CSC 342

Semester I: 1425-1426H (2004-2005 G)

Software Engineering

Systems Analysis:

Requirements Structuring

‘Context & DFDs ….’

Instructor:

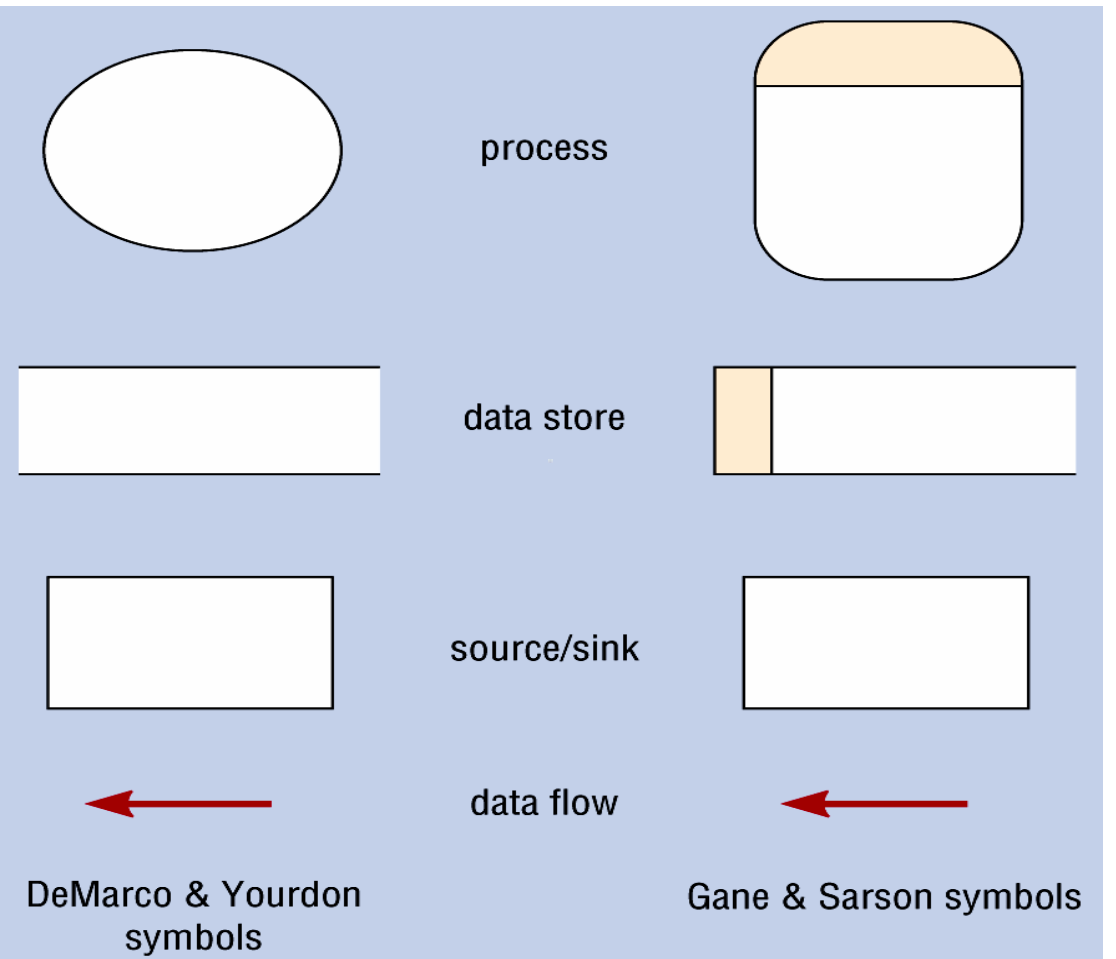
Dr. Ghazy Assassa

Objectives

- To introduce requirements structuring that follows Facts Finding Techniques:

- Introduce the following:
  - System context diagram
  - Data Flow Diagrams DFD's
  - Entity Relationship diagram: E-R diagram
  - Decision tables & Decision trees
  - Natural Language

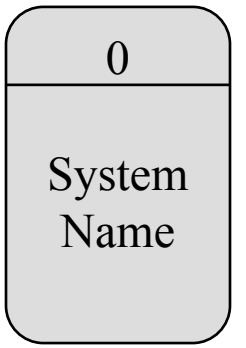
Comparison of DeMarco & Yourdan and Gane & Sarson DFD Symbol Sets



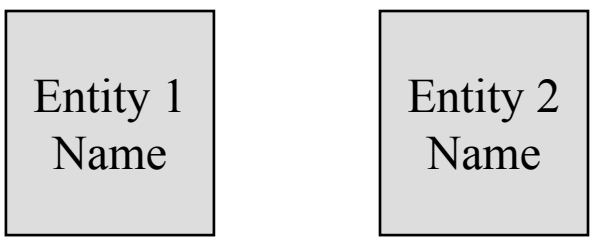
Components of a Context Diagram

3 Symbols:

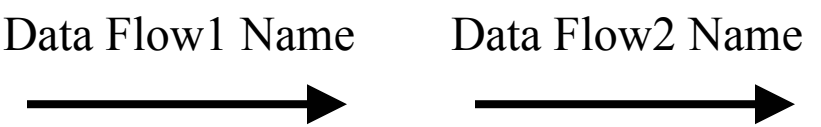
- Process (single process)



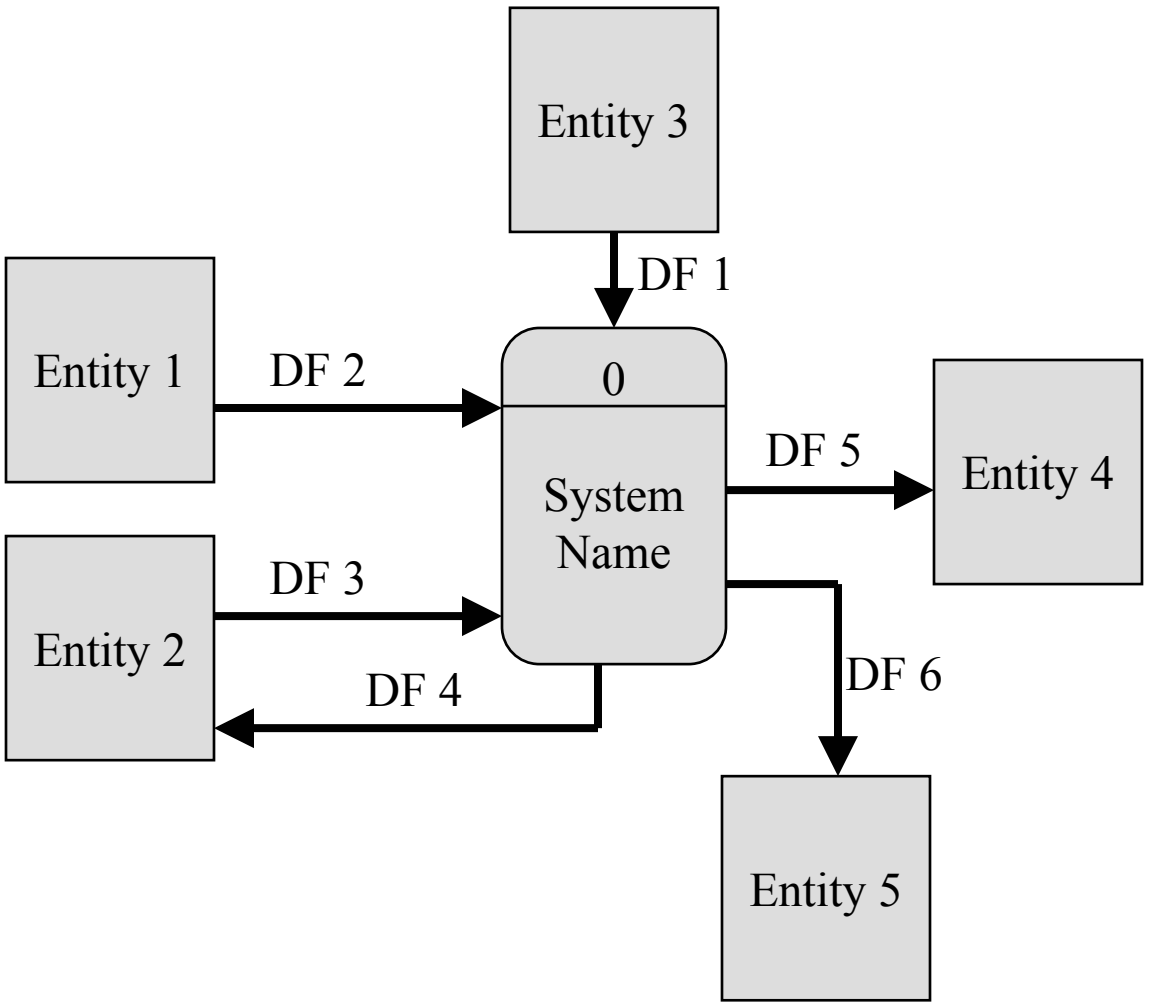
- Entity (1 or many)



- Data Flow (many)



Context Diagram "Sys Interface"



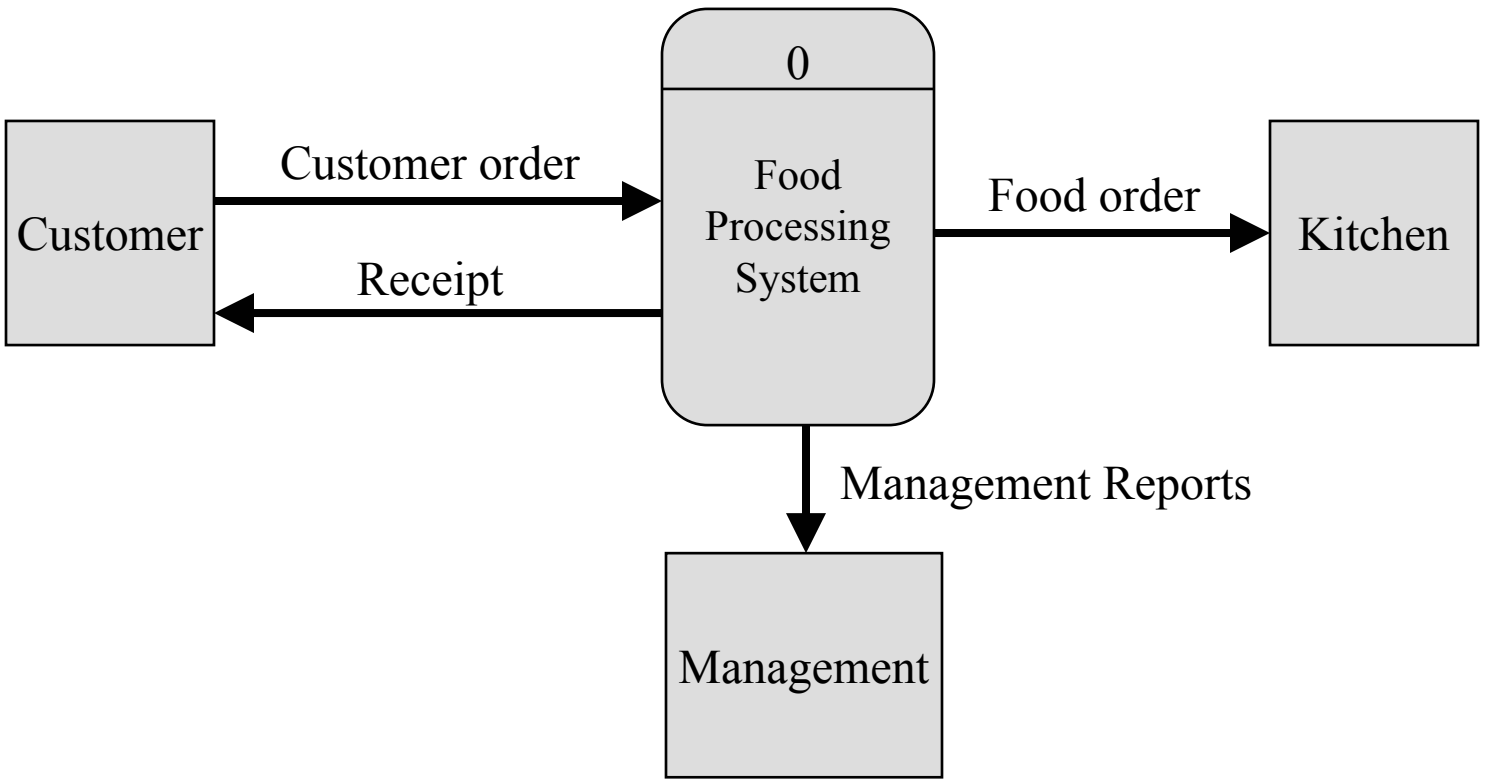
Context Diagram

- System Interface
- Bird's view of the system

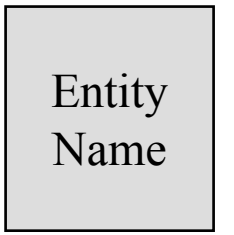
- Context Diagram

  - Shows the scope of the systemm
  - system boundaries,
  - external entities that interact with the system
  - and the major information flows between the entities and the system

Context Diagram "Sys Interface" Example: Food Ordering System



Context Diagram: Entity

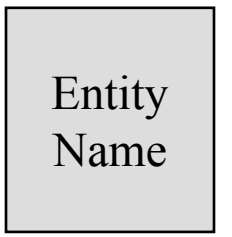


An Entity "External Entity" may be:

- A person interacting with the system (ex: Student , Customer).
  - The person may be inside or outside the business unit under study
- Another organization sending/receiving info (ex: Supplier, Department)
- Another Information system interfacing with the system under study (ex: Bank, GOSI)

Context Diagram: Entity (cont.)

- External entity to the system



- Source entity: provides data flows to the system
- Sink entity: receives data flows from the system
- Considered as a black box:
  - We are not interested in what an entity will do with information or how it will do it

Context Diagram: Data Flow "DF"

Data Flow "DF"

- Data in motion.
- Ex:
  - Data on a an invoice
  - Data on a payroll check

- DF may include a structured group of data that move together, i.e. as a data structure.
  - data structure details to be specified separately (not on the DFD)
  - EX: Invoice data = (Invoice date, customer name, sales person name, list of {item name, quantity, unit price}, invoice total, discount, invoice net amount)

Context Diagram: Data Flow "DF"

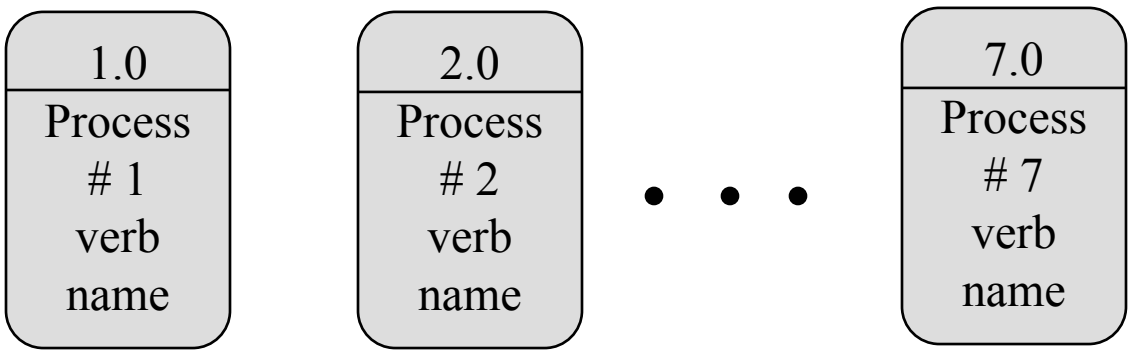
DF may be:

Data Flow Name

- Paper DF for manual systems
- Electronic format DF for automated systems

Data Flow Diagrams DFD :
4 Symbols

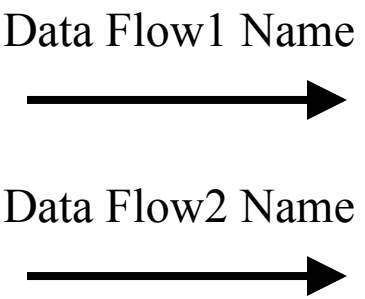
- Process (*many* processes)



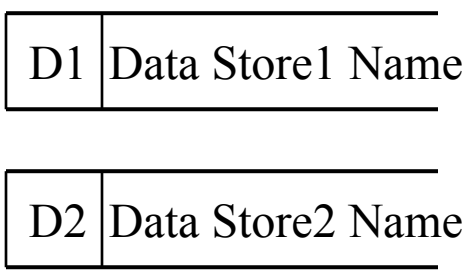
- Entity (1 or many)

Data Flow Diagrams: Components of a DFD

- Data Flow (many)

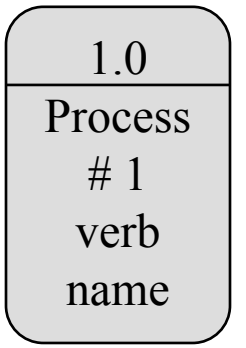


- Data Store (1 or many)



DFD: Processes

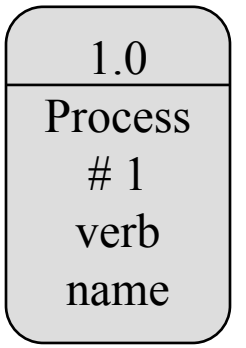
- A Process may be:
  - manual or computerised



- A Process:
  - Represents actions on data (verb)
  - Receives DF for processing
  - Sends DF after processing
  - May:
    - Transform data (calculations)
    - Store data
    - Distribute data

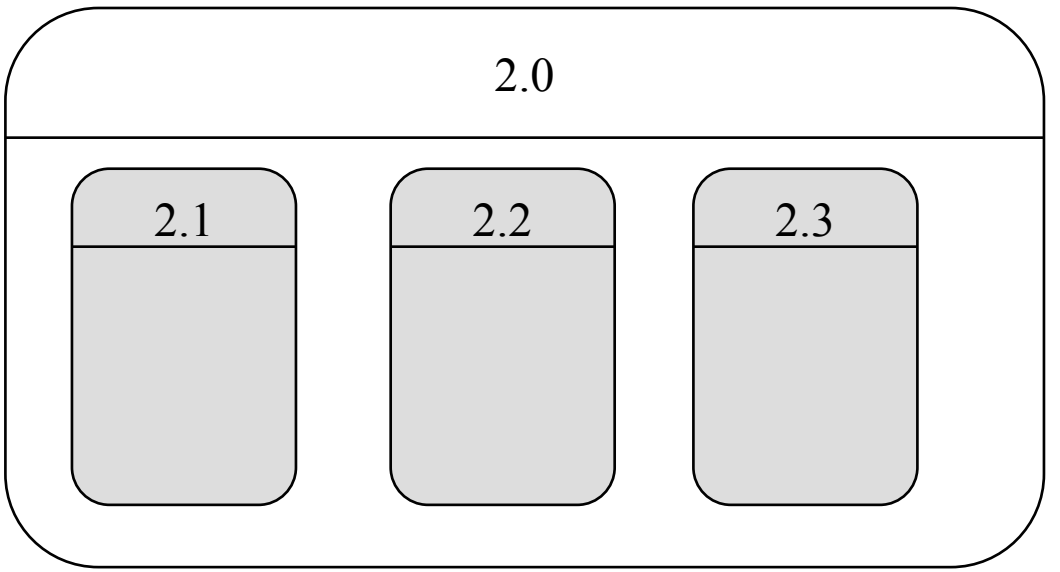
DFD: Process Explosion / Decomposition / Partitioning

- A process:
  - Has a number (ID)
  - Has a verb describing the action on data

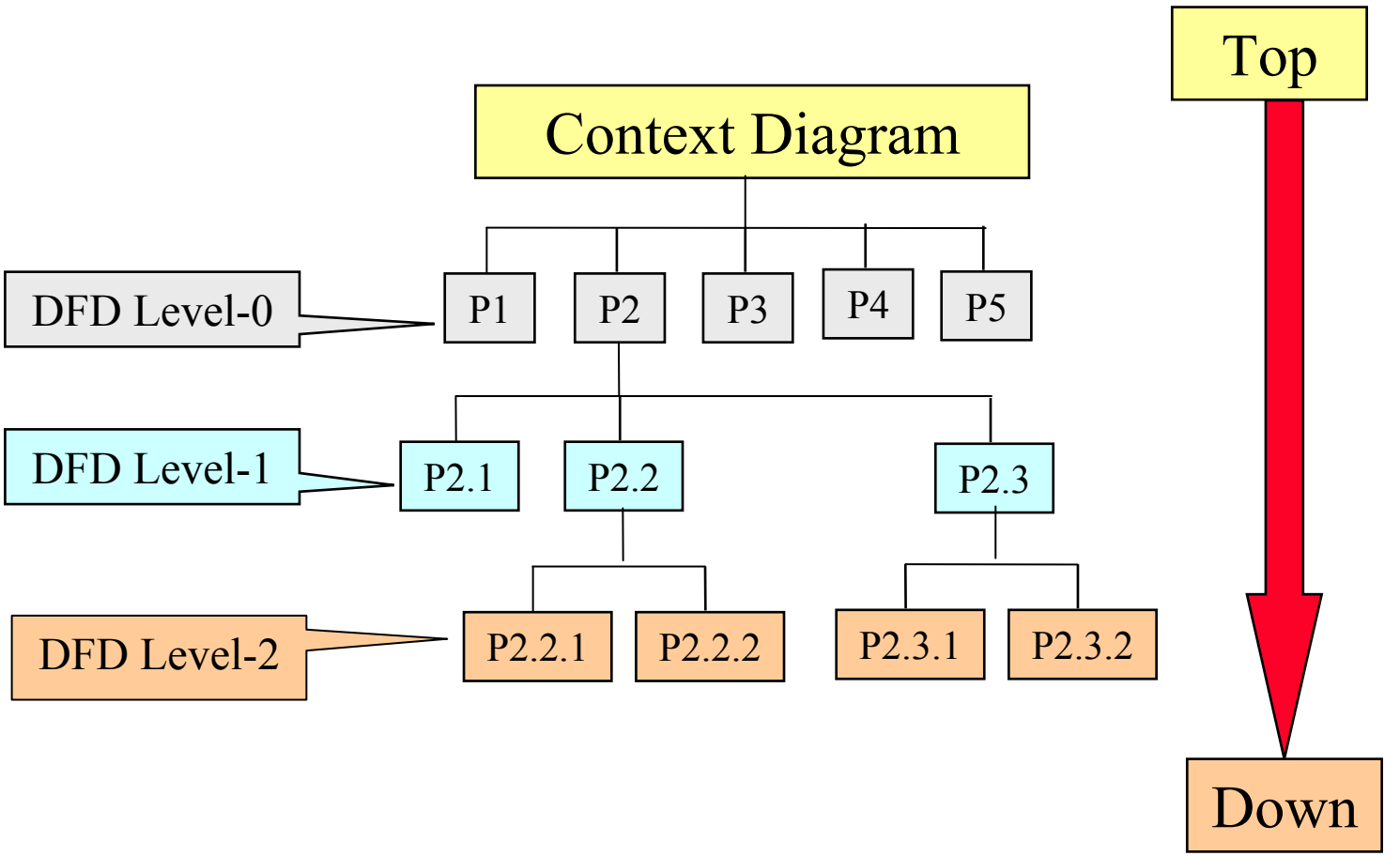


- A complex process may be exploded (de-composed) in many sub-processes "process hierarchy"
- Ex:
  - Process 2.0 may be exploded into 3 sub-processes: 2.1, 2.2, 2.3
  - Process 2.2 may be exploded into 4 and sub-processes: 2.2.1, 2.2.2, 2.2.3, 2.2.4
  - And so on, till getting simple processes as leaves of the tree

DFD: Process Explosion / Decomposition



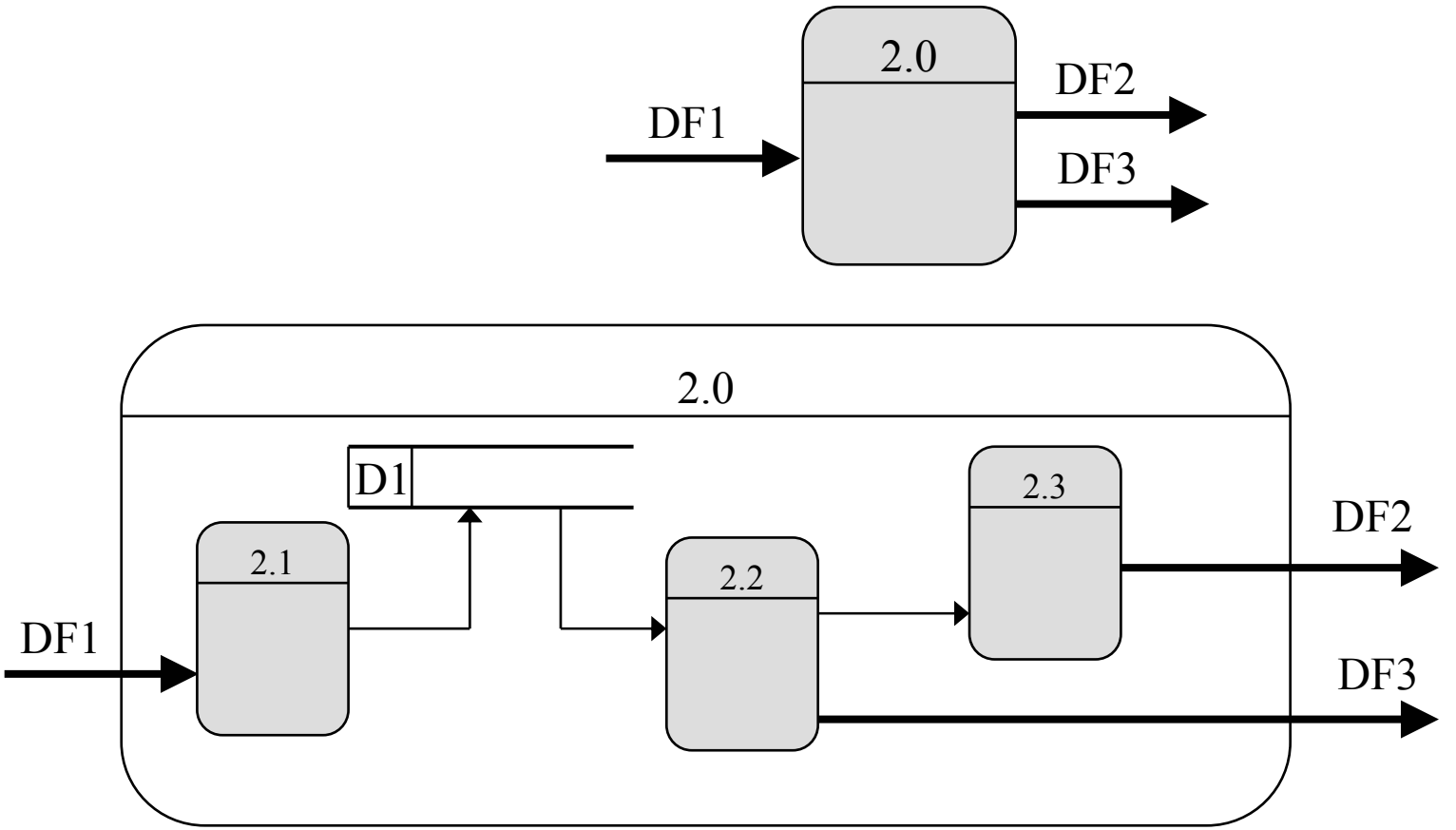
DFDs Decomposition Tree



Balancing DFDs

- When a process is decomposed to a lower level you must conserve inputs and outputs of this process at both levels of decomposition

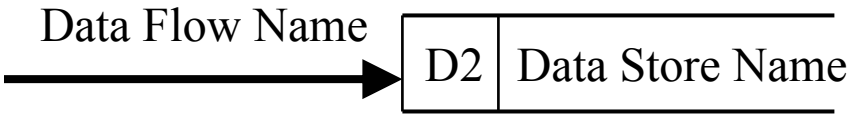
Balancing DFDs



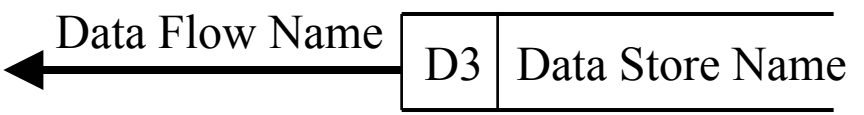
DFD: Data Store

- Data at rest in
  - Files in folder (paper)
  - Computer files (one or more)
  - Computer database

- Data may be written to a data store "update"



- Data may be read from a data store "query"



DFD: Data Store (cont.)

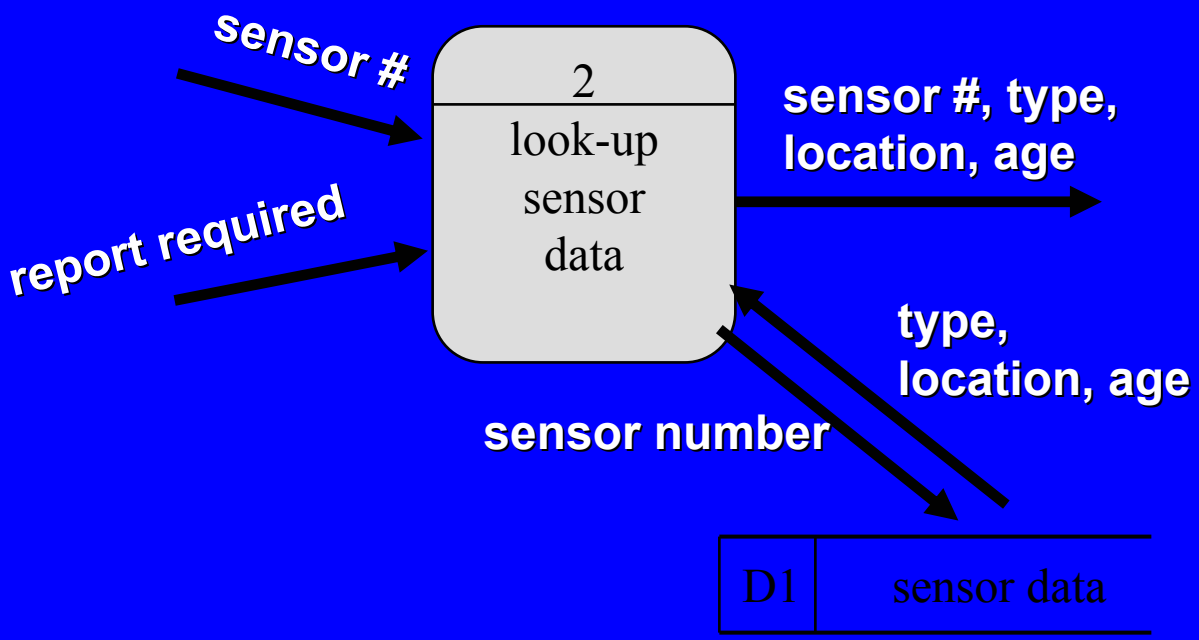
Examples of Data stores:

- Customer info: customer_name, ID, address, fax, etc..
- Student info: student_name, ID, birth_date, address, major, GPA, etc..

- For a relational databases, data stores will provide the normalised relations

Data Store

Data is often stored for later use.



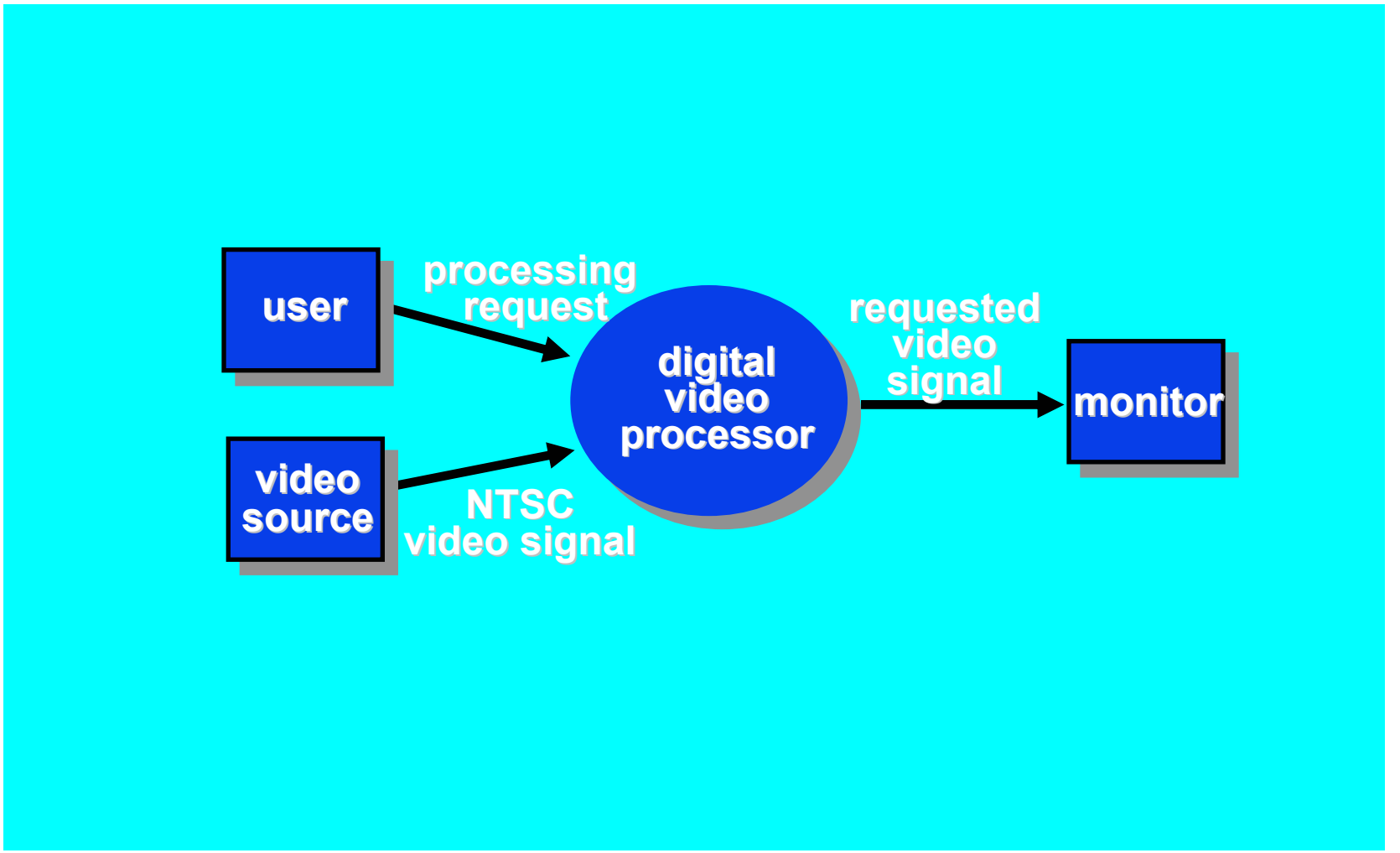
Data Flow Diagramming: Guidelines

- all icons must be labeled with meaningful names
- the DFD evolves through a number of levels of detail
- always begin with a context level diagram
- always show external entities in context level diagram
- always label data flow arrows
- do not represent process logic
- process logic is described separately using structured English, decision tables, decision trees

How to construct a context diagram

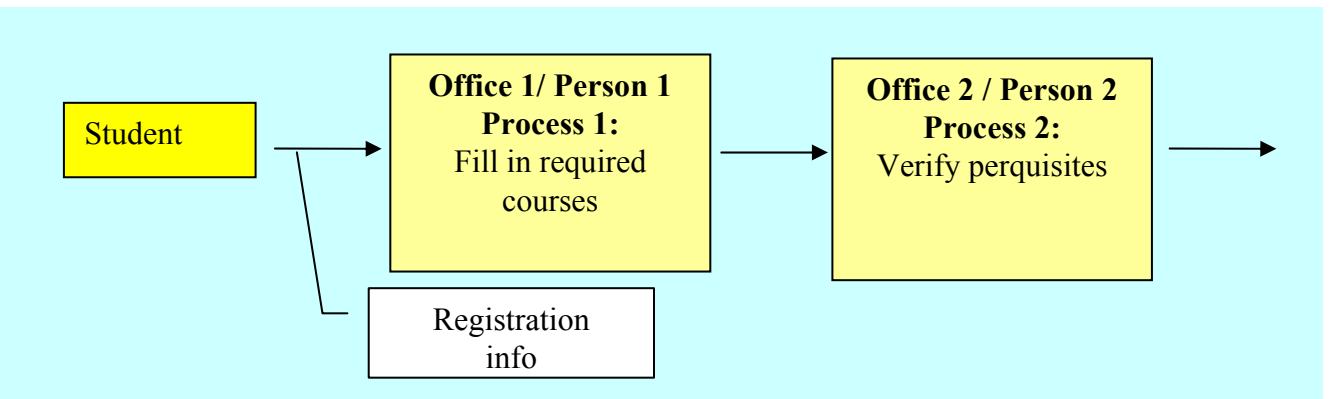
- Determine external entities to the system
  - Source entities that provide data to the system
  - Sink entities that receive data from the system
- Determine data flow (single, or group of data)
  - from external source entities
  - To external sink entities
- Draw system context diagram

Context Diagram: Example



Confusion: an entity or a process ?

- Do not be confused by: Is some thing an entity or a process?
- This may occur when
  - some data are processed in one place **(office 1)**
  - and the processed data is moved to another place **(office 2)** for additional processing.



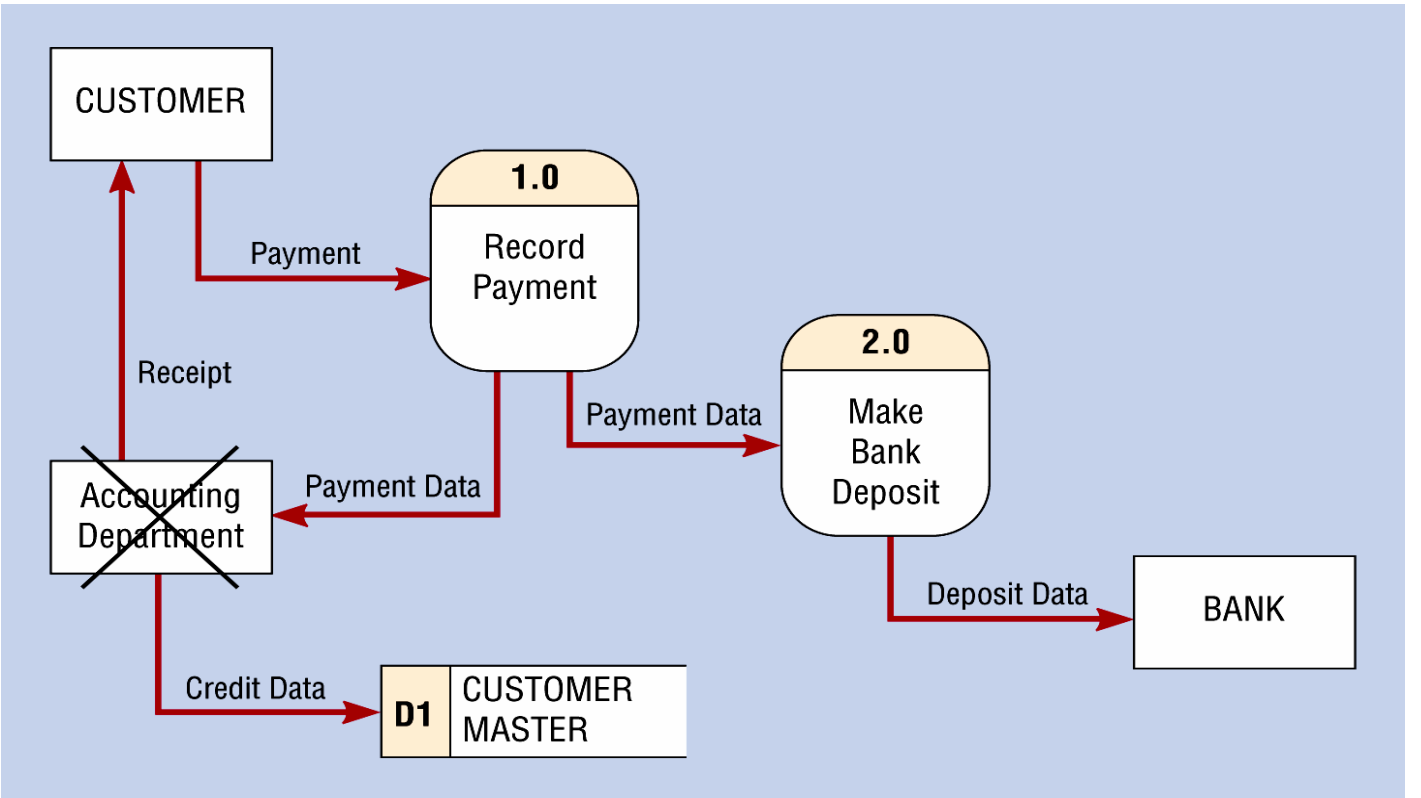
Confusion: an entity or a process ?

In DFDs:

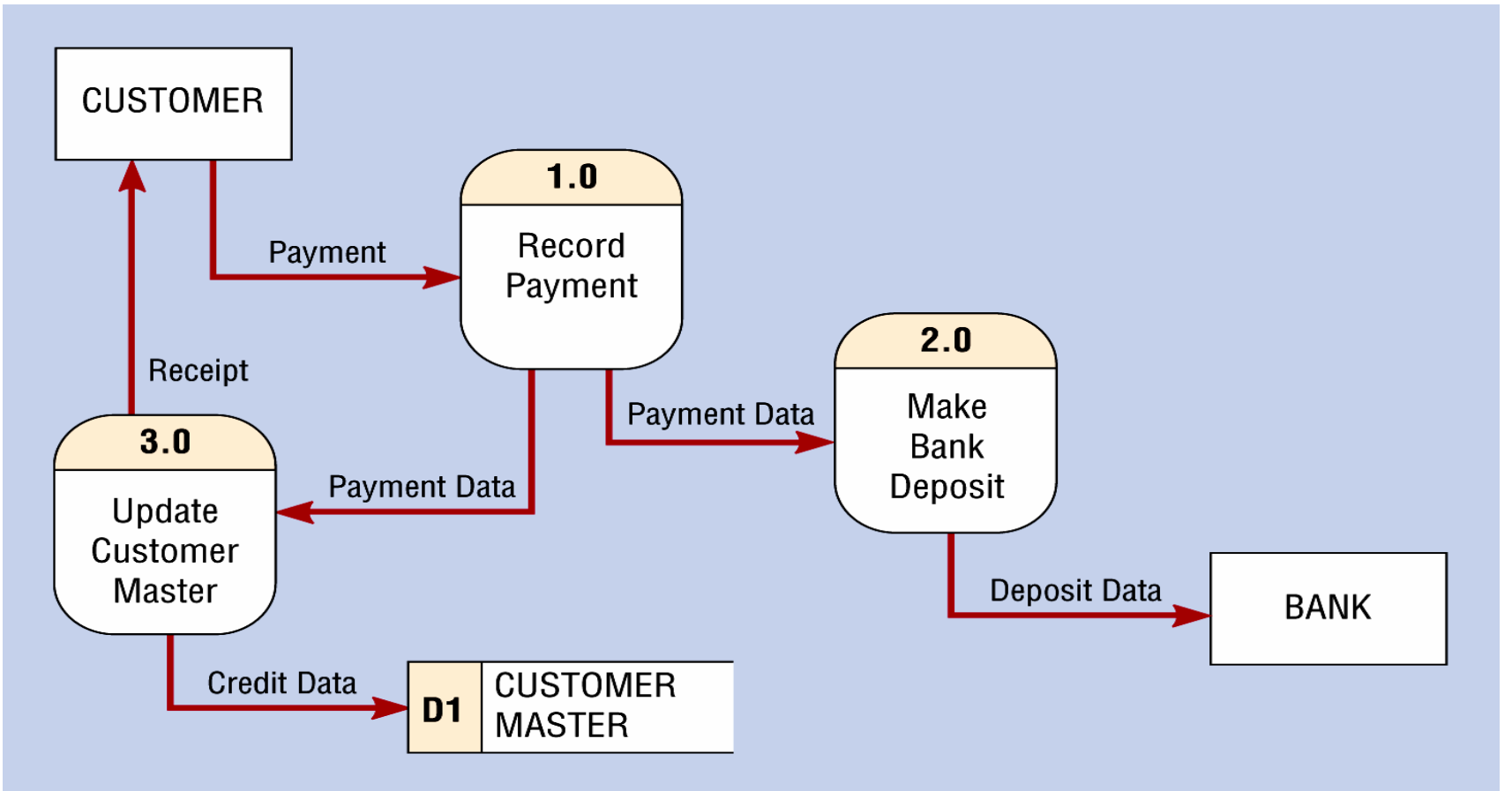
- **Not relevant:**
  - **where data are physically processed (entity !!!)**
  - **Who is processing data (entity !!!)**

- **But relevant: how data are processed within the system**

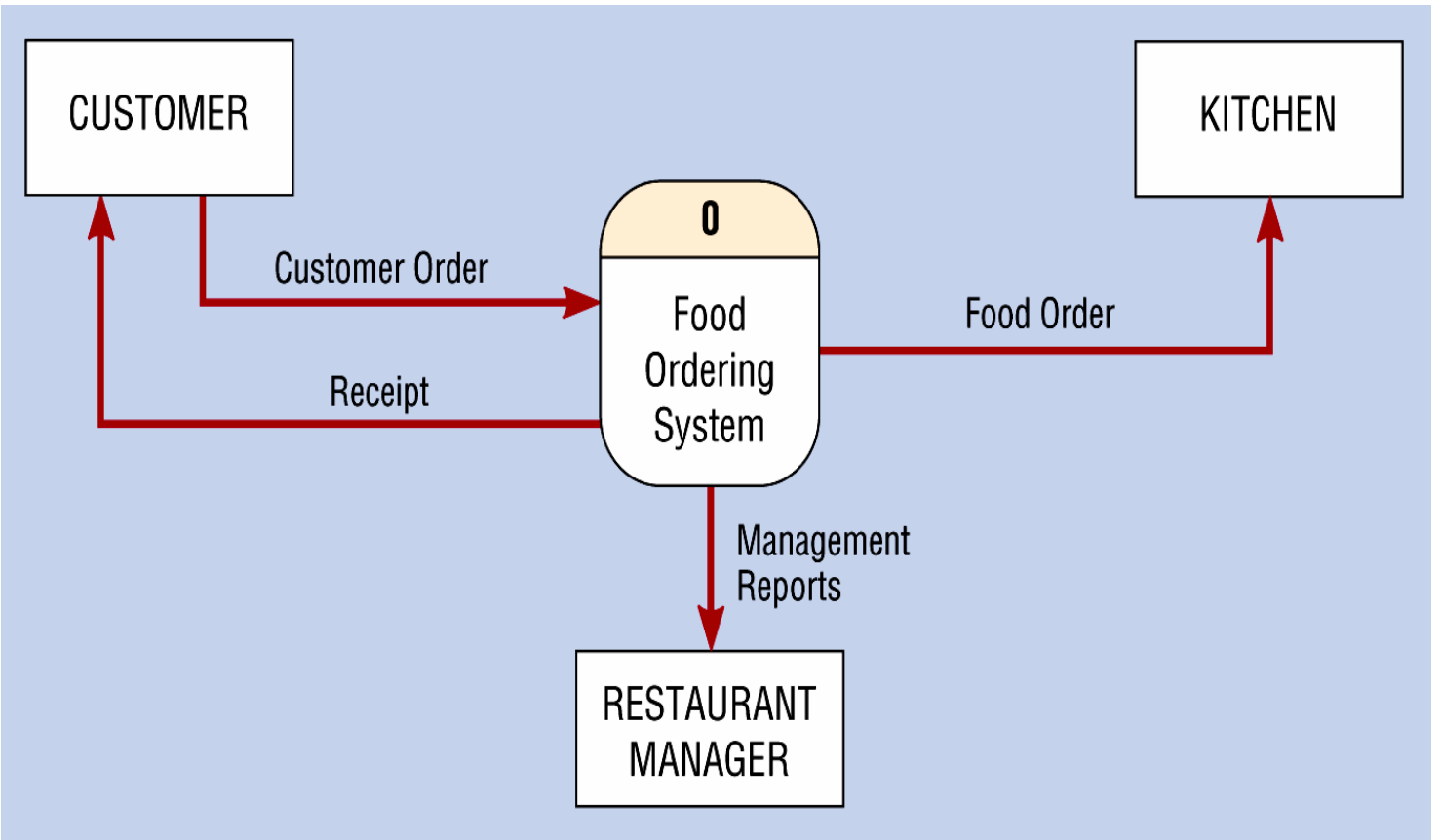
Wrong DFD


CUSTOMER
1.0
Record Payment
Payment
Receipt
2.0
Make Bank Deposit
Payment Data
Accounting Department
Payment Data
Deposit Data
BANK
Credit Data
D1
CUSTOMER MASTER

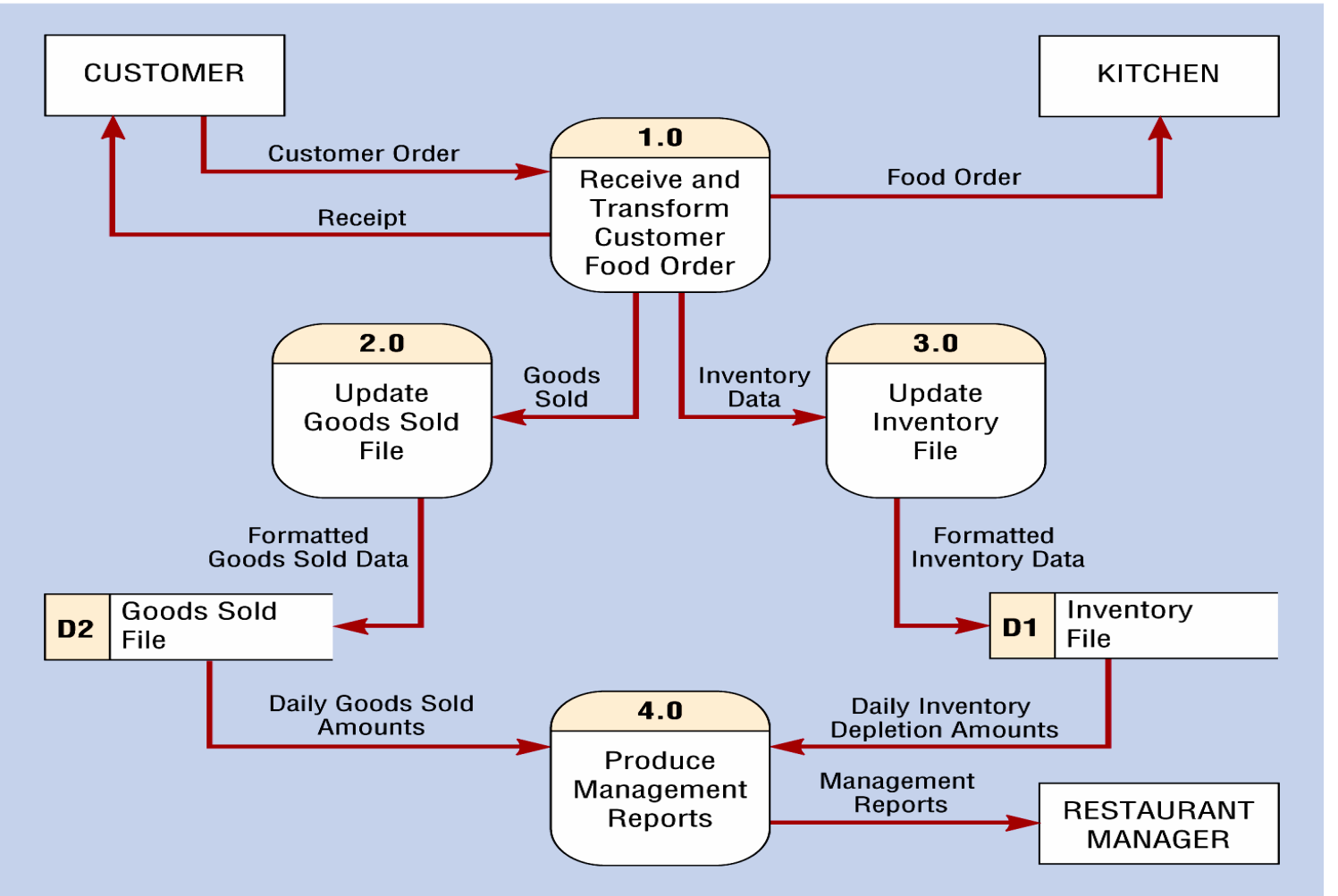
Correct DFD


CUSTOMER
1.0
Record Payment
Payment
Receipt
2.0
Make Bank Deposit
Payment Data
3.0
Update Customer Master
Payment Data
Deposit Data
BANK
Credit Data
D1
CUSTOMER MASTER

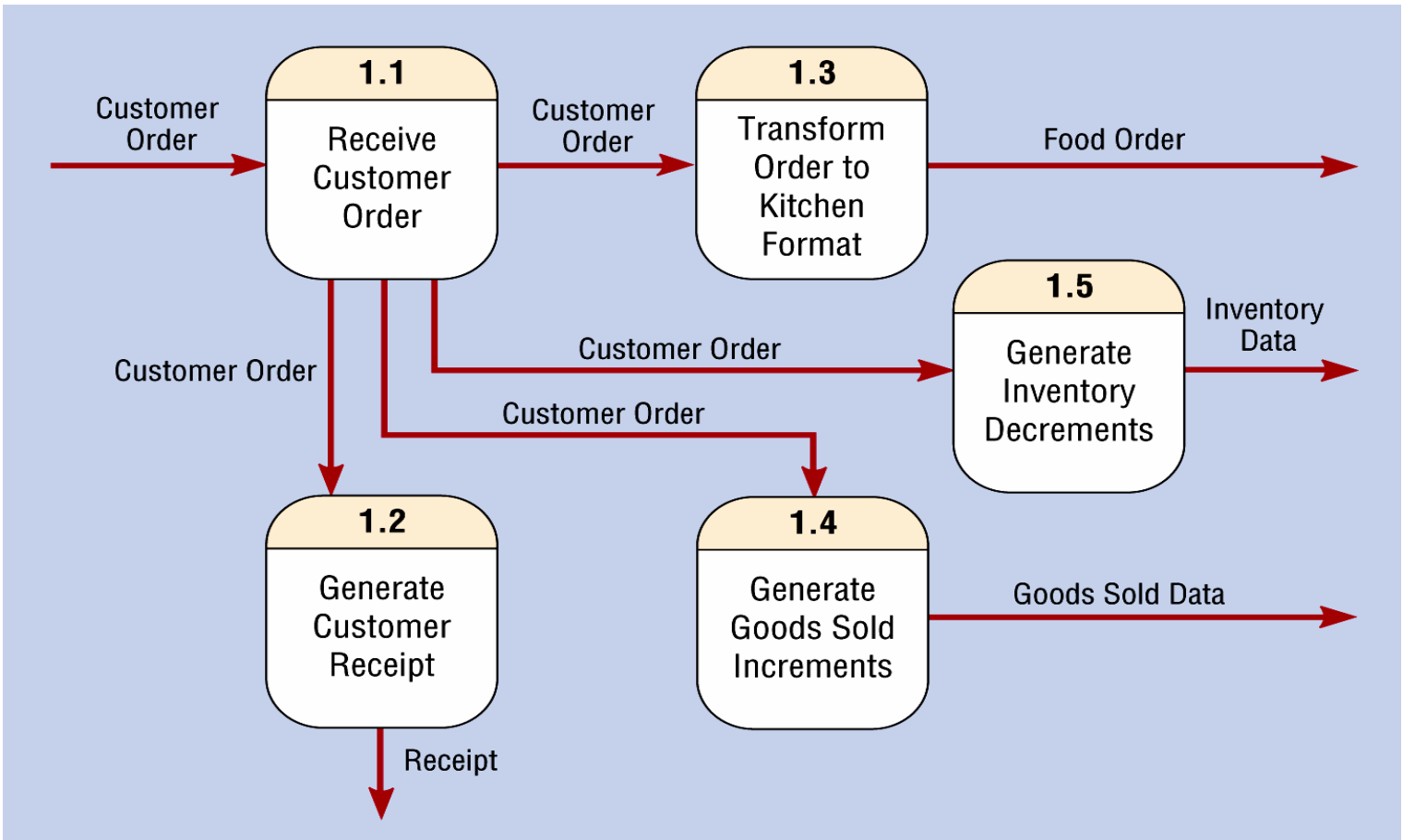
Context Diagram of a Food Ordering System (again)



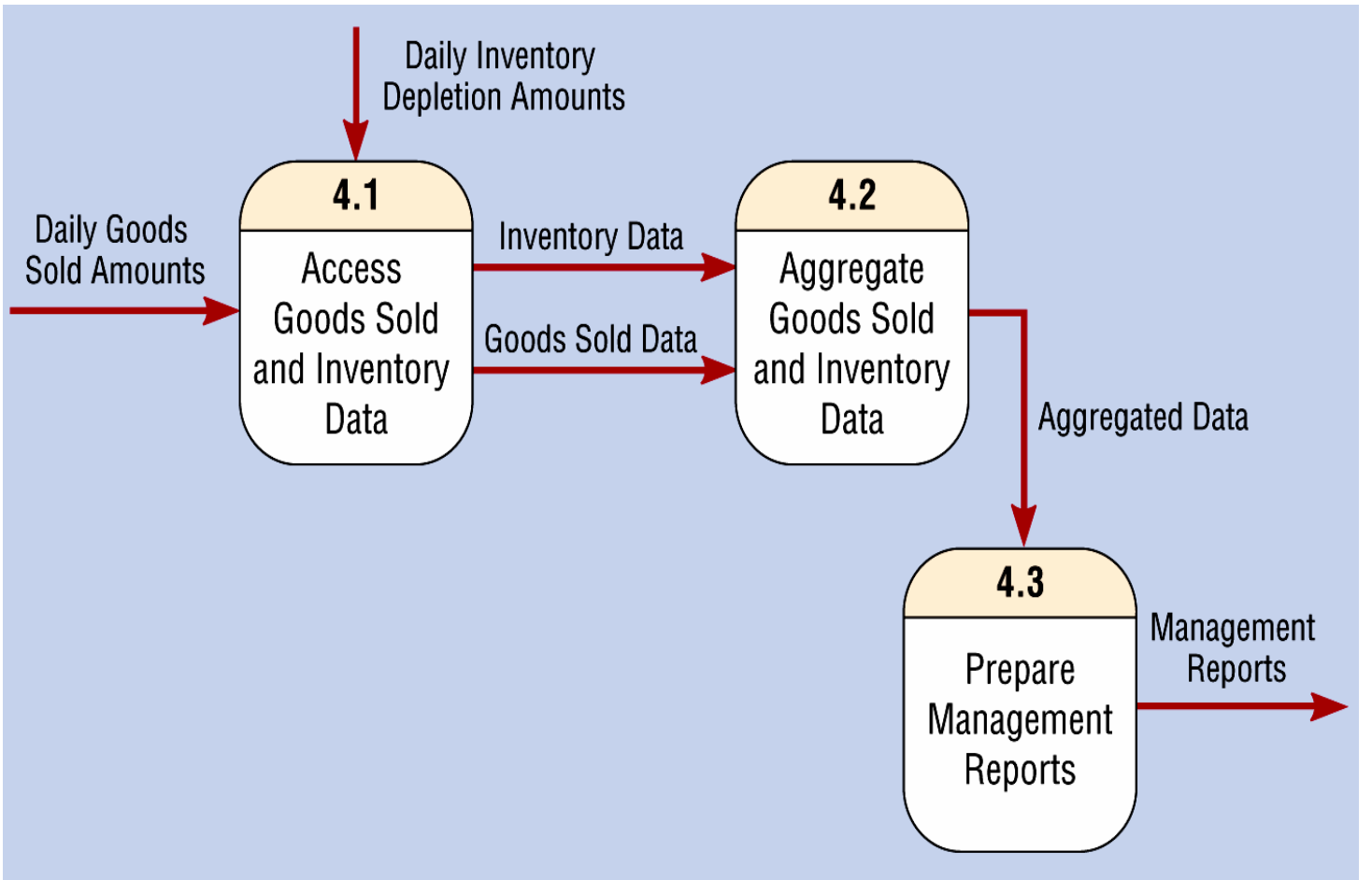
Level-0 DFD


CUSTOMER
KITCHEN
1.0
Receive and Transform Customer Food Order
Customer Order
Receipt
Food Order
2.0
Update Goods Sold File
3.0
Update Inventory File
Goods Sold
Inventory Data
Formatted Goods Sold Data
Formatted Inventory Data
D2
Goods Sold File
D1
Inventory File
Daily Goods Sold Amounts
Daily Inventory Depletion Amounts
4.0
Produce Management Reports
Management Reports
RESTAURANT MANAGER

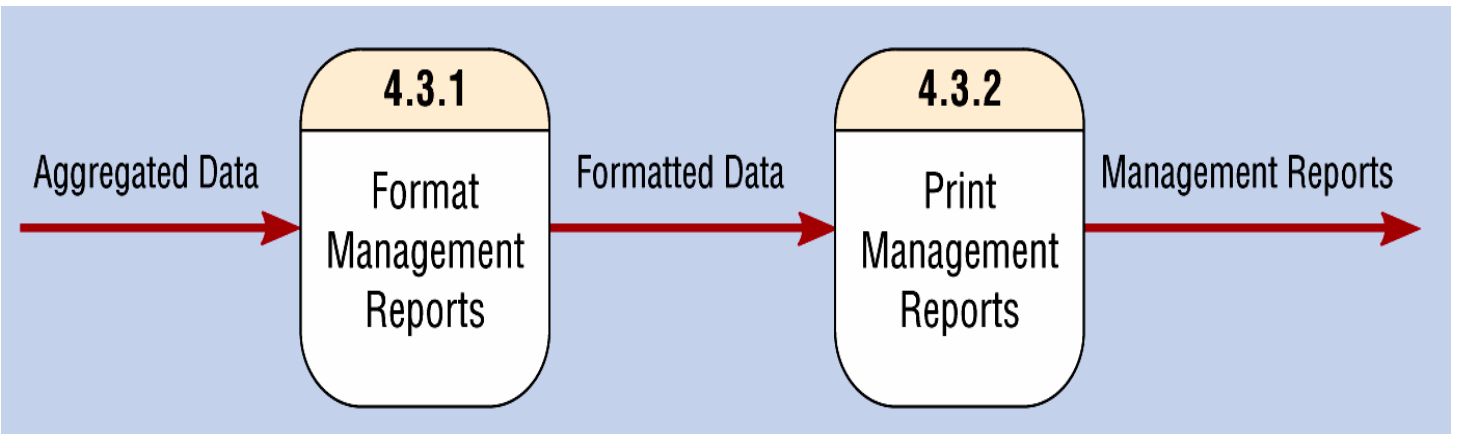
Level-1 DFD of Process 1.0 from the Level-0 DFD



Level-1 DFD of Process 4.0 from the Level-0 DFD



Level-2 DFD of Process 4.3 from the Level-1 DFD for Process 4.0



- Basic rules that apply to all DFDs
  - Inputs to a process are always different than outputs
  - Objects always have a unique name
    - In order to keep the diagram uncluttered, you can repeat data stores and sources/sinks on a diagram

DFD Rules

- Process
  A. No process can have only outputs (a miracle)
  B. No process can have only inputs (black hole)
  C. A process has a verb phrase label

- Data Store
  D. Data cannot be moved directly from one store to another
  E. Data cannot move directly from an outside source to a data store
  F. Data cannot move directly from a data store to a data sink
  G. Data store has a noun phrase label

DFD Rules: Source/Sink

- Source/Sink
  H. Data cannot move directly from a source to a sink
  I. A source/sink has a noun phrase label

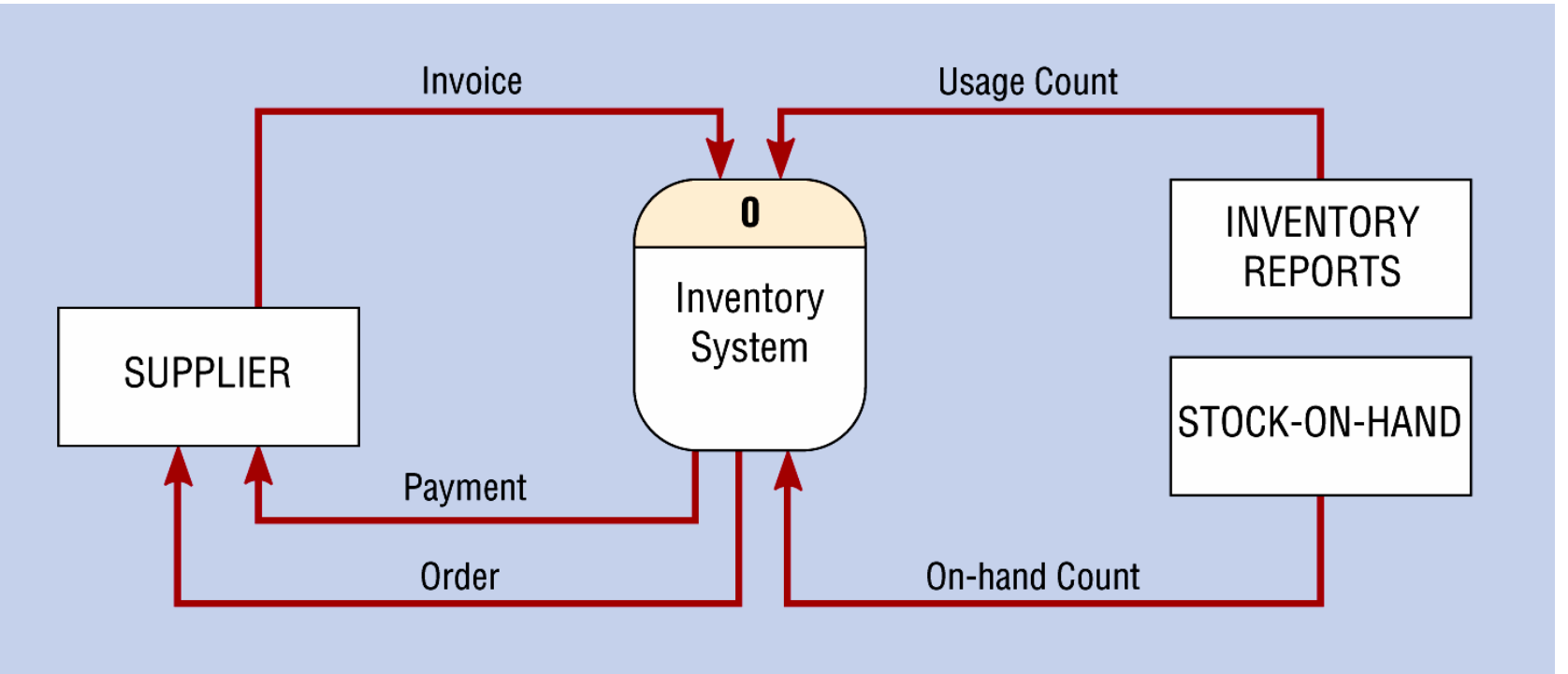
DFD Rules: Data Flow

- Data Flow
  - J. A data flow has only one direction of flow between symbols
  - K. A fork means that exactly the same data goes from a common location to two or more processes, data stores or sources/sinks
  - L. A join means that exactly the same data comes from any two or more different processes, data stores or sources/sinks to a common location
  - M. A data flow cannot go directly back to the same process it leaves
  - N. A data flow to a data store means update
  - O. A data flow from a data store means retrieve or use
  - P. A data flow has a noun phrase label

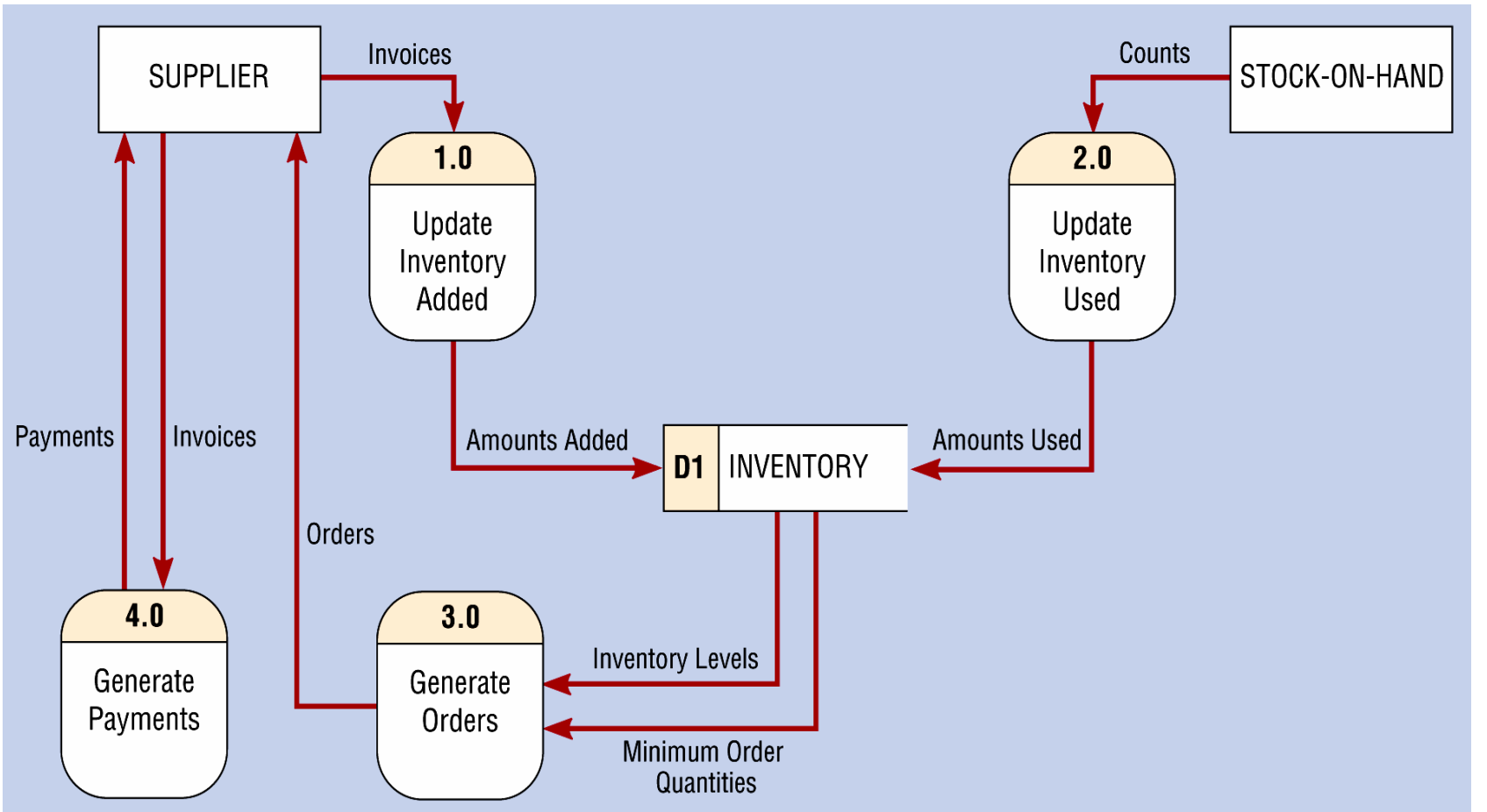
Another example

Inventory Control System for the previous Food Processing Company

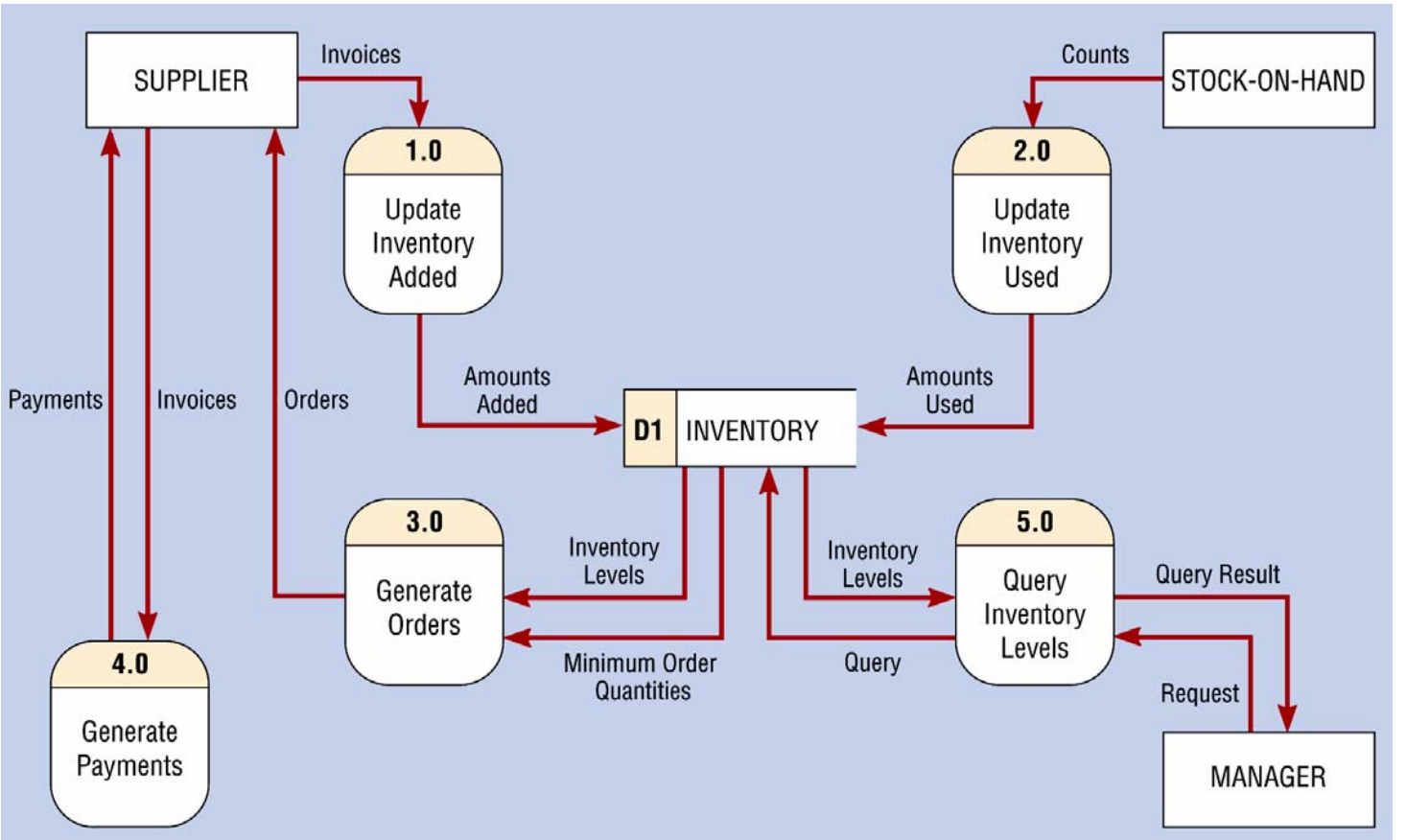
Inventory Control System: Context Diagram



Level-0 DFD "Current Inventory Control System"



Level-0 DFD "New Inventory Control System"



End of Process Modelling "DFDs"

Logic Modelling

Sys Modelling and Deliverables

Done

Now

System Modelling	Deliverables
Interface Modelling	Context diagram
Process Modelling	DFDs
Logic Modelling (Process Logic)	• **Structured English** • **Decision Tables** • **Decision Trees** • **Other diagrams ..**
Data Modelling	ER diagram

Logic Modelling

- **Logic modelling:**

  Representing internal structure & functionality of processes on a DFD

- DFDs **do not show the logic inside** the processes

- Logic modeling can also be used to show when processes on a DFD occur

Deliverables of Process Logic Modelling

- **Structured English /Pseudocode / Equations /Narrative text**
- **Decision tables**
- **Decision trees**
- **Diagrams and/or charts**
  - **State-Transition diagrams**
  - **Sequence diagrams (OOAD)**
  - **Activity diagrams (OOAD)**

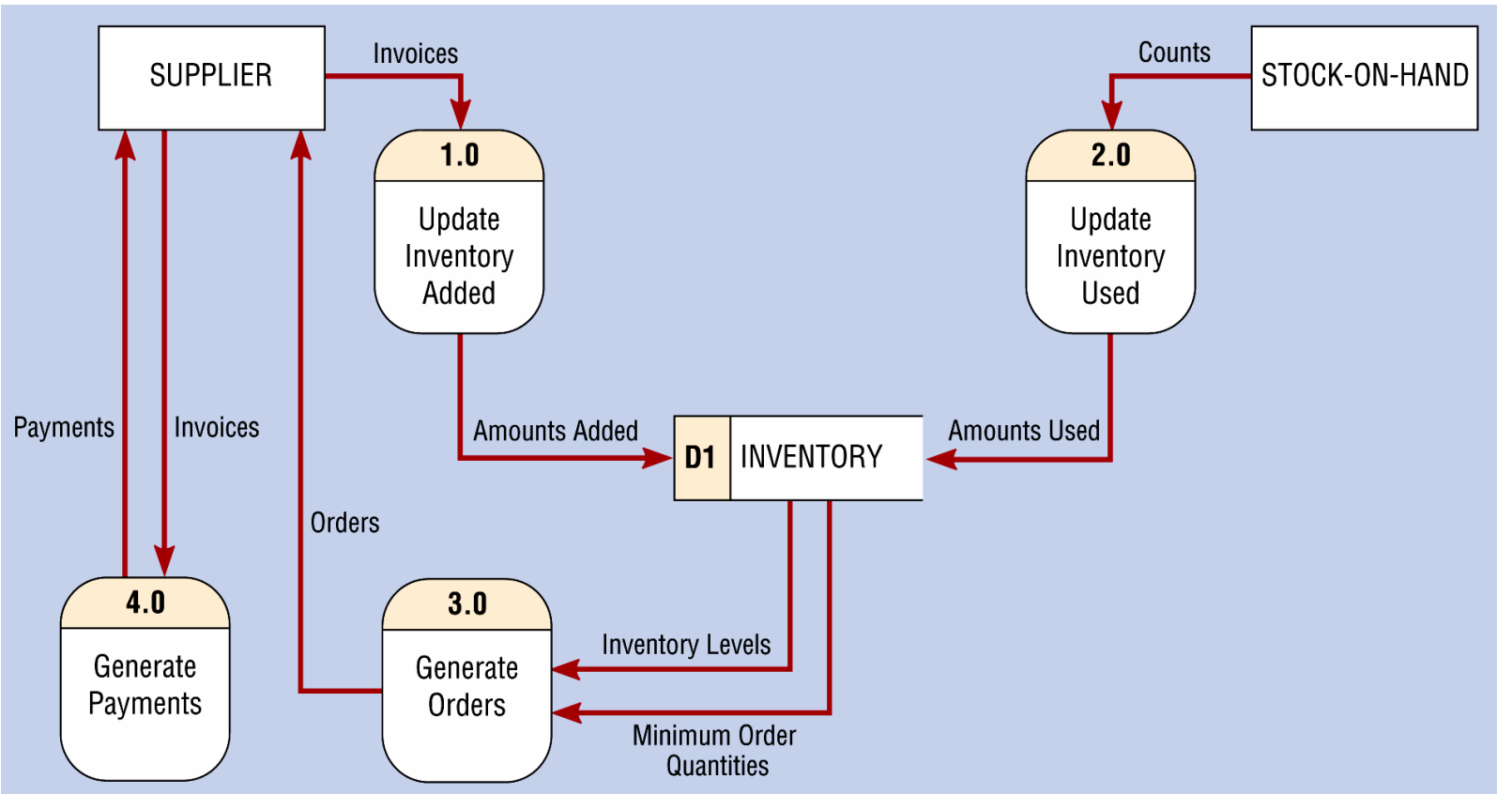
Logic Modelling

Structured English /Pseudocode

- Similar to programming language
  - If conditions
  - Case statements

- Example for **process 3.0** of Level-0 DFD "Current Inventory Control System"

Level-0 DFD "Current Inventory Control System" (Again!)



Process 3.0 Logic Modelling

Structured English /Pseudocode

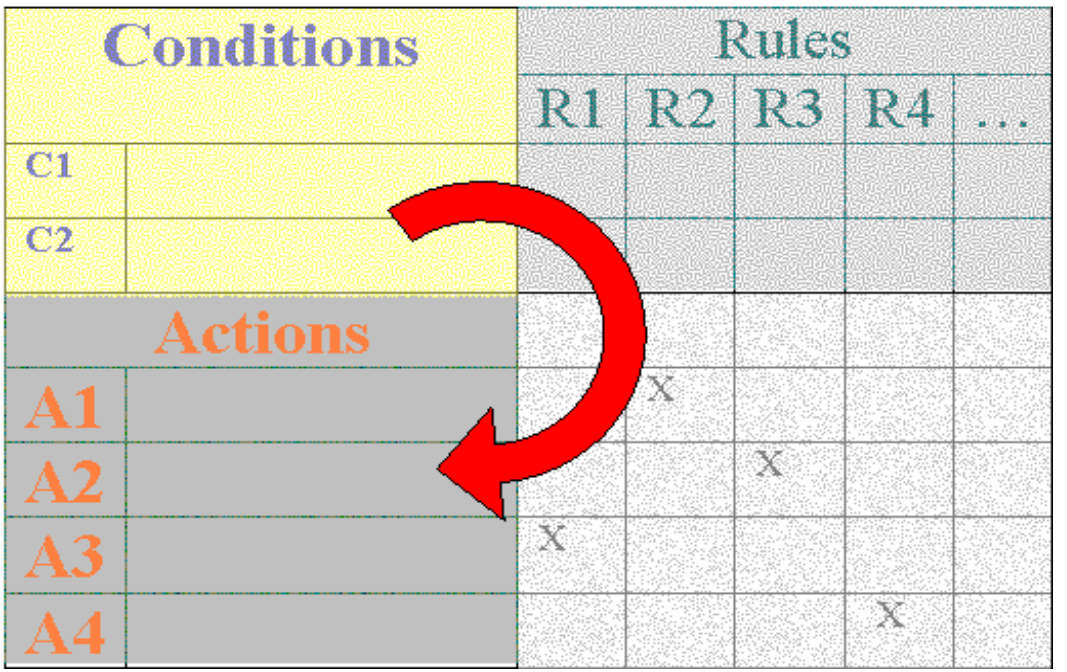
- Process 3.0: Generate Orders

```
DO
        READ next Item Inventory record
        BEGIN IF
          IF Quantity-in-stock < minimum-reorder –level
          THEN generate order for this item
        END IF
UNTIL end of file
```

Logic Modelling: Decision Tables

- A table representation of the logic of a decision
- Specifies the possible conditions and the resulting actions
- Best used for complicated decision logic

Logic Modelling: Decision Tables



If C1 AND C2 Then Action

Logic Modelling: Decision Tables Number of rules

If

- C1 has n1 values
- C2 has n2 values
- .....
- Ci has ni values
- …….
- Cm has nm values

Number of rules = n1x n2 x … x ni x…x nm

$$= \prod_{i=1}^{i=m} (ni)$$

Logic Modelling: Decision Tables

How to create a decision table

- Name the condition and values each condition can assume
- Name all possible actions that can occur
- List all rules
- Define the actions for each rule
- Simplify the table

Logic Modelling: Decision Tables

Example: Decision table for a payroll system

- **Employee type: 2 values**
  - **Salaried 'S' :**
  - **By hour 'H':**
- **Hours worked: 3 values**
  - **< 40**
  - **= 40**
  - **> 40**
- **Number of rules = 6**

Logic Modelling: Decision Tables

Example: **Complete** decision table for a payroll system

	Conditions/ Courses of Action	Rules					
		1	2	3	4	5	6
Condition Stubs	Employee type	S	H	S	H	S	H
	Hours worked	<40	<40	40	40	>40	>40
Action Stubs	Pay base salary	X		X		X	
	Calculate hourly wage		X		X		X
	Calculate overtime						X
	Produce Absence Report		X				

Logic Modelling: Decision Tables

Example: **SIMPLIFIED** decision table for a payroll system

Conditions		Rules			
		R1	R2	R3	R4
C1	Employee type	S	H	H	H
C2	Hours worked		<40	40	> 40
A1	Pay base salary	X			
A2	Calculate hourly wage		X	X	X
A3	Calculate overtime				X
A4	Produce absence report		X		

Logic Modelling: Decision Tree

Logic Modelling: Decision Tree

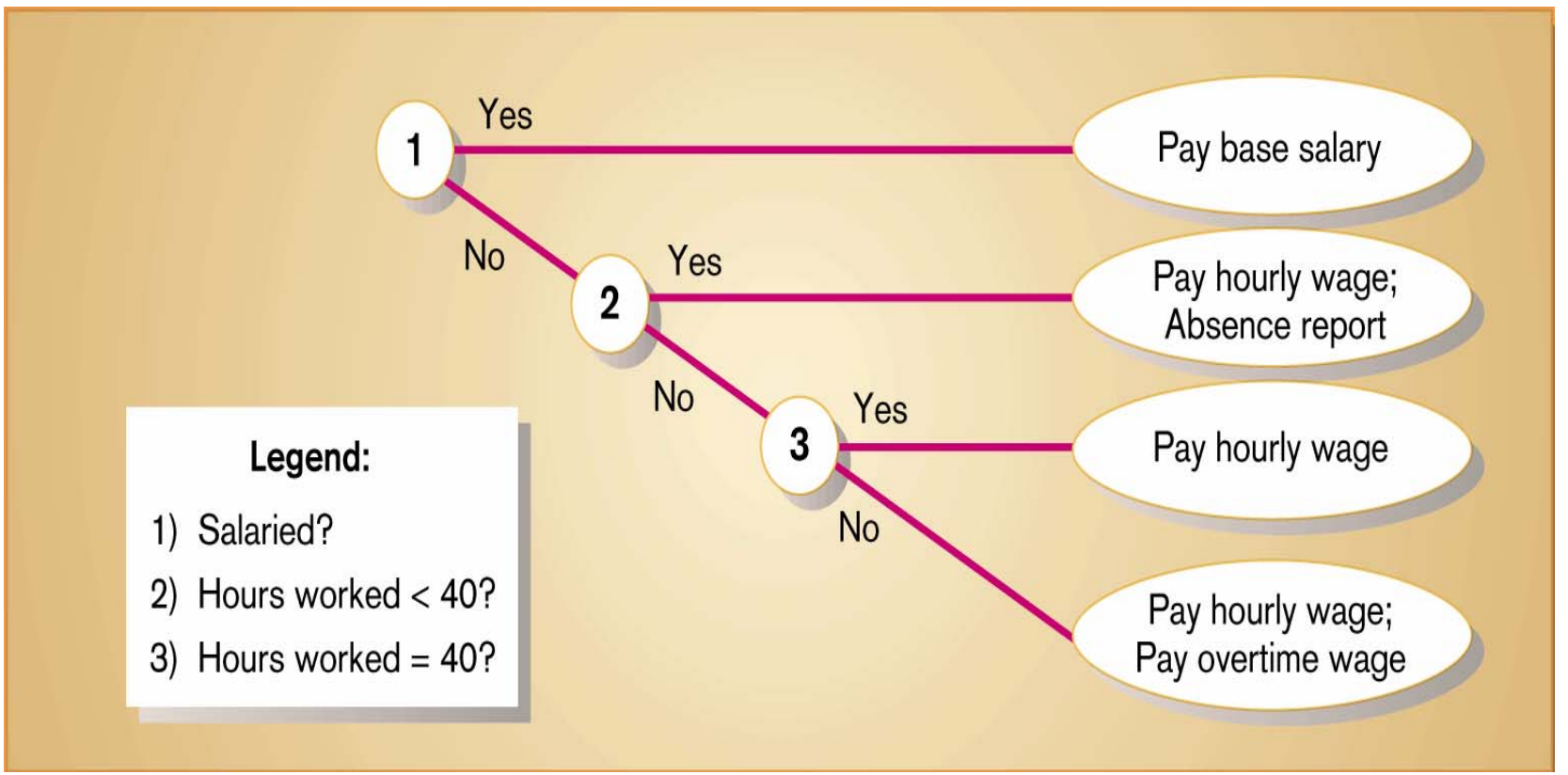
- A graphical representation of a decision situation

- Two elements:
  - **Nodes**: Decision points
  - **ovals**: Actions

- Read from left to right
- All possible actions are listed on the far right

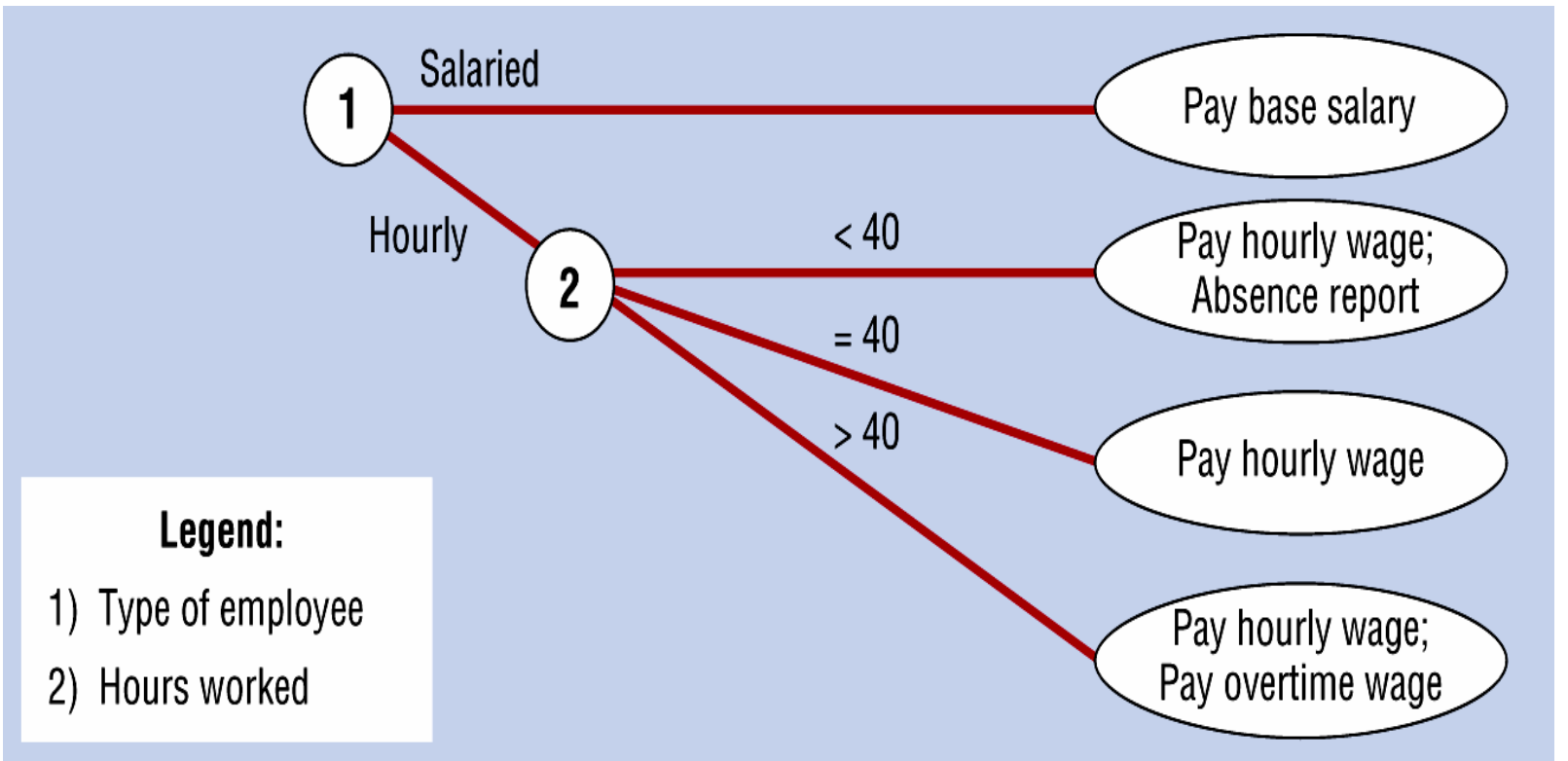
Logic Modelling: Decision Tree

Two choices per decision point



Logic Modelling: Decision Tree

Multiple Choices per Decision Point



Logic Modelling: Comparison

Criteria	Structured English	Decision Tables	Decision Trees
Determining Conditions and Actions	Second Best	Third Best	Best
Transforming Conditions and Actions into Sequence	Best	Third Best	Best
Checking Consistency and Completeness	Third Best	Best	Best

Logic Modelling

Logic Modelling

End …

What is software?

- Software is:
  - Computer programs + associated documentation

- Software products may be:
  - **Generic Software** - developed to be sold to a range of different customers
  - **Bespoke (custom) Software** - developed for a single customer according to given specification

What is Product Engineering?

- Producing a product
- Optimally
  - Highest quality
  - Minimum resources
- Under specified constraints

Software Engineering

Concerned with developing:

- High quality software
- Within budget
- On time

Software costs

- Software costs often dominate system costs.
- System costs includes:
  - Hardware cost
  - Software cost
  - people cost

- For systems with a long life, maintenance costs may be several times development costs

What is software engineering?

- Software engineering is concerned with all aspects of software production
- Software engineering is a modelling activity: Deals with complex systems through modelling
- Software engineering is a problem-solving activity: models are used to search for an acceptable solution

What is software engineering? (cont.)

- Software engineering is not a mathematical science: it relies on empirical methods
- Software engineering is a knowledge acquisition activity

What is the difference between software engineering and product engineering?

- Product engineering
  - A prototype is realised & tested
  - Thousands of items are manufactured
- Software engineering
  - Every software is a prototype
  - No manufacturing

Software engineers

- Software engineers should
  - adopt a systematic and organised approach to their work and
  - use appropriate tools and techniques depending on
    - the problem to be solved,
    - the development constraints and
    - the resources available

What is the difference between software engineering and computer science?

- Computer science is concerned with theory and fundamentals

- Software engineering is concerned with the practicalities of developing useful software

What is the difference between software engineering and system engineering?

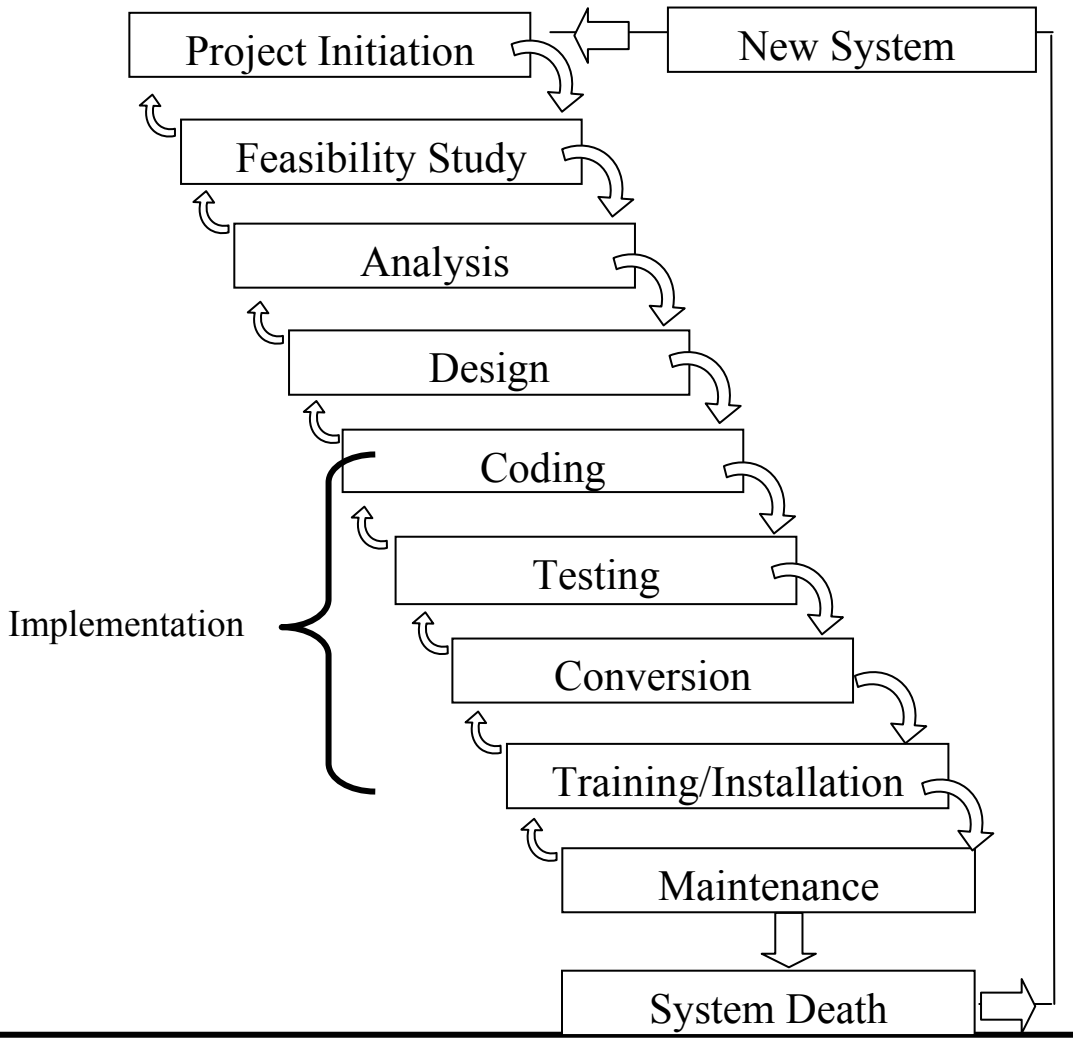
- System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering.
- Software engineering is part of system engineering
- System engineers are involved in system specification, architectural design, integration and deployment

System Development Life Cycle "SDLC"

Includes the following phases:

- Project Initiation
- Feasibility Study
- Analysis
- Design
- Implementation
  - Coding + Documentation
  - + Testing
  - + Conversion
  - + Training + Installation
- Maintenance

System Development Life Cycle "SDLC"



System Development Life Cycle "SDLC" (cont.)

Project Initiation

- Client faces a problem
- Needs for an improvement

- Sources of potential projects:
  - Top management
  - Steering committee
  - Users

:

System Development Life Cycle "SDLC" (cont.)

Feasibility Study

- Alternative solutions
- Economic feasibility (cost)
- Technical feasibility (technology available)
- Schedule feasibility (delivery date)
- Human Resources feasibility (New staff, Training)

System Development Life Cycle "SDLC" (cont.)

Analysis: Determine and structure system requirements

- Facts Finding / Requirement Elicitation / Domain Info / Requirements capturing

- Requirements Structuring into Diagrams & Text
  - Natural language for text
  - Context Diagram
  - Data Flow Diagrams "DFDs"
  - ER Diagrm
  - Decision Trees / Decision tables
  - Use Case Diagrams, Use Case description (for Object Oriented Analysis" OOA")

System Development Life Cycle "SDLC" (cont.)

Design: Create new System designs

- System Architecture design
- Database Design "Normalised relations"
- Input GUI design "Graphical User Interface"
  - Forms
  - Menus
  - Icons
  - Dialogue boxes, etc
- Output design
  - Reports
  - Queries

- *Interface Design with*
  - *Subsystems (modules)*
  - *other external systems (bank sys, GOSI sys, ...)*

System Development Life Cycle "SDLC" (cont.)

Implementation: Translate designs into a working system

- Coding
- Testing
- Documentation
- Data conversion (from old to new system)
- Training
- Installation

System Development Life Cycle "SDLC" (cont.)

Maintenance: Evolving system

- Requirements WILL CHANGE to reflect dynamic environment of business
- Continuous process

What is a software process?

- A set of activities whose goal is the development or evolution of software

- Generic activities in all software processes are:
  - Specification - what the system should do and its development constraints
  - Development - production of the software system
  - Validation - checking that the software is what the customer wants
  - Evolution “Maintenance” - changing the software in response to changing demands

What is a model?

- An abstract representation of a system that enables us to answer questions about the system

- Used with too large, too small, too complex, or too expensive systems

What is a software process model?

- An abstract representation of a software process, presented from a specific perspective

- Examples of process perspectives are:
  - Workflow perspective - sequence of activities
  - Data-flow perspective - information flow
  - Role/action perspective - who does what

Software Generic Process Models

4 Generic process models:

1. Waterfall model
2. Evolutionary development model
3. Formal transformation model
4. Integration from reusable components model

Costs of software engineering

- Roughly:
  - **60%** of costs are development costs,
  - **40%** of costs are testing costs.

- For custom software, maintenance costs often exceed development costs

Costs of software engineering (cont.)

- Costs vary depending on the type of system being developed and the requirements of system attributes such as performance and system reliability

- Distribution of costs depends on the development model that is used

What are software engineering methods?

- Structured approaches to software development which include:
  - Model descriptions
    - Descriptions of graphical models which should be produced
  - Rules
    - Constraints applied to system models
  - Recommendations
    - Advice on good design practice
  - Process guidance
    - What activities to follow

What is CASE tool (Computer-Aided Software Engineering)

- Software system (tool)
- Provides automated support for software process activities.

- Upper-CASE
  - Tools to support the early process activities of requirements and design
- Lower-CASE
  - Tools to support late process activities such as programming, debugging and testing

Attributes of good software

- The software should deliver the required functionality and performance to the user

- The software and should be maintainable, dependable and usable

Attributes of good software (cont.)

- Maintainability
  - Software should be designed keeping in mind that it will **evolve** to meet changing needs (changes in Business Environment)
- Reliability
  - Software must be reliable
- Efficiency
  - Software should not make wasteful use of system resources
- Usability
  - Software must be usable by the users for which it was designed

Key challenges facing software engineering

Challenges in coping with:

- Legacy systems
  - Old, valuable systems that must be maintained and updated
- Heterogeneity
  - Systems are distributed and include a mix of:
  - Hardware, and
  - Software written using different languages, and
  - Different operating systems
- Delivery
  - Pressure for faster delivery of software

Professional and ethical responsibility

- Software engineering involves wider responsibilities than simply the application of technical skills

- Software engineers must behave in an honest and ethically responsible way if they are to be respected as professionals