always one of the following: Persons (or groups of persons), Places, Objects, Events (about which data is captured), Concepts (about which data is important),
  - One can identify data stores with REAL framework
  - Data stores depicted on a DFD store all instances of data entities (depicted on an ERD)
- Examples: Accounts receivable.



# DFD Symbols – External Entity

SUPPLIER

- An External Entity is a provider (source) or receiver (sink) of data and info of the system
- An External Entity is not part of the system: the externality depends on how the system is defined
- An external entity (agent) defines a person, organization unit, or other organization that lies outside of the scope of the project but that interacts with the system being studied.
  - External agents define the "boundary" or scope of a system being modeled.
  - As scope changes, external agents can become processes, and vice versa.
  - Almost always one of the following: Office, department, division inside the business but outside the system scope, an external organization or agency, another business or another information system, one of system's end-users or managers
- Examples: Customer, student, supplier.



# Context Diagram

- An overview of an organizational system that shows the system boundaries, external entities that interact with the system, and the major information flows between the entities and the system.
- Single process (labeled "0") represents the entire system. The sources/sinks represent its environmental boundaries.
- No data stores appear on context diagram.

# Strategies for Developing DFD

- Top-down Model
  - First create the context diagram, then diagram 1, then all the child diagrams for diagram 1, and so on.
- Bottom-up strategy
  - First identify all functional primitives, data stores, external entities, and data flows. Then group processes with other related symbols to develop the lowest-level diagrams. Next, group those diagrams in a logical way to form the next higher level. Continue to work upwards until you reach Context Level.



# DFD Building Procedure

- Step 1: Context Diagram
  - Identify the system and its boundaries (the context)
  - Identify external entities (providers, receivers of system info)
  - Identify external data flows (input, output)
- Step 2: Level-1 DFD
  - Identify what is being done between each input and its corresponding output
  - Identify the processes
  - Identify external data flows between external entities and processes
  - Identify internal data flows between processes and data stores
- Step 3: Level-2 DFD's
  - Sub-processes (primitive processes) of Level-1 processes

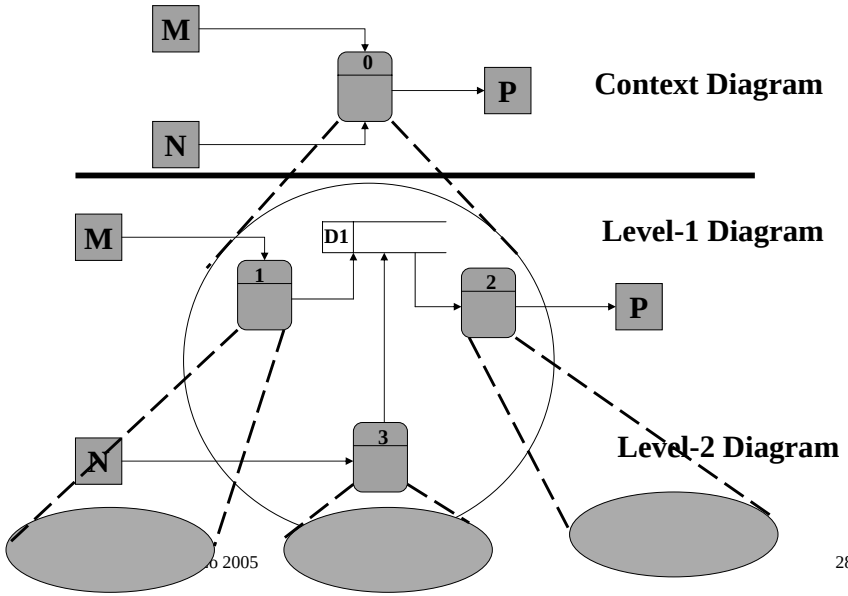


# Lower-Level Diagrams

- When lower-level diagrams are needed to show detail, it is essential that they be leveled and balanced.
- **leveling** is the process of drawing a series of increasingly detailed diagrams, until the desired degree of detail is reached.
- **Balancing** maintains consistency among the entire series of diagrams, including input and output data flows, data definition, and process descriptions.
- A balanced set of DFD preserves the input and output data flows of the parent on the child DFD.



# Decomposition of Context Diagram



# Conventions for DFD

- Conventions, or rules you should use when constructing DFD:
  - Each context diagram must fit on one page.
  - The process name in the context diagram should be the name of the information system.
  - Use unique names within each set of symbols.
  - Do not cross lines.
  - Use a unique reference number for each process symbol.

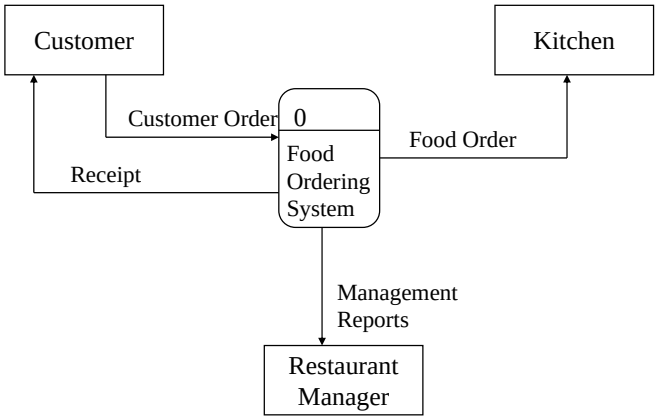


# Example

- Hoosier Burger uses an Information System to take customer orders, send the orders to the kitchen, monitor goods sold and inventory, and generate reports for management.
- The context DFD for the system is given in next page.
- The next step for the analyst to think about which processes are represented by the single process in the context diagram. 4 separate processes identified. These processes represent the major functions of the system.
- Level-1 diagram represents the primary individual processes in the system at the highest possible level. Each process has a number which ends in .0 (corresponding to the level number of the DFD).



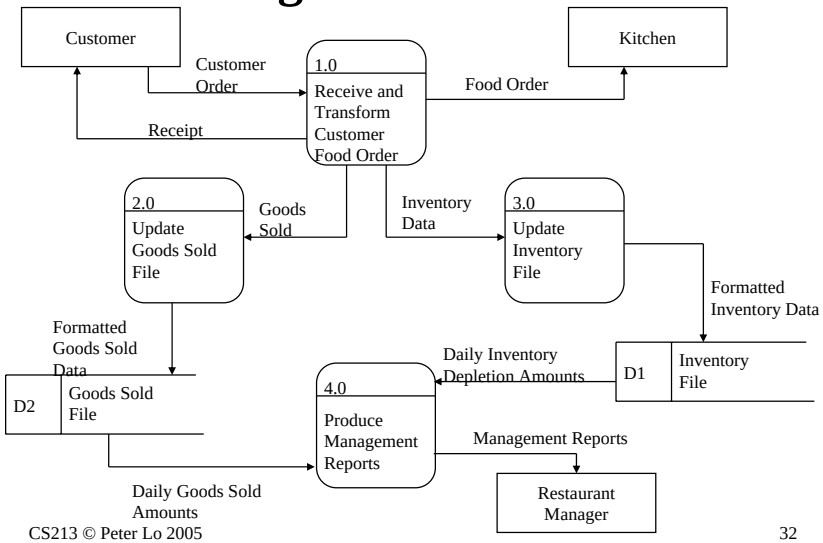
# Context Diagram



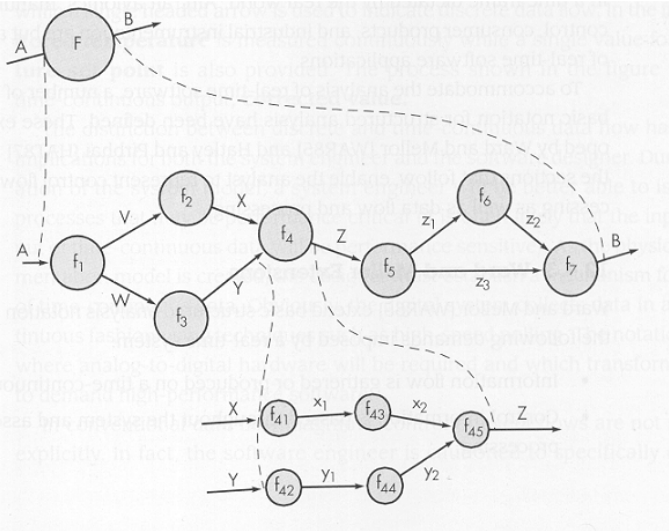

Context DFD of Hoosier Burger's food ordering system



# Level 1 Diagram



# Data Flow Diagrams (Example)



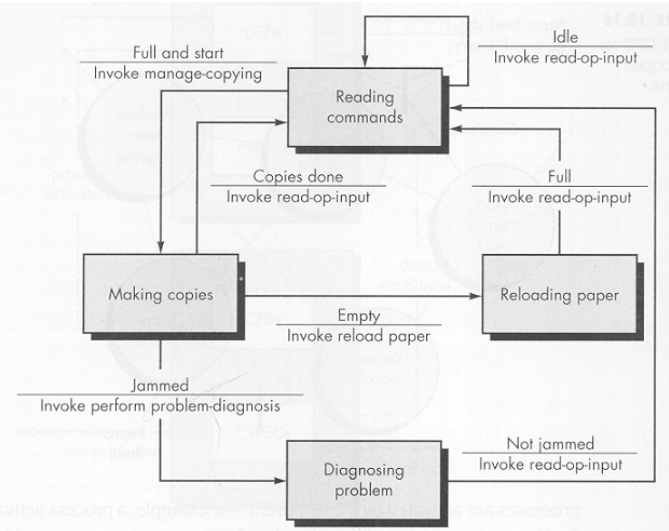



# State Transition Diagram (STD)

- A State Transition Diagram shows how an object changes from one state to another, depending on events that affect the object.
- It represents the behavior of a system by depicting its states and the events that cause the system to change state.
- A state is any observable mode of behavior.
  - Examples of "states" in photocopier software could be monitoring state, alarm state etc.

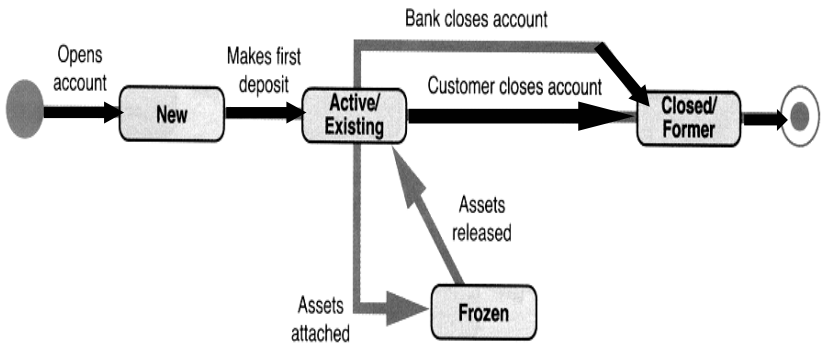


# State Transition Diagram (STD)






# State Transition Diagram (STD)



# Data Dictionary

- An organized listing of all data elements that are appropriate to the system, with precise, rigorous definitions so that both user and system analyst will have a common understanding of inputs, outputs, components of (data) stores and even intermediate calculations



# Data Dictionary

| | |
|---|---|
| 3. Data Element : | Leader ID |
| Description : | Uniquely identifies the Leader in a sales team |
| Aliases : | Leader_ID |
| Type : | Alphnumeric |
| Format : | X999 |
| Length : | 4 |
| Range : | From : E001 To : E999 |
| Typical Value : | - |
| Specific Values : | - |
| Other characteristics : | Base, continuous |

| | |
|---|---|
| 4. Data Element : | User ID |
| Description : | Uniquely identifies an Employee |
| Aliases : | User_ID |
| Type : | Alphanumeric |
| Format : | X999 |
| Length : | 4 |
| Range : | From : E001 To : E999 |
| Typical Value : | - |
| Specific Values : | - |
| Other characteristics : | Base, continuous |

# Appendix:

DFD Rules

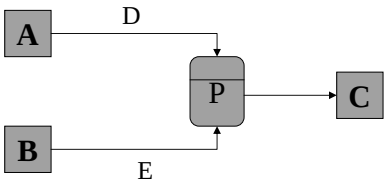


# Data Flow Diagramming Rules

- There are two DFD guidelines that apply most of the time:
  - The inputs to a process are different from the outputs of that process.
  - Objects on a DFD have unique names.
- The following rules allow you (or a CASE tool) to evaluate DFD for correctness.

# Data Flow Diagramming Rules

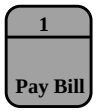
- **Rule 1**: Unique label for each symbol to avoid confusion






# Data Flow Diagramming Rules

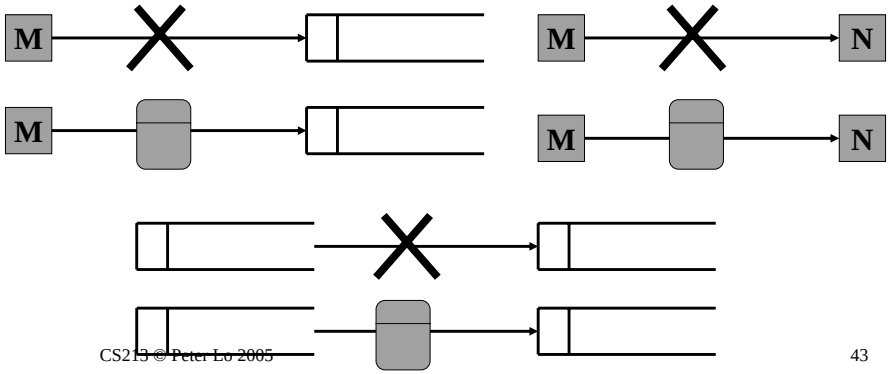
- **Rule 2**: Use an action VERB to label a process






# Data Flow Diagramming Rules

- **Rule 3**: Must be one process associated with each data flow.



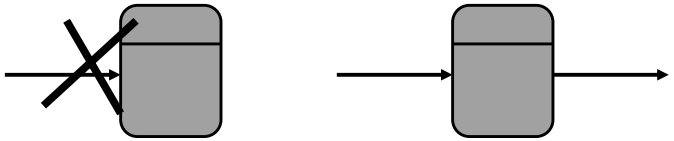



# Data Flow Diagramming Rules

- **Rule 4**: Shaded corner must appear in ALL occurrences of a duplicated symbol in a same diagram.

# Data Flow Diagramming Rules

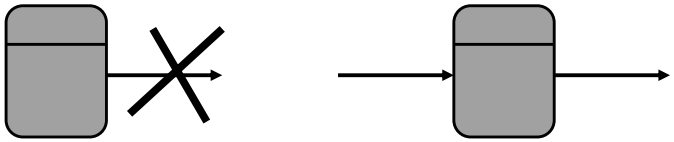
- **Rule 5**: No process without output data flow.





# Data Flow Diagramming Rules

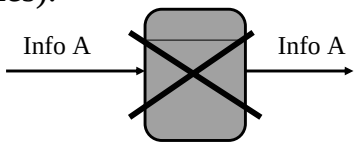
- **Rule 6**: No process without input data flow.





# Data Flow Diagramming Rules

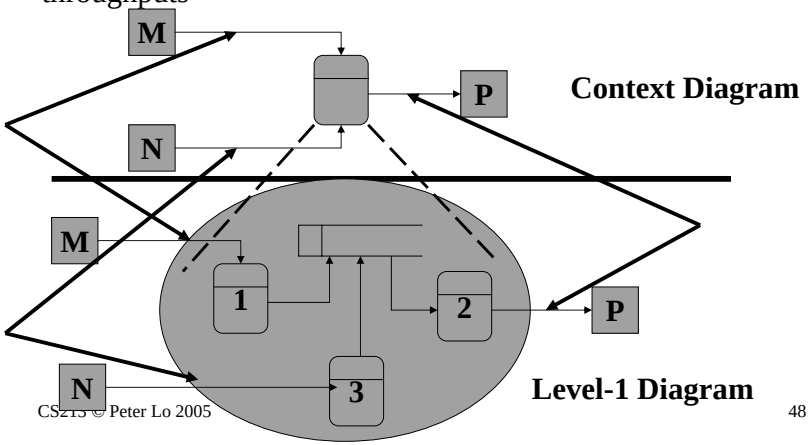
- **Rule 7**: No need for routing (without transforming) a data flow with a process (non value-added activities).





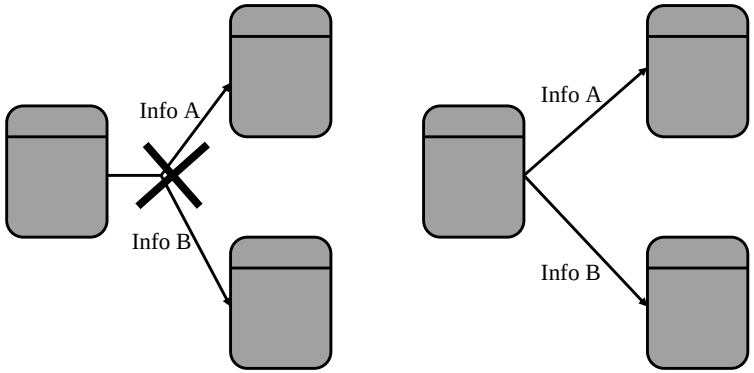

# Data Flow Diagramming Rules

- **Rule 8**: Identical input, output data flows for parent and child processes (but the child processes can have their own throughputs



# Data Flow Diagramming Rules

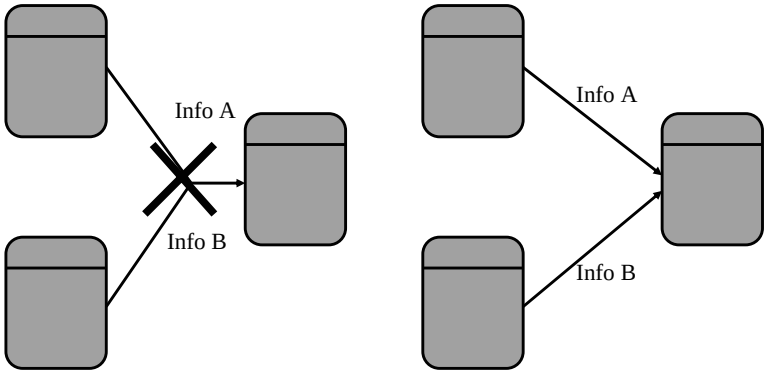
- **Rule 9**: Data flows cannot split






# Data Flow Diagramming Rules

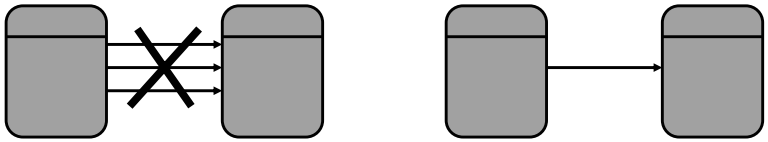
- **Rule 10**: Data flows cannot combine






# Data Flow Diagramming Rules

- **Rule 11**: A data packet can combine many data elements being transmitted at the same time to the same destination





# Data Flow Diagramming Rules

- **Rule 12**: Double-headed arrows are forbidden [in-flow (update) and out-flow (extract info) of a data store are different]