Requirements Analysis-I

Requirements Analysis

It must be possible to test each requirement (Testability).

Categorization and prioritization of all requirements.

Prioritizing (Ranking) Use Cases

- Strategy :
  - pick the use cases that significantly influence the core architecture
  - pick the use cases that are critical to the success of the business.
  - a useful rule of thumb - pick the use cases that are of highest risk!!!

Requirements Analysis

- Completeness

  Self contained, no omissions.

- Error conditions

  State what happens in case of an error. How should the implementation react in case of an error condition?

Consistency of Requirements

- *Different requirements must be consistent.*

  Example:

  R1.2: The speed of the vehicle will never exceed 250 mph.

  R5.4: When the vehicle is traveling at a speed greater than 300 mph, a special "supervisory body" safety mechanism will be automatically activated.

Requirements Development Process

1. Define the project's vision and scope
2. Identify user classes
3. Identify appropriate representatives from each user class
4. Identify and document the hierarchy of users (for requirements decision-making process)
5. Select and plan requirements elicitation techniques
6. Apply the elicitation techniques to develop and prioritize the main use cases for the system (architecturally significant use cases). **One approach** is to identify all use cases but detail those with the highest **priority** first.

Requirements Development Process

7. Identify **quality attributes**, other **non-functional** requirements and constraints
8. Develop **functional requirements** from use cases.
9. Develop analysis models as needed to clarify problem domain and/or requirements
10. Optionally develop prototype to verify requirements with end users

System modelling

- System modelling helps the analyst to understand the functionality of the system and models are used to communicate with customers
- Different models present the system from different perspectives
  - *External perspective showing the system's context within the environment*
  - *Behavioural perspective showing the behaviour of the system (how it performs various activities)*
  - *Structural perspective showing the system architecture and/or data architecture*

System modelling …

- Structured methods incorporate system modelling as an inherent part of the method
- Methods define a set of models, a process for deriving these models and rules and guidelines that should apply to the models
- CASE tools support system modelling as part of a structured method
- *System Models :* Abstract descriptions of systems whose requirements are being analysed

Model types

- *Data processing model* showing how the data is processed at different stages
- *Composition model* showing how entities are composed of other entities
- *Architectural model* showing principal sub-systems
- *Classification model* showing how entities have common characteristics
- *Response model* showing the system's reaction to events

Key Definition

- *Decomposition* is the process of modeling the system and its components in increasing levels of detail.
- *Balancing* involves insuring that information presented at one level of a DFD is accurately represented in the next level DFD.

What are Data Flow Diagrams?

- Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs.
- Data Flow Modeling represents the flow of information around a system, the way it is changed and stored and the *'sources'* and *'sinks'* of information outside the system.
- DFDs show how information flows around a system, they:
  - represent a situation (scenario) from the viewpoint of the data (it's flow);
  - are a technique to analyze the processes in the system from most abstract level to most detailed level.

Data Flow Diagrams

The purpose and value of the data flow diagram is primarily *data* discovery within the environment, not *process* mapping - hence the name "data flow diagram" (DFD).

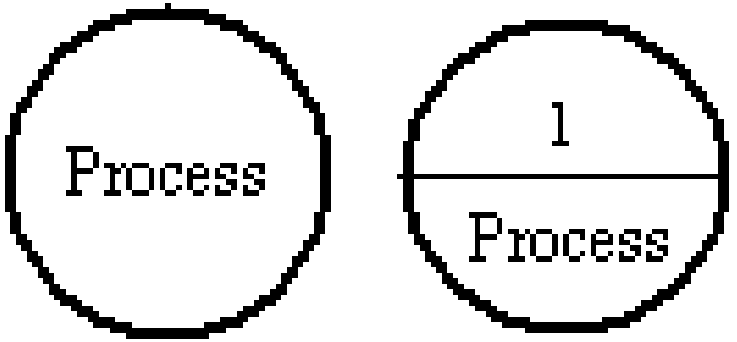
- Data Flow Diagrams (DFDs) take a *'top-down'* approach, expanding the system description into more and more detail via a series of *'levels'*, so a set of DFDs will comprise a **Context Diagram (**Level 0 DFD**)**, a Level 1 DFD, and perhaps some Level 2 DFDs, (one for each complicated Process at Level 1).

Data Flow Diagrams : Objectives

- to graphically document boundaries of a system;
- to provide hierarchical division/breakdown of the system;
- to show movement of information between a system and its environment;
- to document information flows within the system;
- to aid communication between users and analysts, analysts and designers, designers and developers, developers and testers etc.

Data Flow Diagram Notations

- **Process**
  A process transforms incoming data flow into outgoing data flow
- **Process Notation**

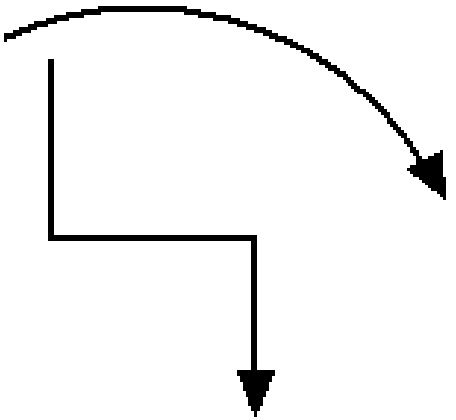


Data Flow Diagram Notations

- **DataStore** : Datastores are repositories of data in the system. They are sometimes also referred to as files.
- **Datastore Notations**

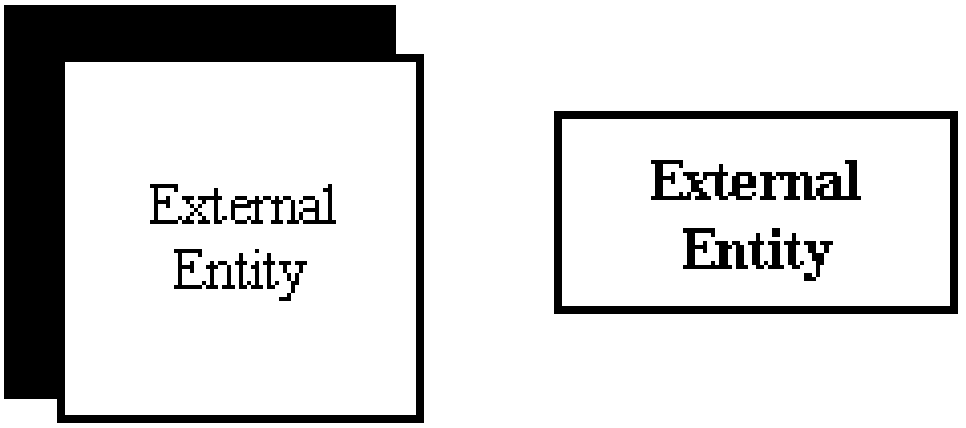
Data Flow Diagram Notations

- **Dataflow** : Dataflows are lines through which packets of information flow. Label the arrows with the name of the data that moves through it.

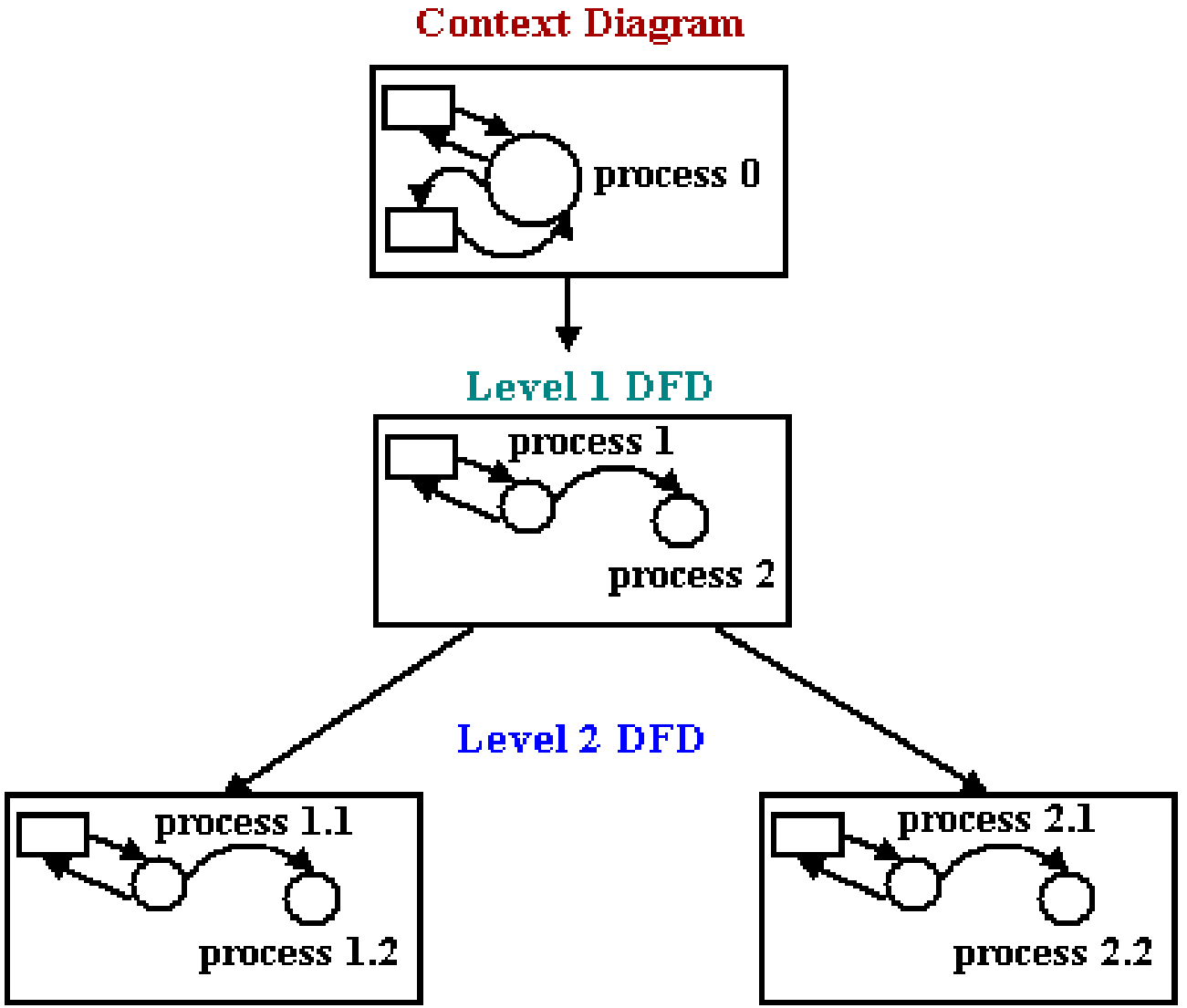


Data Flow Diagram Notations

- **External Entity** External entities are objects outside the system, with which the system communicates. External entities are sources and destinations of the system's inputs and outputs.

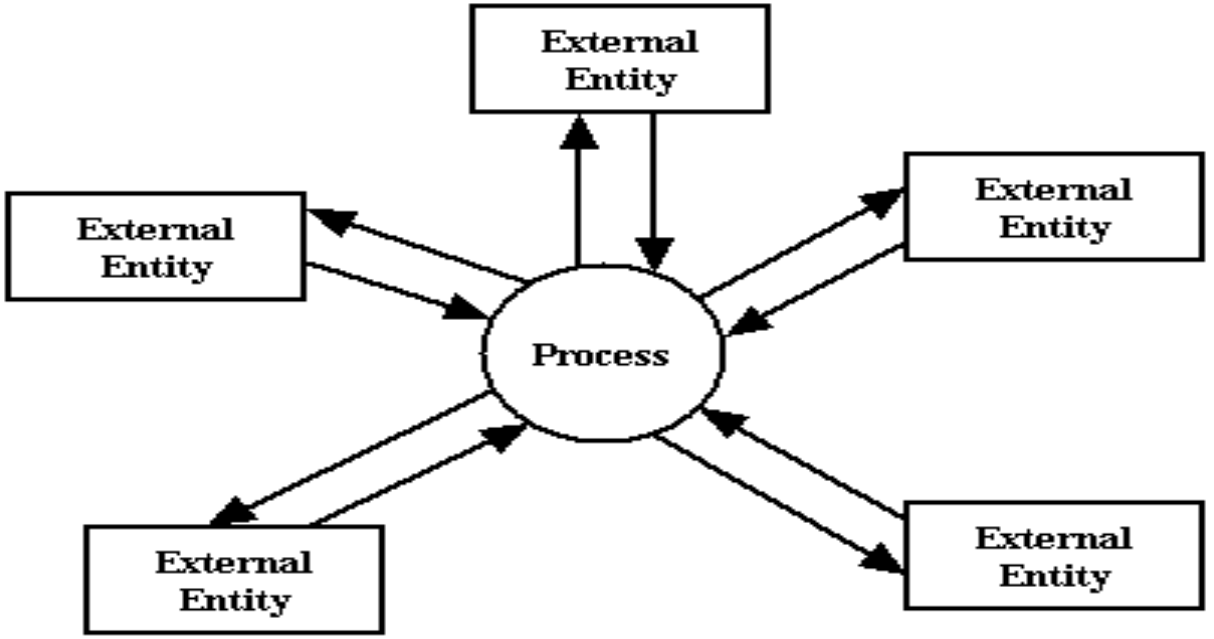


Data Flow Diagram Layers

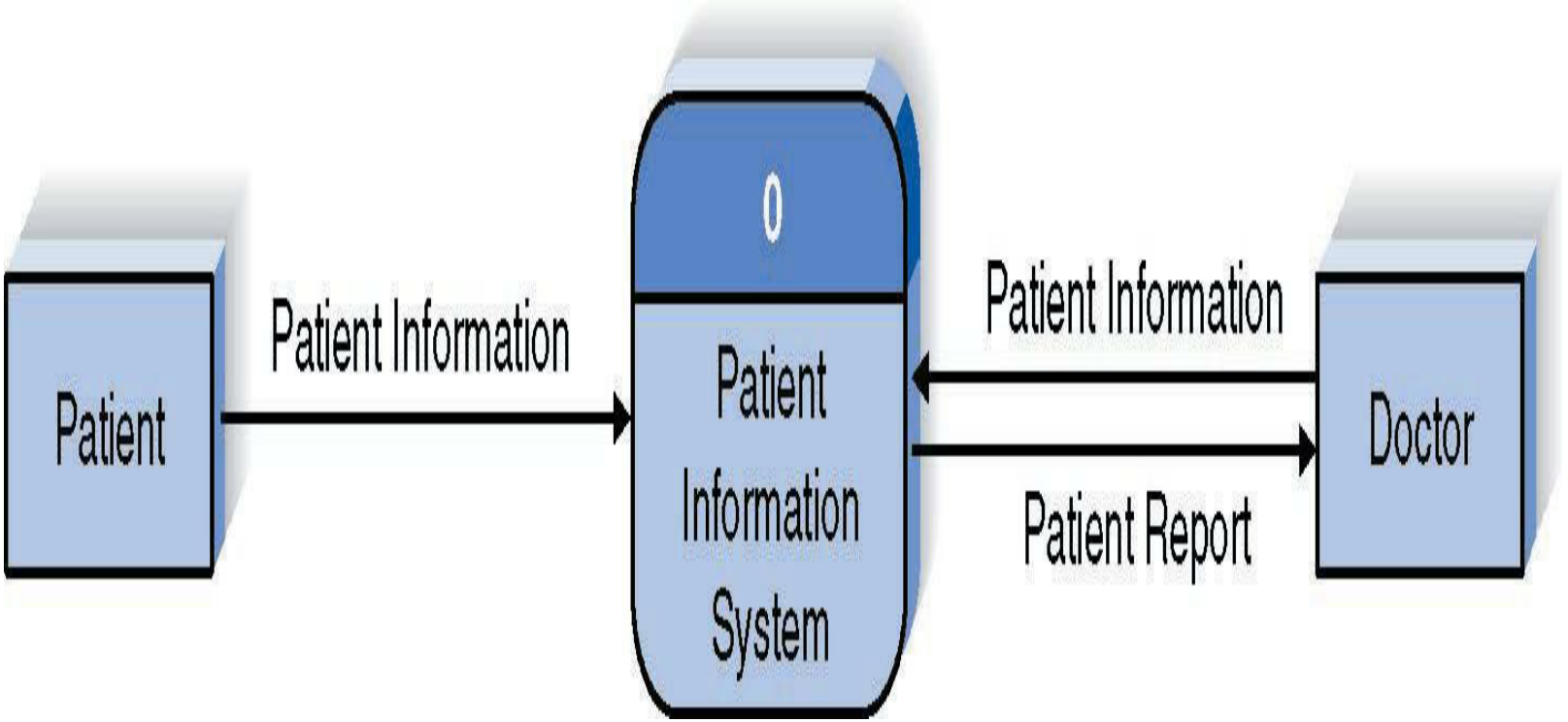


Context Diagrams

- A context diagram is a top level (also known as Level 0) data flow diagram. It only contains one process node (process 0) that generalizes the function of the entire system in relationship to external entities.



Example(Context Level DFD):

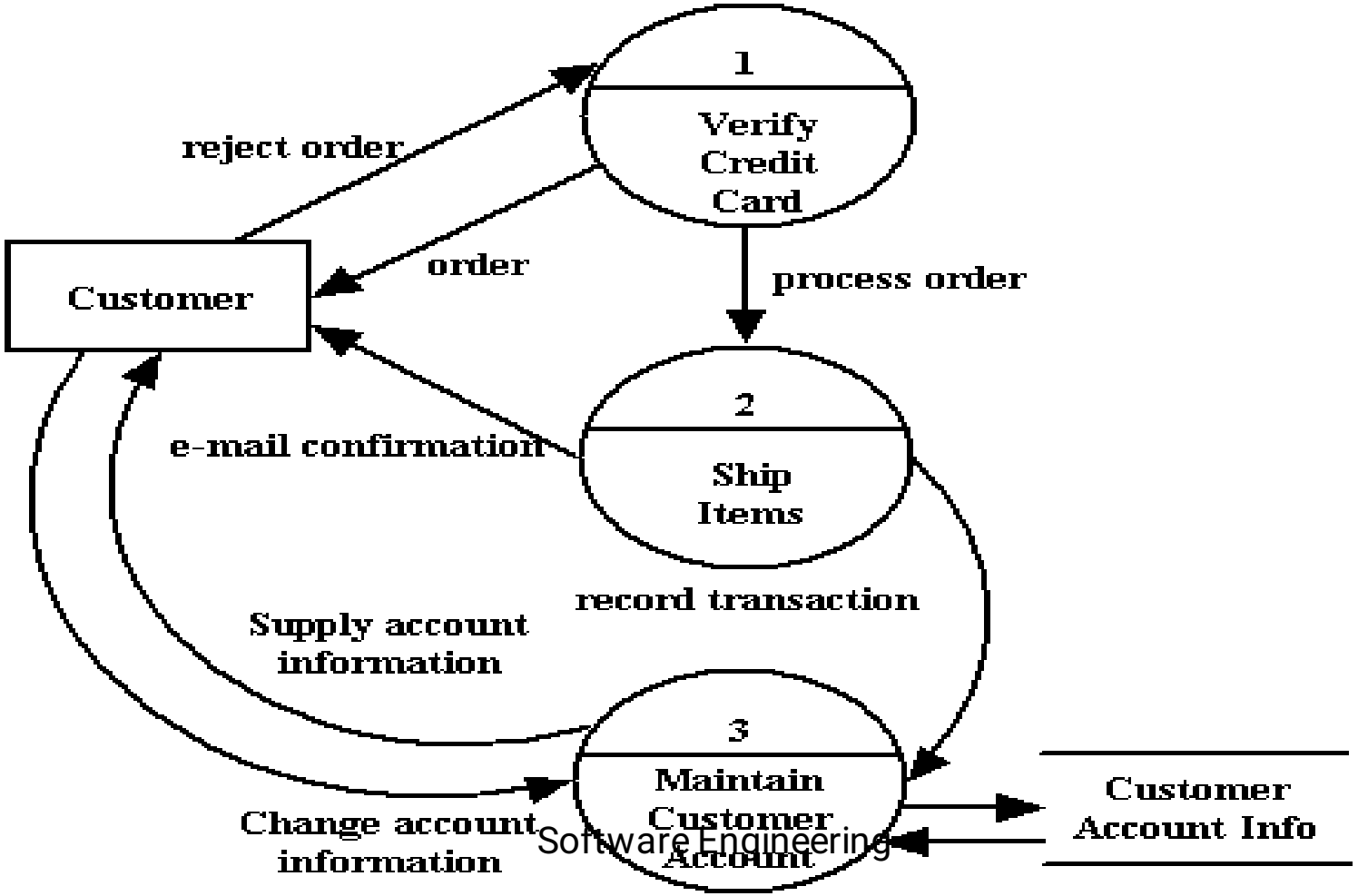


Context Diagram…

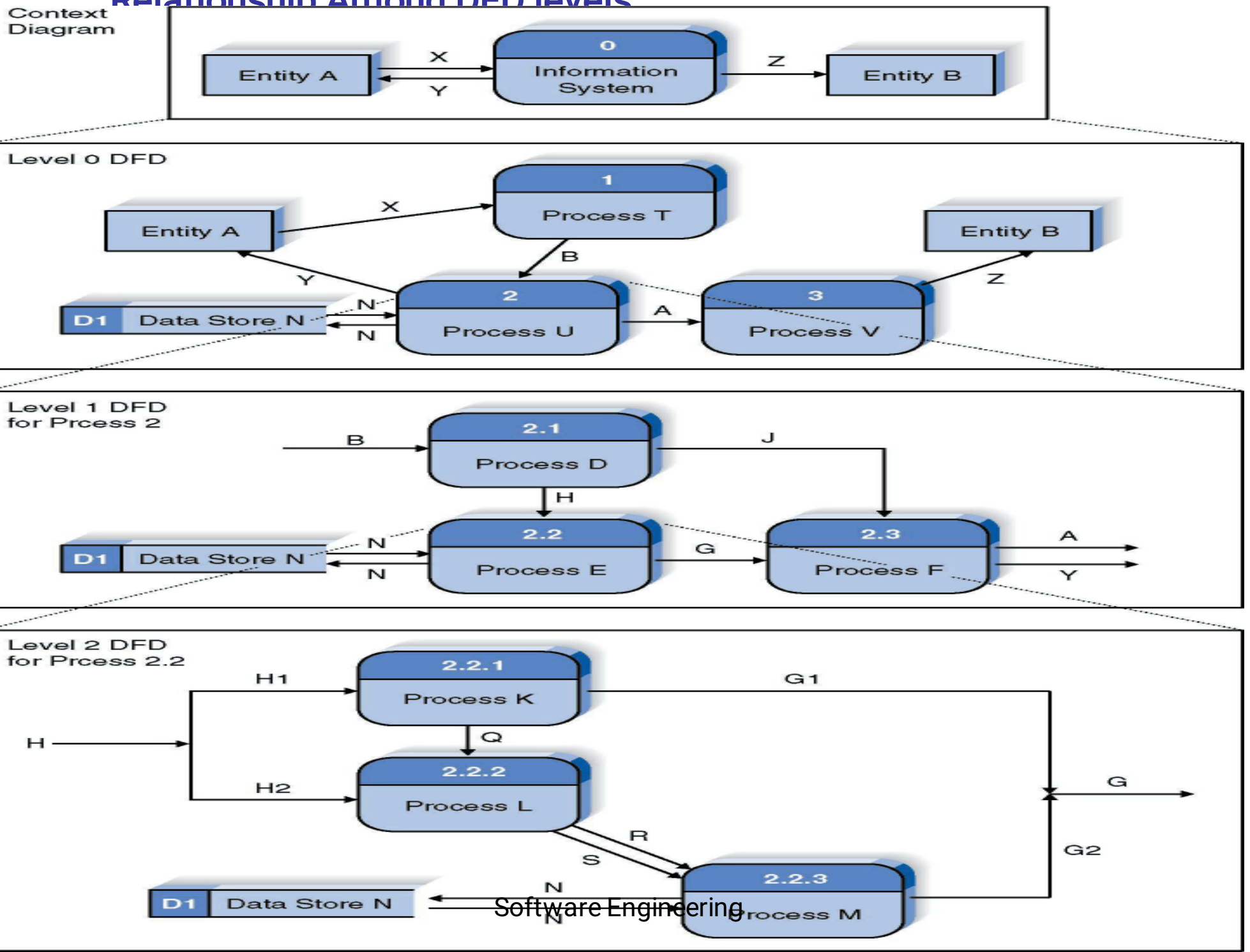
- Shows the context in which the business process exists
- Shows the overall business process as just *one* process
- Shows all the outside entities that receive information from or contribute information to the system
- No Data Stores

DFD levels

- The first level DFD shows the main processes within the system. Each of these processes can be broken into further processes until you reach pseudo-code.
- (for example Online shopping system)



Relationship Among DFD levels



Level 1 Diagram

- Shows all the processes that comprise the overall system
- Shows how information moves from and to each process
- Adds data stores

Level 2 Diagrams

- Shows all the processes that are sub processes of a process on the level 1 diagram
- Shows how information moves from and to each of these sub processes
- Shows in more detail the content of higher level process
- Level 2 diagrams may not be needed for all level 1 processes

Level 3 Diagrams

- Shows all processes that comprise a single process on the level 2 diagram
- Shows how information moves from and to each of these processes
- Level 3 diagrams may not be needed for all level 2 processes
- Correctly numbering each process helps the user understand where the process fits into the overall system

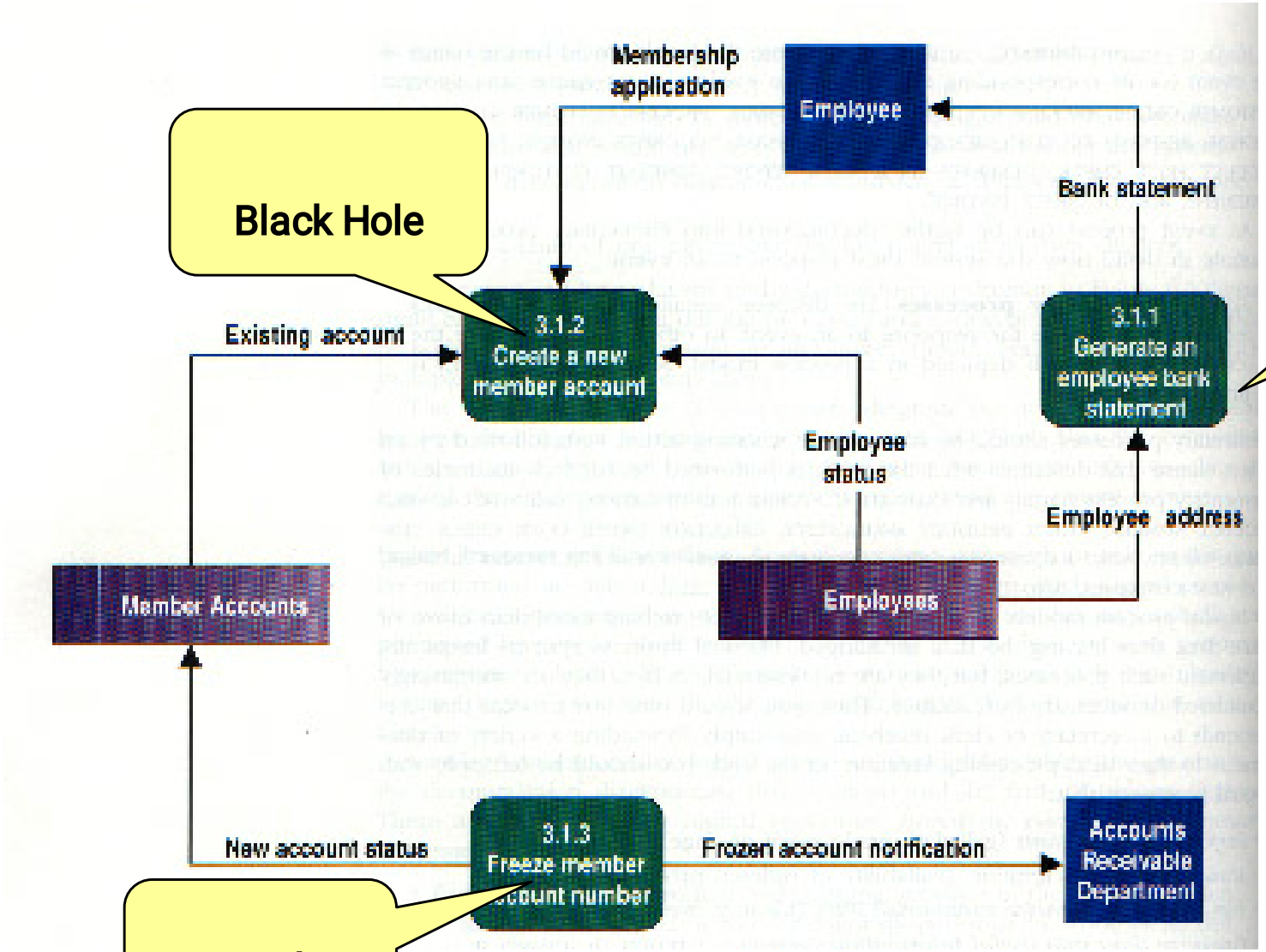
Steps in Building DFDs

- Build the context diagram
- Create DFD fragments for each scenario
- Organize DFD fragments into level 1
- Decompose level 1 DFDs as needed
- Validate DFDs with user

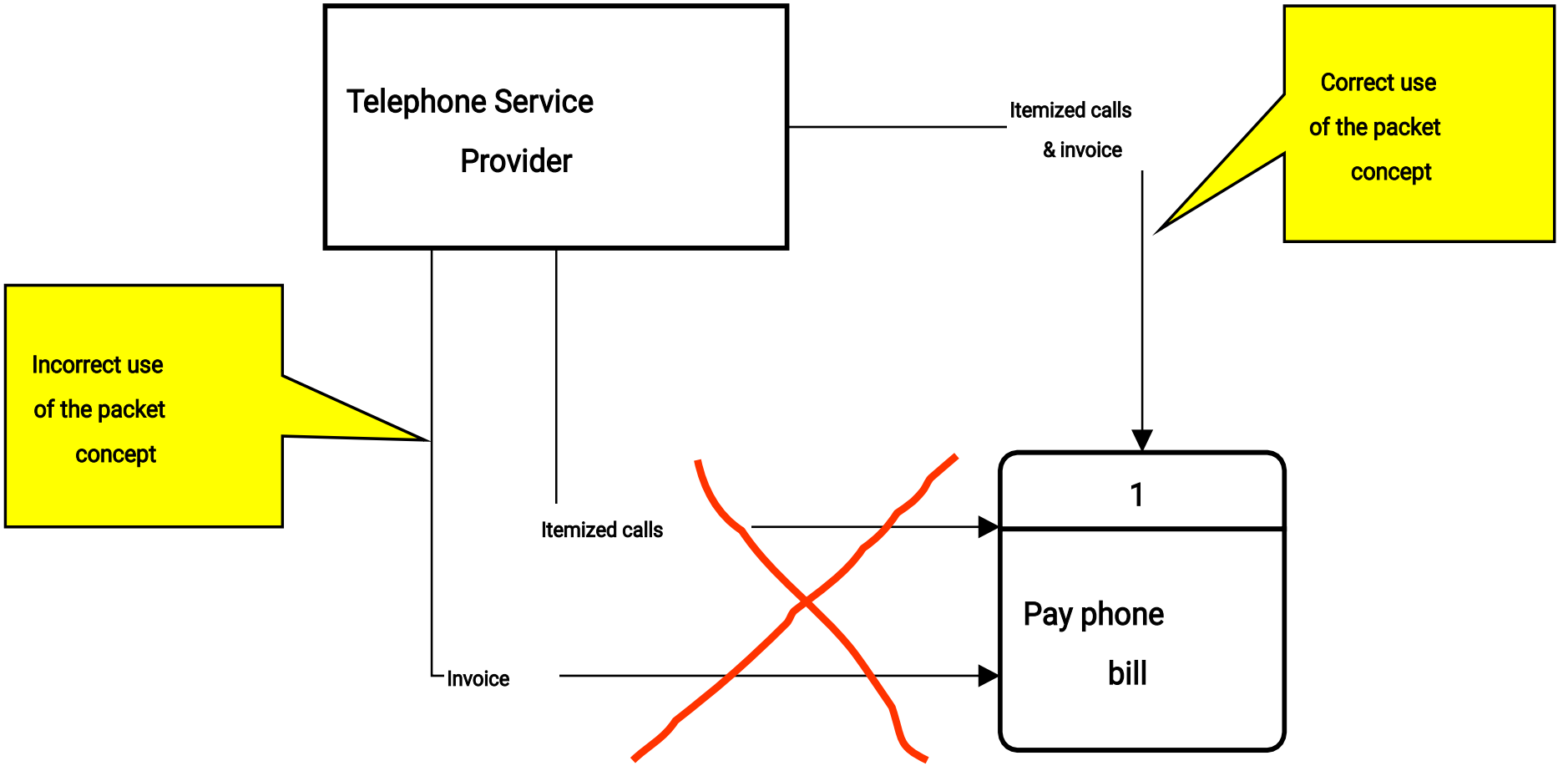
DFD Fragment Tips

- All process names must be verb phrases
- Maintain organization's viewpoint in naming processes
- Layouts often place
  - processes in the center
  - inputs from the left
  - outputs to the right
  - stores beneath the processes

DFD – Common Errors



DFD – Packet Concept



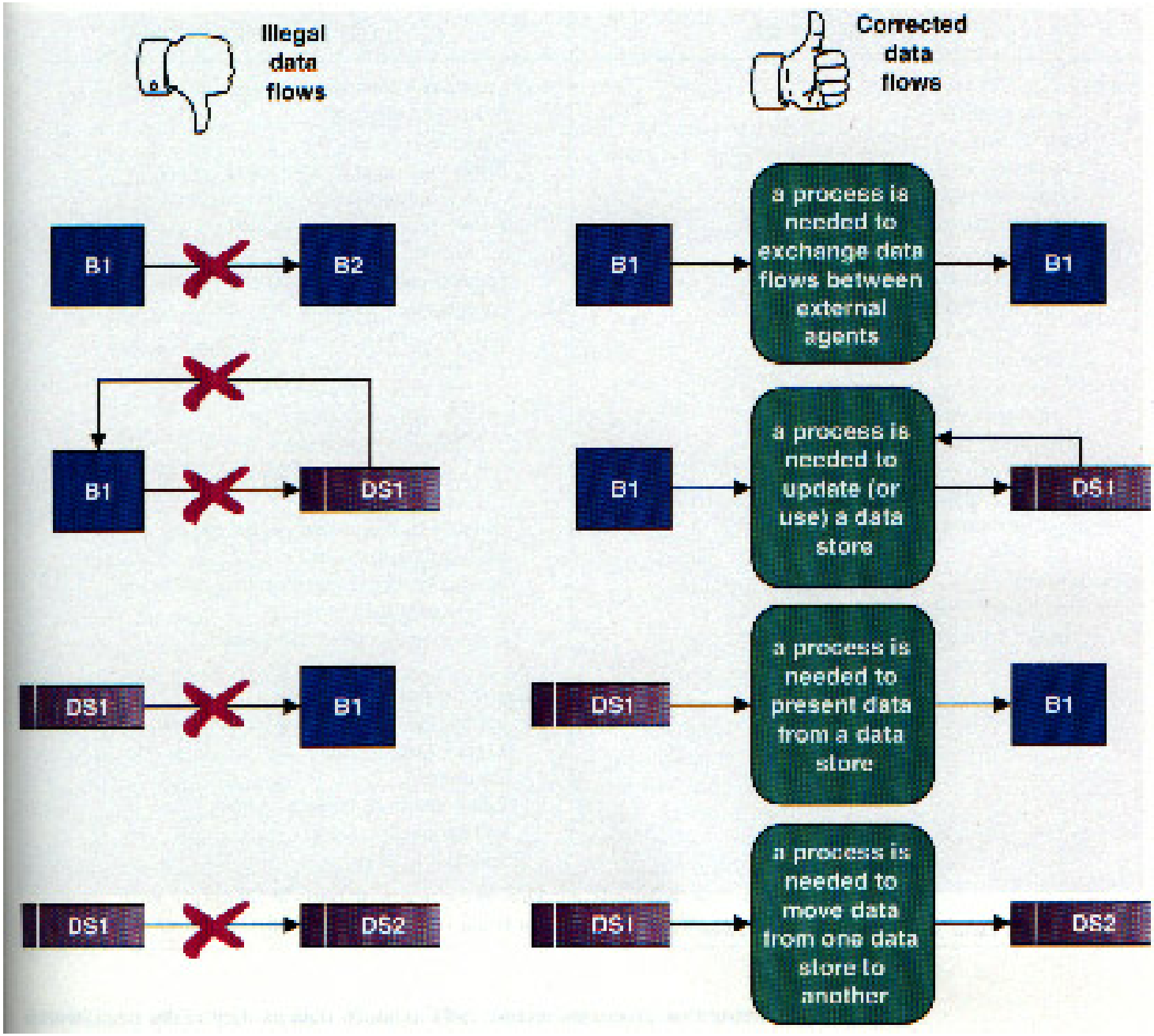
Level 1 Tips

- Generally move from top to bottom, left to right
- Minimize crossed lines
- Iterate as needed
  - *The DFD is often drawn many times before it is finished, even with very experienced systems analysts*

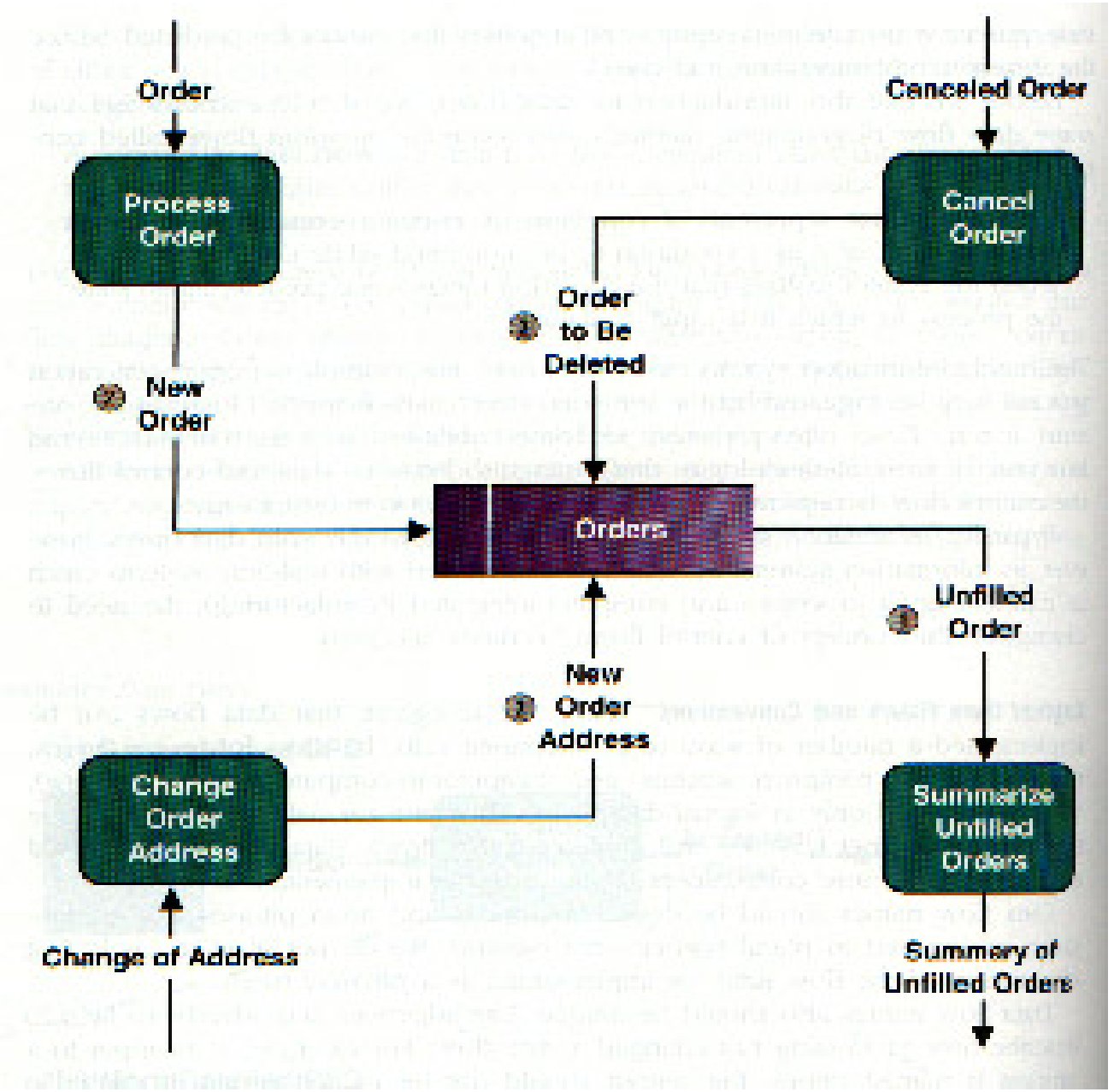
Tips for Level 2 and Below

- Sources for inputs and outputs listed at higher level
- List source and destination of data flows to processes and stores within each DFD
- Depth of DFD depends on overall system complexity
  - Two processes generally don’t need lower level
  - More than seven processes become overly complex and difficult to read

Illegal Data Flows



Flows to & from Data Stores



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

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

Data Flow Diagramming Rules

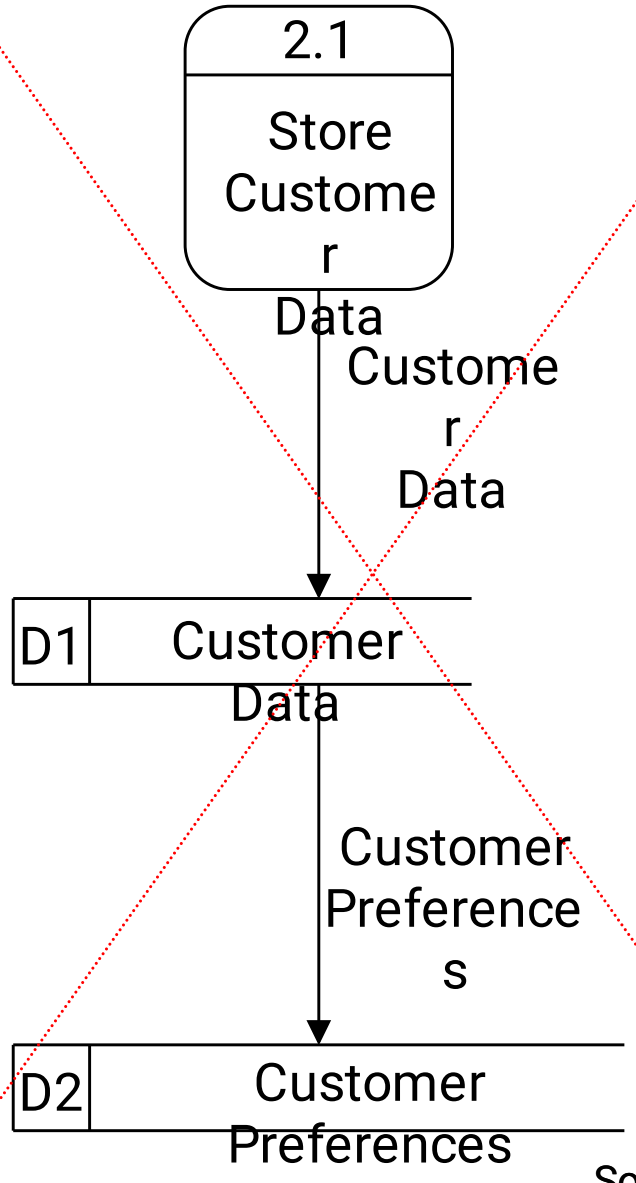
- **Data flows**
  - data flows are unidirectional
  - a data flow may fork, delivering exactly the same data to two different destinations
  - two data flows may join to form one only if the original two are exactly the same
  - no recursive data flows
  - data flows (and data stores and sources/sinks) are labeled with noun phrases
  - The inputs to a process are different from the outputs
  - Every object in a DFD has a unique name
  - 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

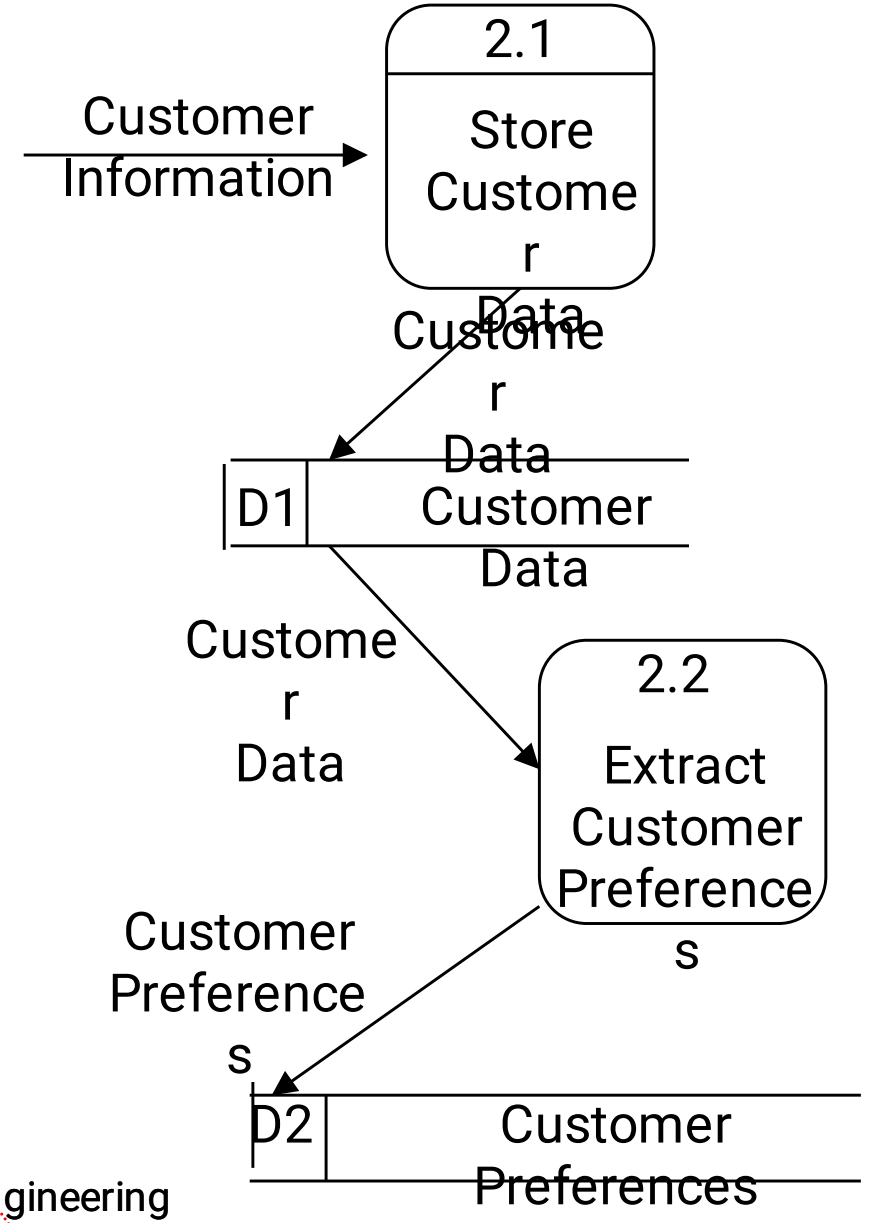
Simple Guidelines

- the level 0 data flow diagram should depict the software/system as a single bubble
- primary input and output should be carefully noted
- refinement should begin by isolation candidate processes, data objects, and stores to be represented at the next level
- all arrows and bubbles should be labeled with meaningful names
- information flow continuity should must be maintained from level to level
- one bubble at a time should be refined.

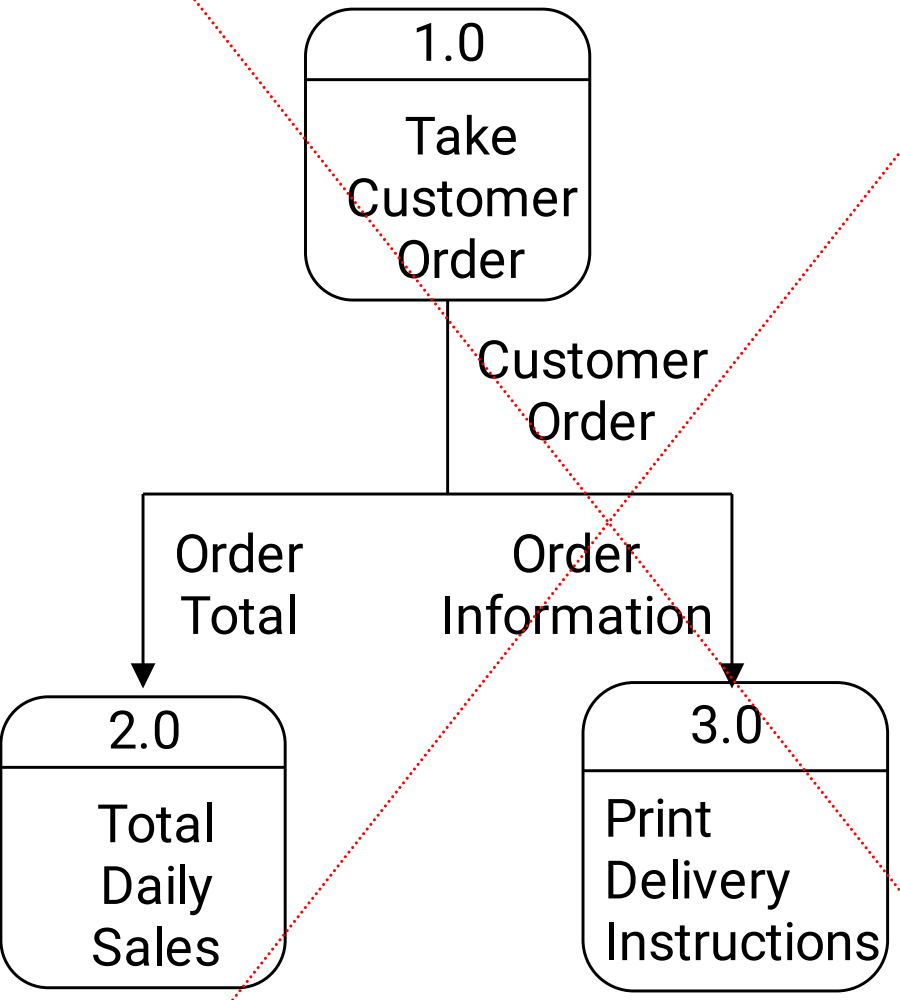
Examples

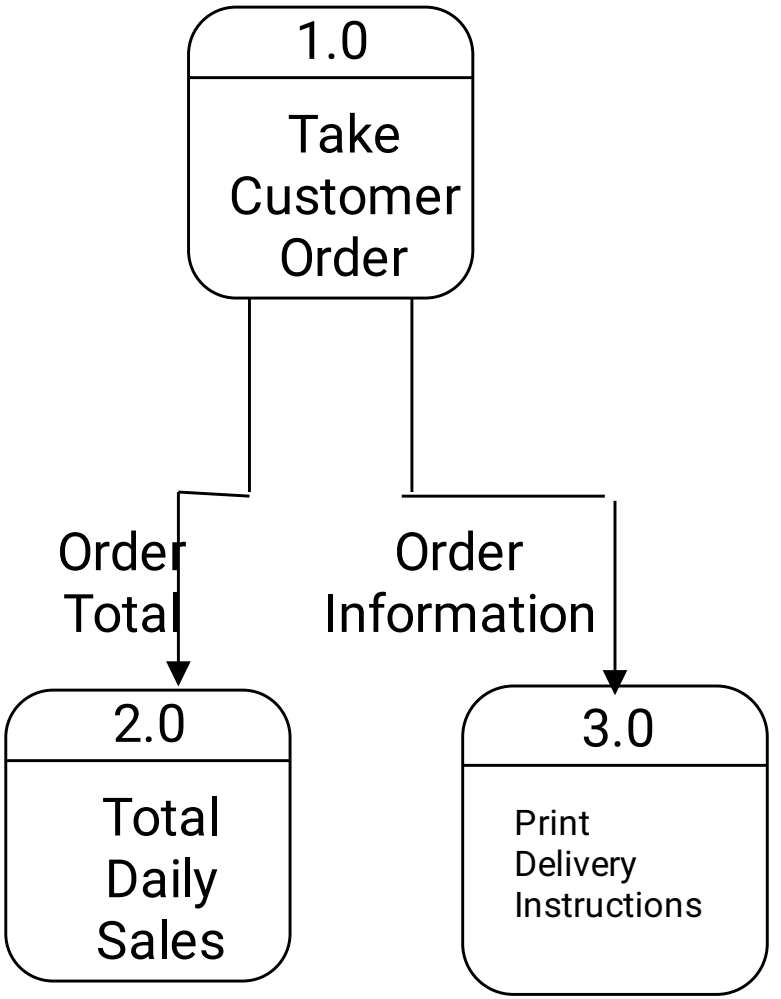
Examples


2.1
Store Customer Data
Customer Data
D1
Customer Data
Customer Preferences
D2
Customer Preferences

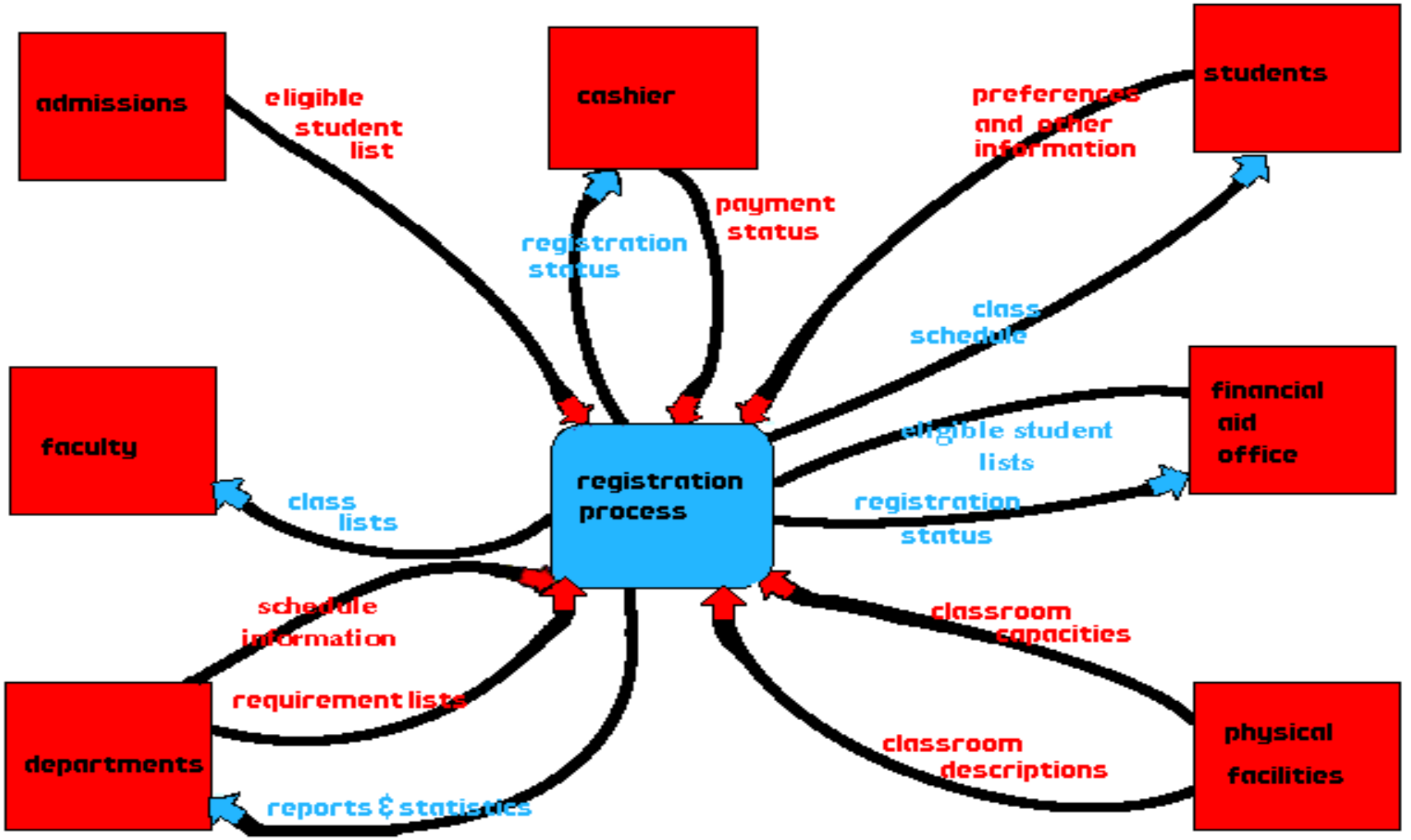

Customer Information
2.1
Store Customer Data
Customer Data
D1
Customer Data
Customer Data
2.2
Extract Customer Preferences
Customer Preferences
D2
Customer Preferences

Examples

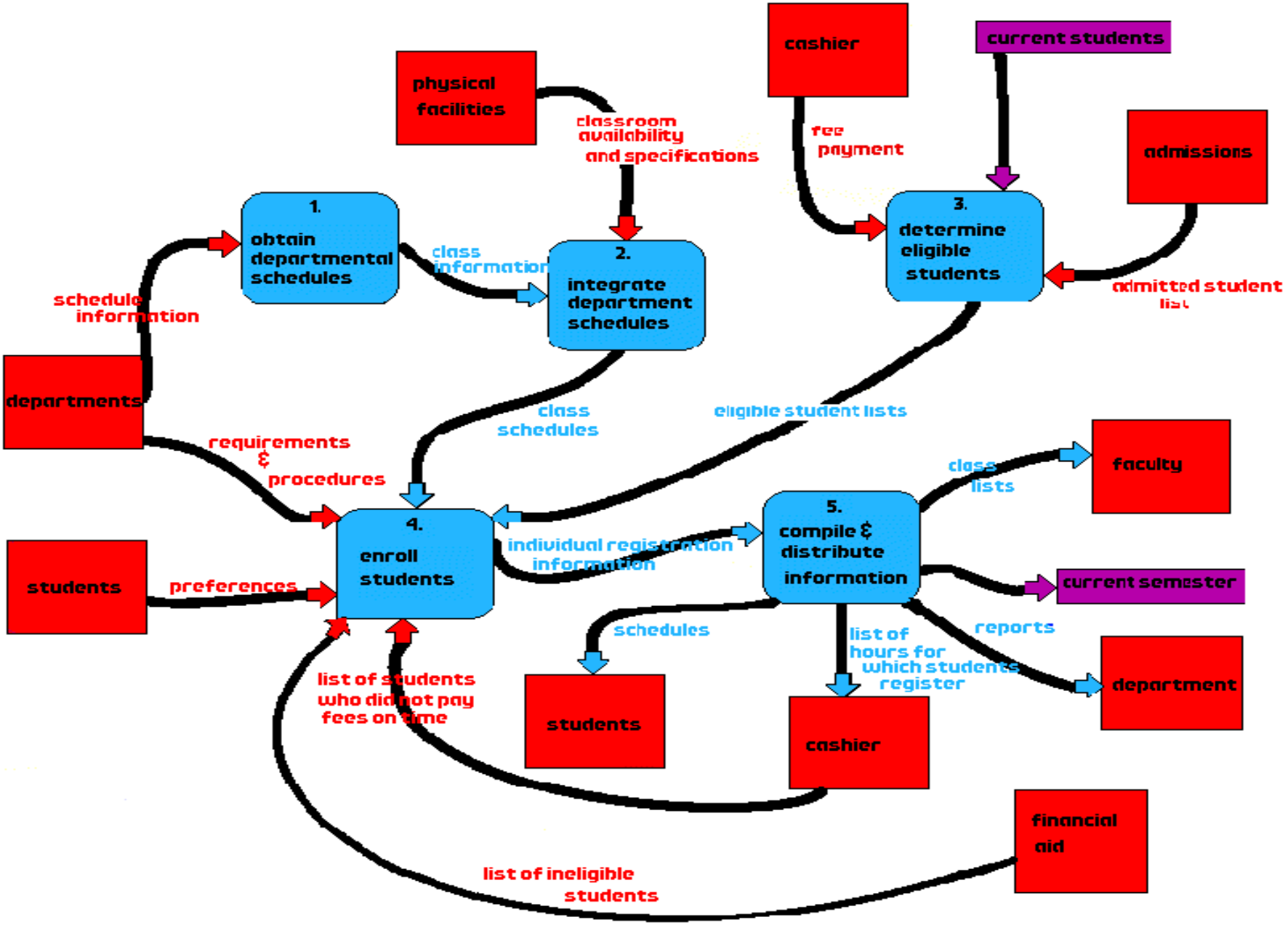

1.0
Take Customer Order
Customer Order
Order Total
Order Information
2.0
Total Daily Sales
3.0
Print Delivery Instructions


1.0
Take Customer Order
Order Total
Order Information
2.0
Total Daily Sales
3.0
Print Delivery Instructions

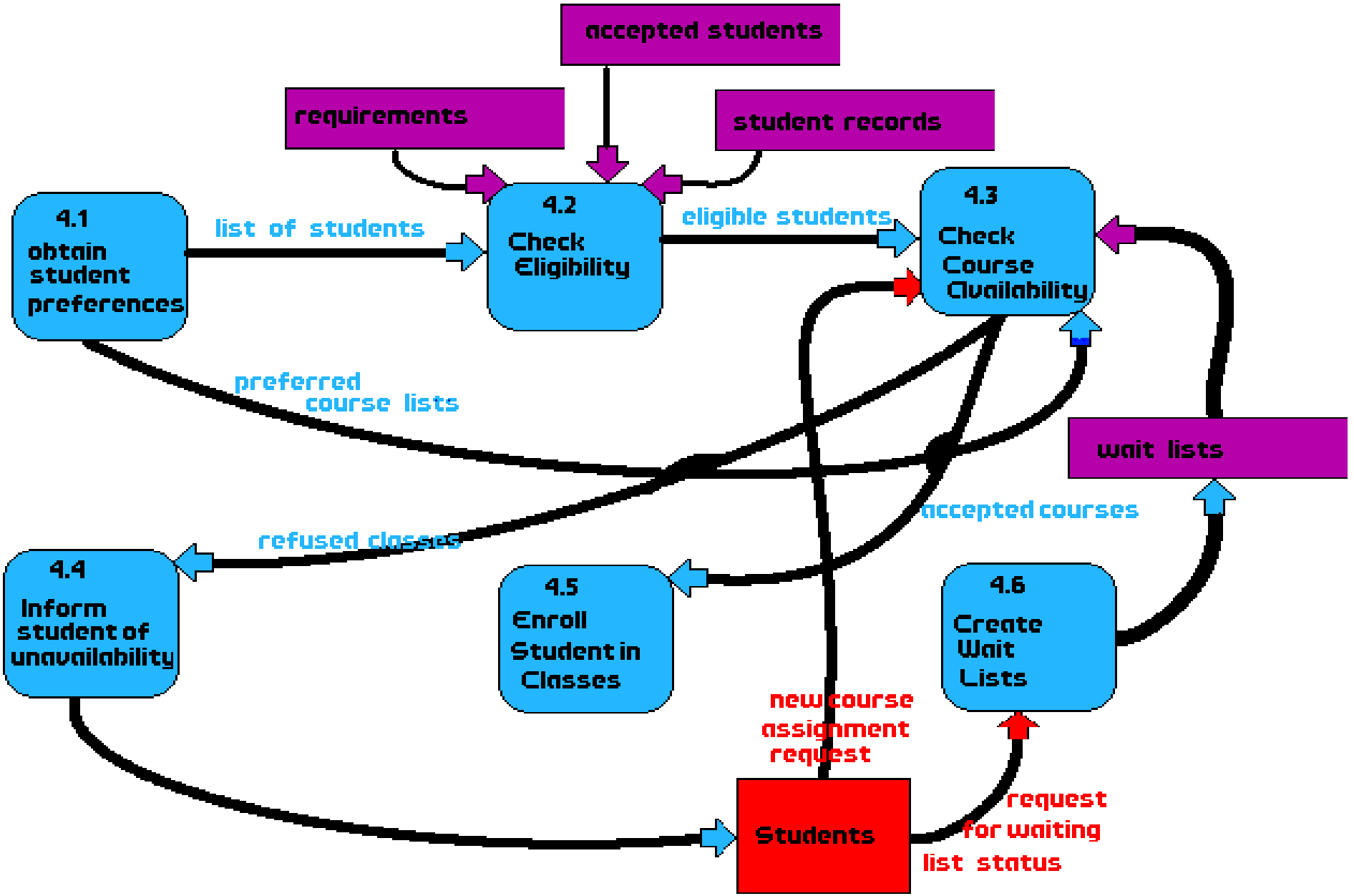
Example :To demonstrate the flows of information for *the registration process*. The first of these is the *context diagram*



Level 1 Diagram



Explosion of Process 4, **Enroll Students.**



Validating the DFD

- Syntax errors
  - Assure correct DFD structure
- Semantics errors
  - Assure accuracy of DFD relative to actual/desired business processes
- User walkthroughs
- Role-play processes
- Examine lowest level DFDs
- Examine names carefully

How should we gather requirements? Use cases?

- Use cases hold functional requirements in an easy-to-read text format
- They make a good framework for non-functional requirements & project details.
- Use cases show only the Functional requirements.

Summary

- The Data Flow Diagram (DFD) is an essential tool for creating formal descriptions of business processes and data flows.
- Use cases record the input, transformation, and output of business processes.
- Eliciting scenario descriptions and modeling business processes are critically important skills for the systems analyst to master.