Structured System Analysis

What is SSA?

- A method for analyzing and converting business requirements into specifications and ultimately computer programs, hardware configuration and related manuals.
  - Structured Analysis became popular in the 1980s and is still used by many.
  - SA were accompanied by different notational methods developed by DeMarco, Ed Yourdon, et al, which includes:
    - data flow diagrams
    - data models
    - Structured charts
    - Etc.

Start with System Level Analysis

- Model the Information Domain
  - Consider the the system under the study as a "Black-box":
  - Identify Source Terminators: Any entity that triggers the system:
    - People
    - Other systems
    - Devices
    - Hidden stimuli
  - Identify Sink Terminators: Any entity that receive something from system:
    - People
    - Devices
    - Etc..
- Draw the highest level Data Flow Diagram (DFD), called a "Context Diagram".

Analysis Principles

- Model Processes and Flow of Data
  - identify functions that transform data objects
- Model the Information Domain
  - define data objects
  - establish data relationships
  - specify data content
- Partition the Models
  - refine each model to represent lower levels of abstraction

Structured System Analysis Element

- Core element is data dictionary (DD), extracted from diagrams such as:
  - Data Flow Diagram (DFD)
  - Entity Relationship Diagram (ERD)
  - State Transition Diagram (STD)
- Fundamental outcome is a Software Requirements Specification Document that include:
  - Systems description (Scope, assumptions, constraints, etc.
  - Process Specification (PSPEC), Control Specification, CSPEC (if applicable).
  - All relevant diagrams that helps expressing system's requirements (DFD, ERD, STD, DD, etc.)

Data Flow Diagram

Focus of a DFD

- Models functions performed by the system.
- Models the interaction between functions:
  - Data passed between modules.
- Shows data transformation performed by the system:
  - What inputs are transformed into what outputs?
- Shows what kind of work is done by the system.
- Shows sources and destinations of data.

Start with System Level Analysis

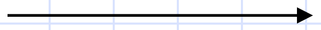
- Model the Information Domain
  - Consider the the system under the study as a “Black-box”:
  - Identify Source Terminators: Any entity that triggers the system:
    - People
    - Other systems
    - Devices
    - Hidden stimuli
  - Identify Sink Terminators: Any entity that receive something from system:
    - People
    - Devices
    - Etc..
- Draw the highest level Data Flow Diagram (DFD), called a “Context Diagram”.

DFD Components

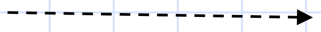
- Data Flow Diagram components include:
  - Process
  - Data/control Flow
  - Data Source / Data Sink (Terminators)
  - Data Store
- Narrative or textual description of data flows and data stores are in data dictionary (normally at the lowest level).
- Description of processes are in process specifications.

DFD Components

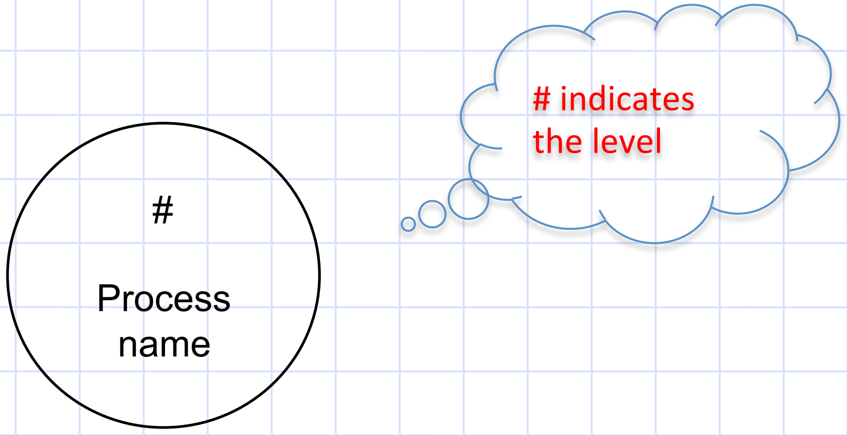
Data Flow:



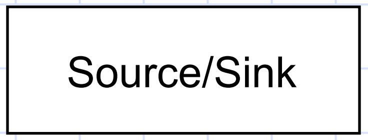
Control Flow:



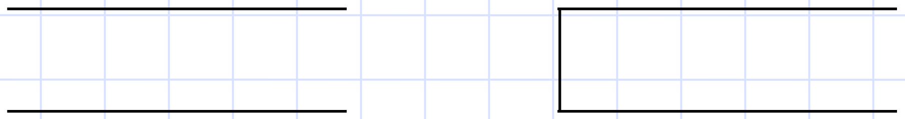
Process:



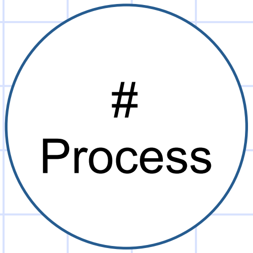
Terminator:



Store:



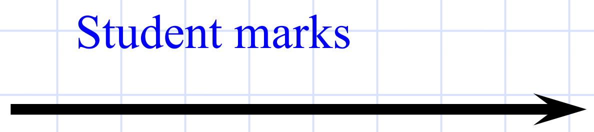
Processes



- Transforms inputs to outputs.
- Name describes the transformation: should be **verb-object phrase**.
  - Example: Receive Order
- Should be numbered.
  - Sequence of processing is not implied by the numbering scheme.
  - Used only as a basis for hierarchical numbering in data flow leveling.

Data/control Flows

- Represents movement of information.
  - Shows the direction of the flow
- Must have label and it must be noun
- Eg:
  - Student marks
  - Order
  - invoice
  - Receipt
  - payment



High-Level (System-Level) DFD

System Level DFD:

- The highest level of the DFD is called **Context** diagram
- System is viewed as a black box
- Identifies the system's boundary
  - Terminators are the outside the system bounder
- Identifies the system interfaces
- Identifies the flow of data into and out of the system
- No details about the system's internals
  - No data stores

Start with List of System's Stimuli

Those entities that trigger the system:

- People
- Devices
- Other Systems
- Time

Event List

Then, for each entity that triggers the system, come up with a list of required functions.

- There are 3 types of events:
  - Flow
  - Temporal
  - Control
- Flow-oriented events are triggered by the arrival of data.
  - Are randomly timed, and can't be predicted.
    - E.g. Patron borrows book.
    - E.g. Heat sensor detects rise in temperature.

Event List (continued)

- Temporal events happen at a scheduled time, and don't involve the transfer of data.
  - E.g. Manager requires month-end report.
- Control events can happen at any time, and don't involve the transfer of data.
  - Are either on or off, and signal the system to take immediate action.
  - Most common in real-time systems.
  - E.g. User turns system on.

Examples

- Problem I – An Automated Food Ordering System (AFOS)
  - An Automated Food Ordering System (AFOS), that allows customers place an order, supplier receive food and material orders, kitchen receives customer's order managers produce inventory report. Also system is supposed to be equipped with a fire alarm in case that kitchen goes on fire.
- Problem II – A Single Tower Elevator (STE)
  - The purpose of this simplified example of an elevator system is to schedule and control an elevator in a building with 4 floors to carry people from on floor to another in a conventional way.

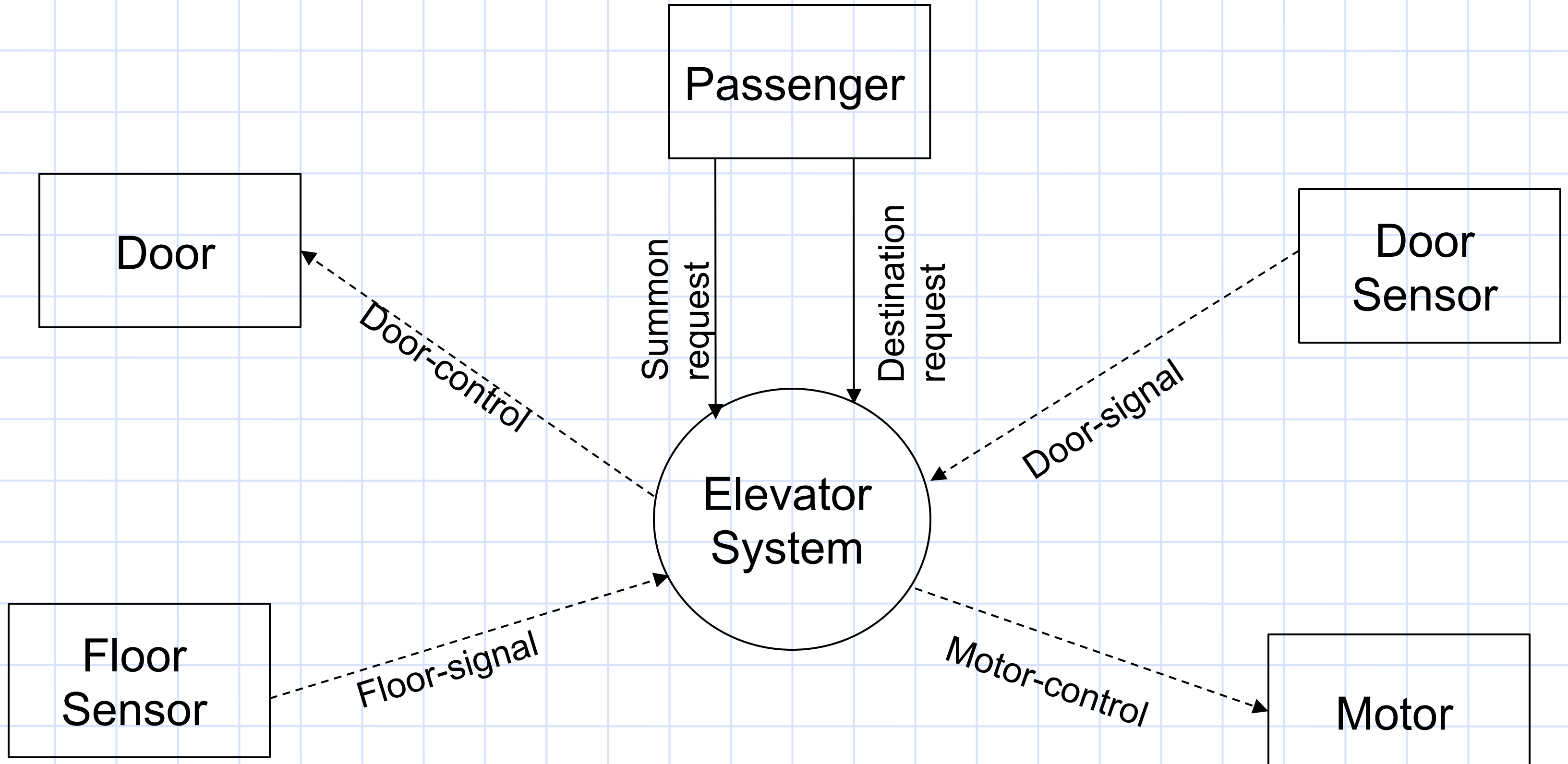
Start with list of stimuli

- Things that invokes a specific function
  - Problem I:
    - Customer
    - Cook
    - Manger
    - Fire Sensor
    - Time
    - Supplier
  - Problem II:
    - Door-close button
    - Door-open button
    - Destination button
    - Floor summon button
    - Stop button
    - Motor
    - Fire Sensor
    - Floor Sensor

A Partial Event List

- Passenger Samoans the elevator.
- Floor sensor captures the arrival of the elevator at the floor.
- Door opens.
- Passenger requests for a destination.
- Elevator motor moves the elevator to the destination.

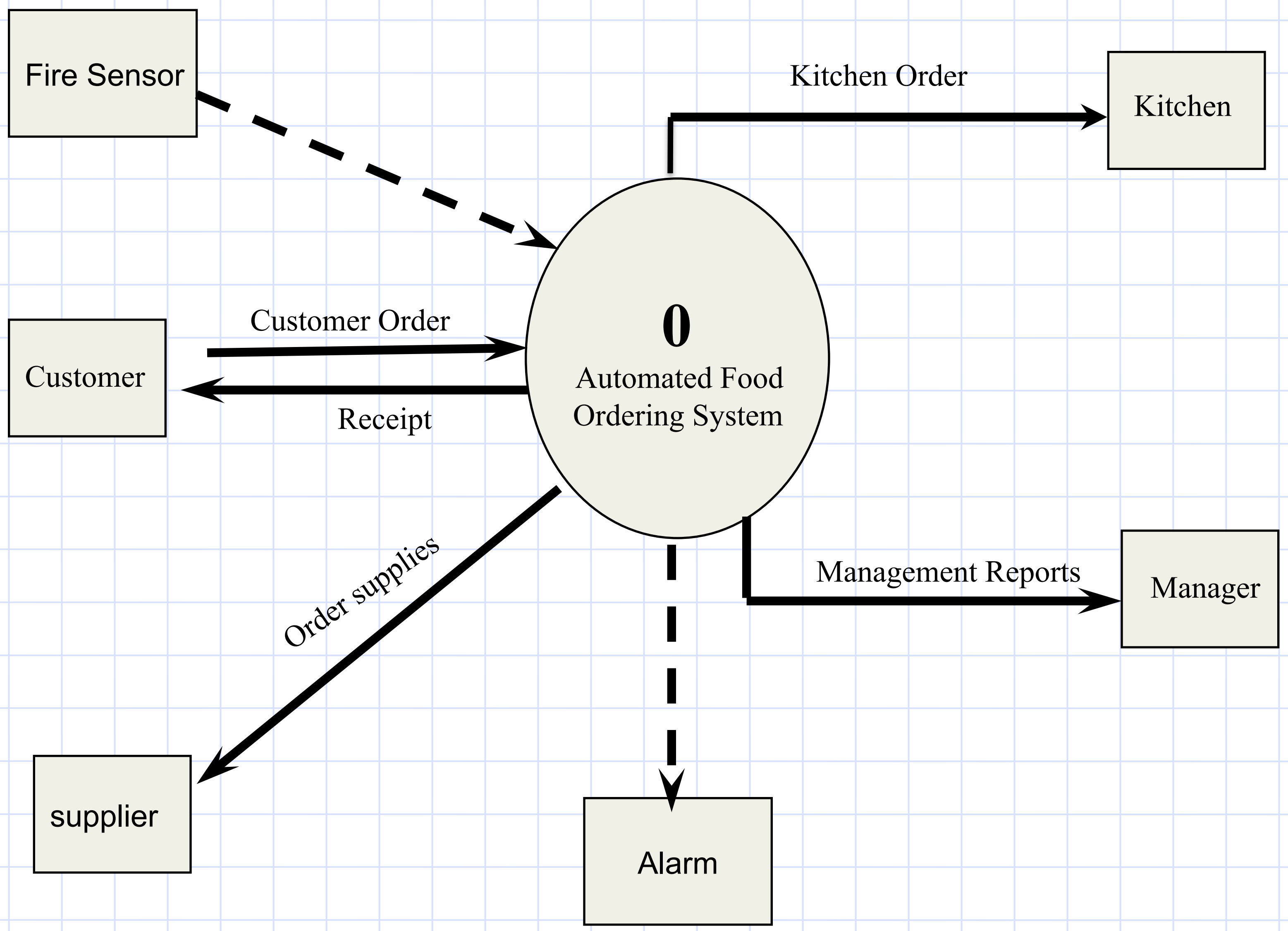
Simplified Context Diagram for Elevator System



Event List

- For each stimulus entity find as many as need events, and place them in different categories (F, T, or C)
  - customers place an order
    (F)
  - supplier receive food and material orders
    (F)
  - kitchen receives customer's order
    (F)
  - managers produce inventory report.
    (T)
  - Fire alarm goes of in case of fire or smoke
    (C)

Level 0 DFD (context diagram) – First Draft



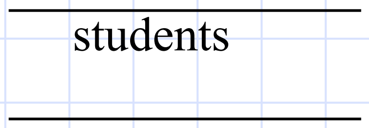
Lower Levels of Analysis

System Decomposition

- Identify the processes needed for interactions between the system and its outside world.
  - Normally at least one process for each terminator
  - Label each process with a number, starting with number 1.
    - Note: numbers do not represent a sequence
- Label each process with a verb-noun phrase.
  - Use terminology familiar to your client
  - Examples:
    - Place order
    - Read flow-data
- Add data stores, as needed.
- Add the description of processes in the process-specifications.

Data Stores

- Collection of data packets at rest.
- Name is plural of data packets.
- May take many physical forms.
- May exist because of:
  - Client requirements
  - Convenience of implementation



Flows From and into a Store

- A flow from a store may be:
  - An entire packet.
  - Many packets.
  - Parts of packets.
- Contents of the store are not modified by a flow from a store.
  - Does a non-destructive read of information in the store.
- A flow into a store modifies the contents of the store.
  - Writes information to, or deletes information from, a store.

Decomposition of DFDs: Leveling

- Each level provides more detail about part of the level above it.
- Context diagram is the highest level in a DFD, and represents the entire system.
- Level 1 is the level immediately below the context diagram, and shows all major functions of the system.

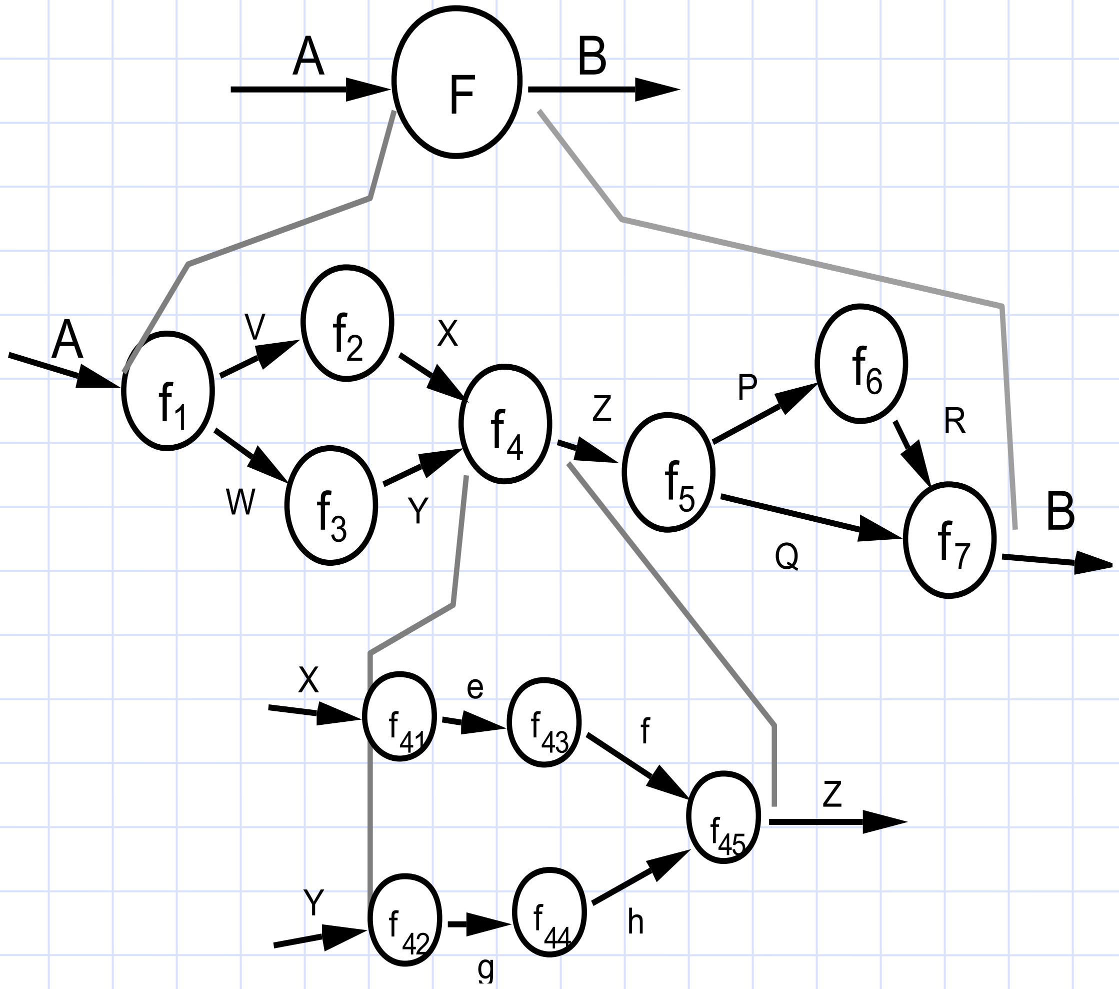
Decomposition… (continued)

- Level-n diagrams are a set of DFDs representing the decomposition of the functions represented at the level immediately above.
- Numbers assigned to processes show their hierarchical organization in the DFD.
- Different audiences will be interested in different levels.

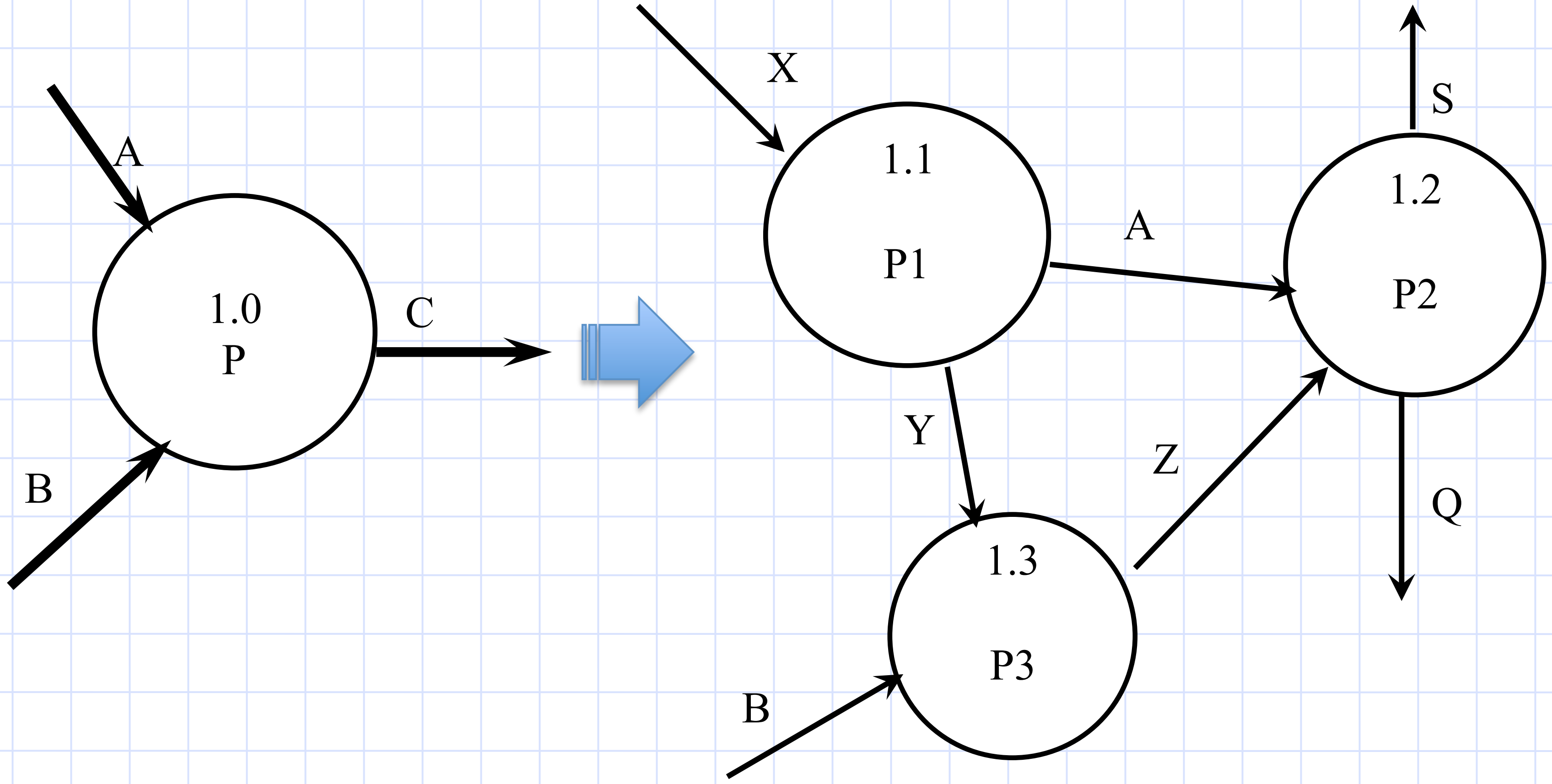
Balancing to Lower levels

- The number and direction of data flows associated with a process at one level must correspond to the number and direction of data flows for the decomposition of that process.
- *A* discrepancy may imply that "this" level is wrong, or that some higher level(s) is wrong (e.g. missing information to or from a terminator).

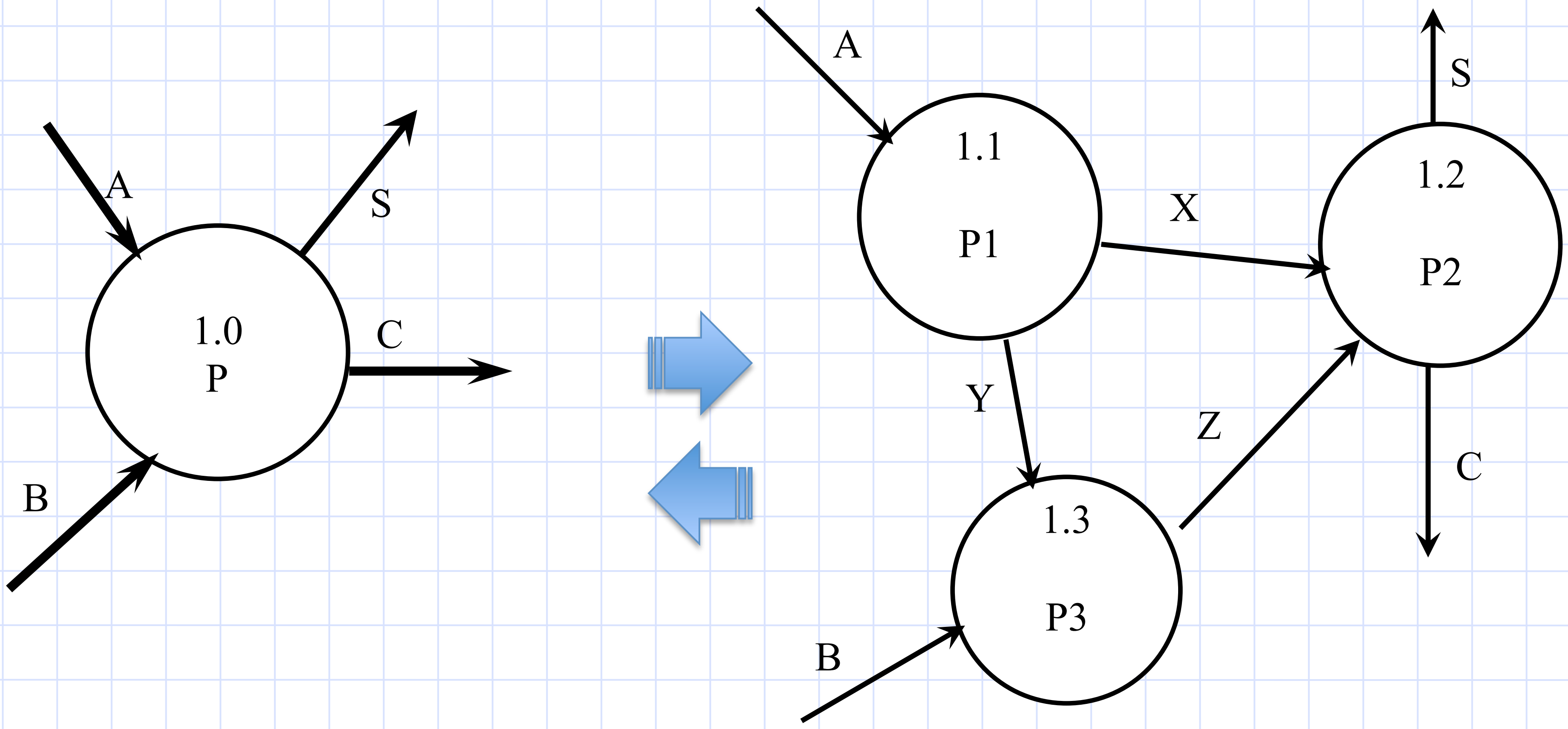
Data Flow Hierarchy



Reference: Software Engineering A Practitioner's Approach, McGraw Hill, 5th Edition


A
1.0
P
C
B
X
1.1
P1
A
1.2
P2
S
Y
Z
Q
1.3
P3
B

Balanced DFD



Check for Models Consistency and Correctness

- Assure technical correctness.
- Make it acceptable to the client.
- Aesthetically pleasing:
  - Use consistent diagramming notations.
  - Use consistent size and shape for processes.
  - Use curved or pipeline flows consistently.
  - Avoid crossed flows.

Check for Models Consistency and Correctness

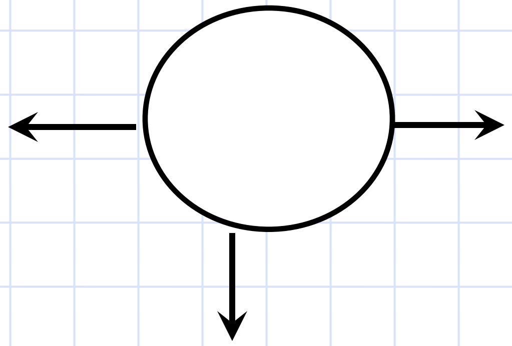
- Make sure your diagrams are correct, complete, and consistent:
  - Beware of unlabeled elements in a DFD.
  - Beware of read-only or write-only stores.
  - Balance between levels of DFDs.
  - Avoid direct communication among data stores
    - Only processes can be means of data communication
  - Avoid direct communication between terminators
  - Avoid direct communication between data stores and terminators

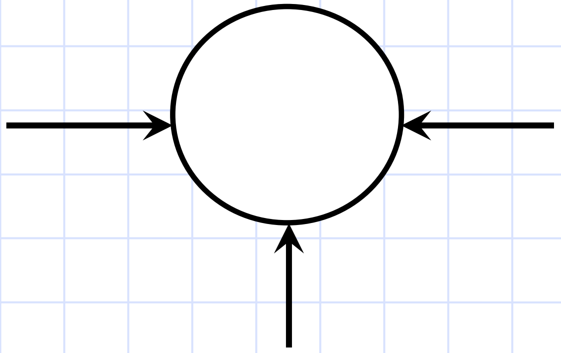
Check for Models Consistency and Correctness

- Each input flow should be needed by the system to recognize a stimulus, or to produce a response to an event.
- Each output flow should be a response to an event (either flow or temporal).
- Each non-temporal event should have an associated stimulus.
- Each event should either produce immediate output, or be stored for later output, or cause a system state change.

Logically Consistent

- Double check “black hole” processes.
- Double check “spontaneous generators”





Data Stores in Leveling

- A data store appears at the first level in which it functions as an interface between two processes.
- The input to or output from a data store appears on every deeper level describing those processes.

Levels Required

- small systems: 1 level
- Medium size systems: 1 or 2 levels
- Large systems: 2 - 3 levels
- Very large systems: 3 - 8 levels