Process & Data Oriented Analysis and Design

System Level Analysis

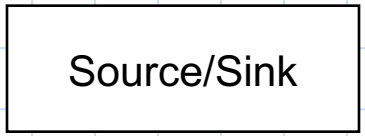
- A common methodology to analysis a system as a collection of data or control processes, and data stores, is known as Structured System Analysis and Design (SSAD)
  - Starts with the idea of considering the the system as a "Black box", that interacts with its surrounding environment. The elements of the surrounding environment that interact with the system are known as terminators.

SSAD Notations

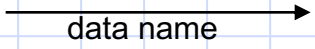
- The focus of this methodology is to identity the flow of data or flow of control from sources (terminators that act as a stimuli to trigger an interaction), to sinks (terminators that receive the data or a control signal). There are a number important notation involved in this methodology:
  - Terminators
  - Process
  - Data/control flow
  - Data store

SSAD Notations

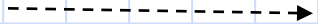
Terminator:



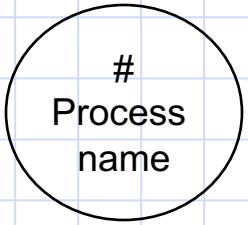
Data Flow:



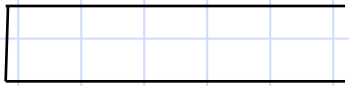
Control Flow:



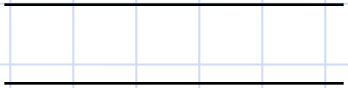
Process:



Data Store:



Alternative notation for data store



Process

- 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/control signal.
  - Shows the direction of the flow
- Data flows need to be labeled. But not the controls:
  - Temperature data, or sensor signal
  - Command
  - Name, address, etc.

How to Start

- Identify Source Terminators: Any entity that triggers the system:
  - A Sensors
  - A User
  - Another system
  - A Device: Clock, button, etc
- Identify Sink Terminators: Any entity that receive something from system:
  - An actuator
  - Another system
  - A Device
  - A Person

Analysis Principles

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

Data/Control Flow Diagram

Focus of a DFD

- Models functions performed by the system.
- Models the interaction between functions:
  - Data/control 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/control.

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.

Process Oriented System Analysis and Design

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 for flow of conrol into and out of the system
- No details about the system's internals

Provide an Event List

- A narrative list of the stimuli which occur outside of the system to which the system must respond.
- Usually phrased terminator-action.
  - E.g.
    - Thermometer checks the temperature
    - Camera sends an image
    - Customer pays bills
    - Passenger summons elevator

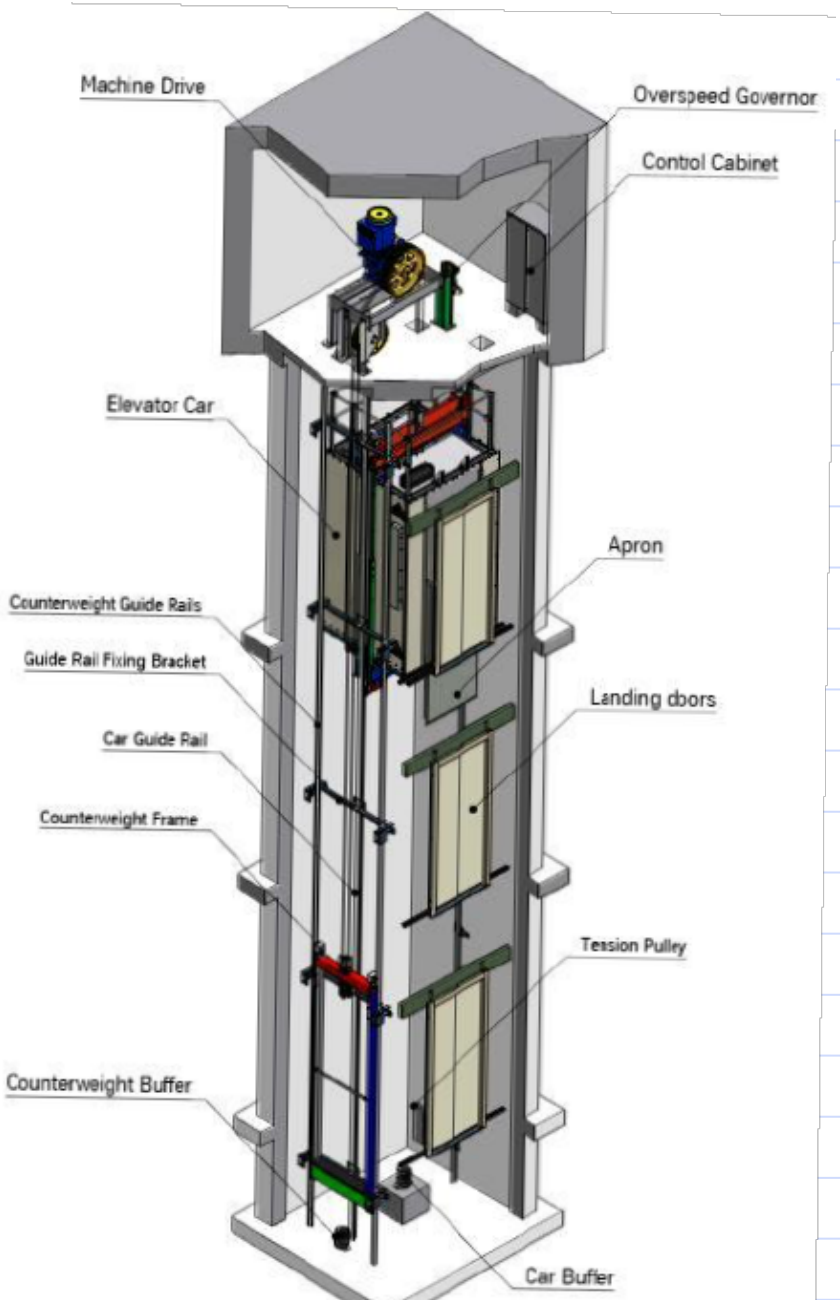
Different Type of Events

- There are three types of events:
  - Control event: sensor or other electronic device event
  - Data flow event: normally created by human, device, other systems
  - Temporal event: time dependent event
- Normally Temporal events happen at a scheduled time,
  - E.g. Manager requires month-end report at 12:00 PM of last Friday of the month.
- 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 embedded and real-time systems.

A Simplified Single Tower Elevator System

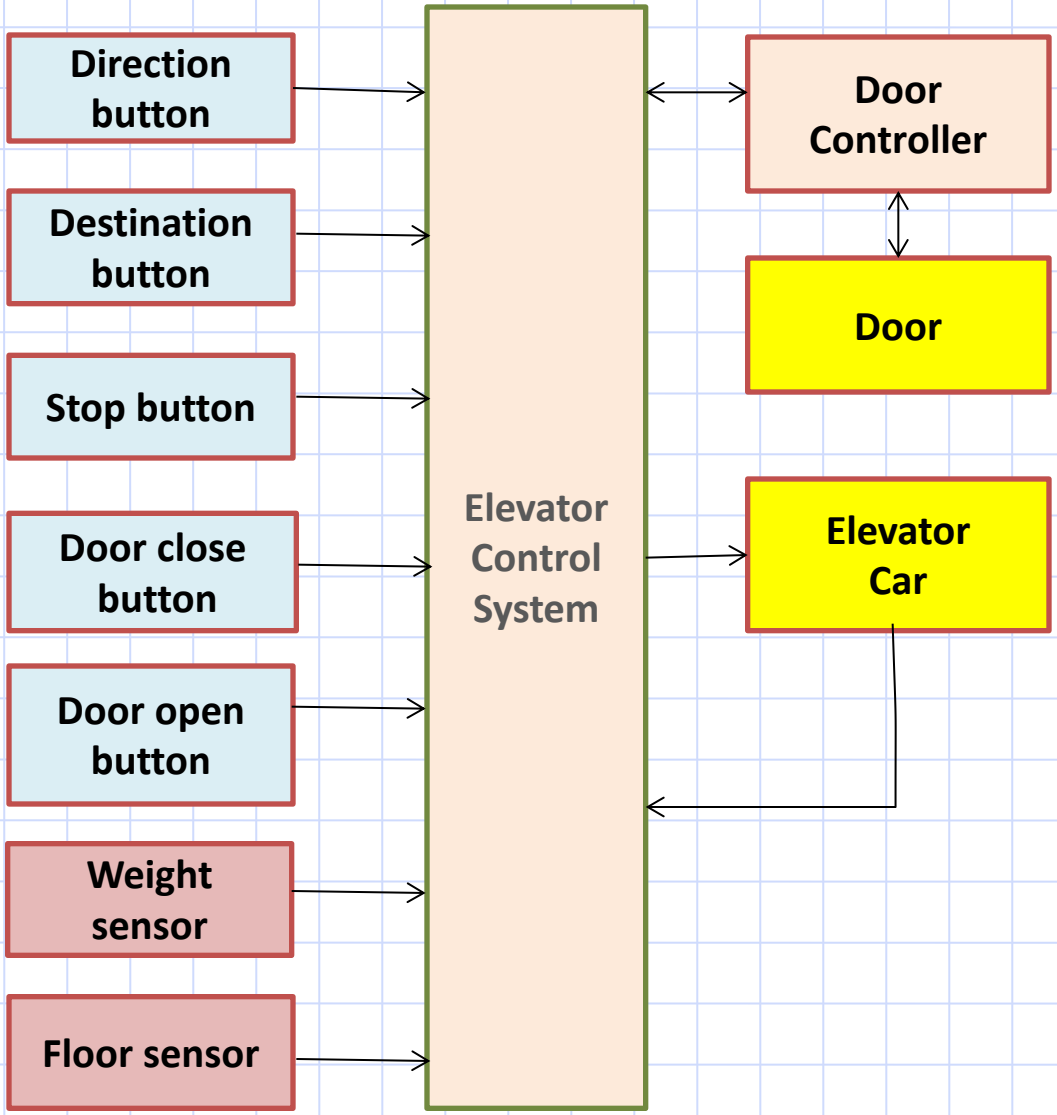
- **Problem Statement:**

  The purpose of this simplified example of an elevator system is to schedule and control an elevator in a building with a few floors to carry people from on floor to another in a conventional way.



Picture from webpage: http://www.electrical-knowhow.com

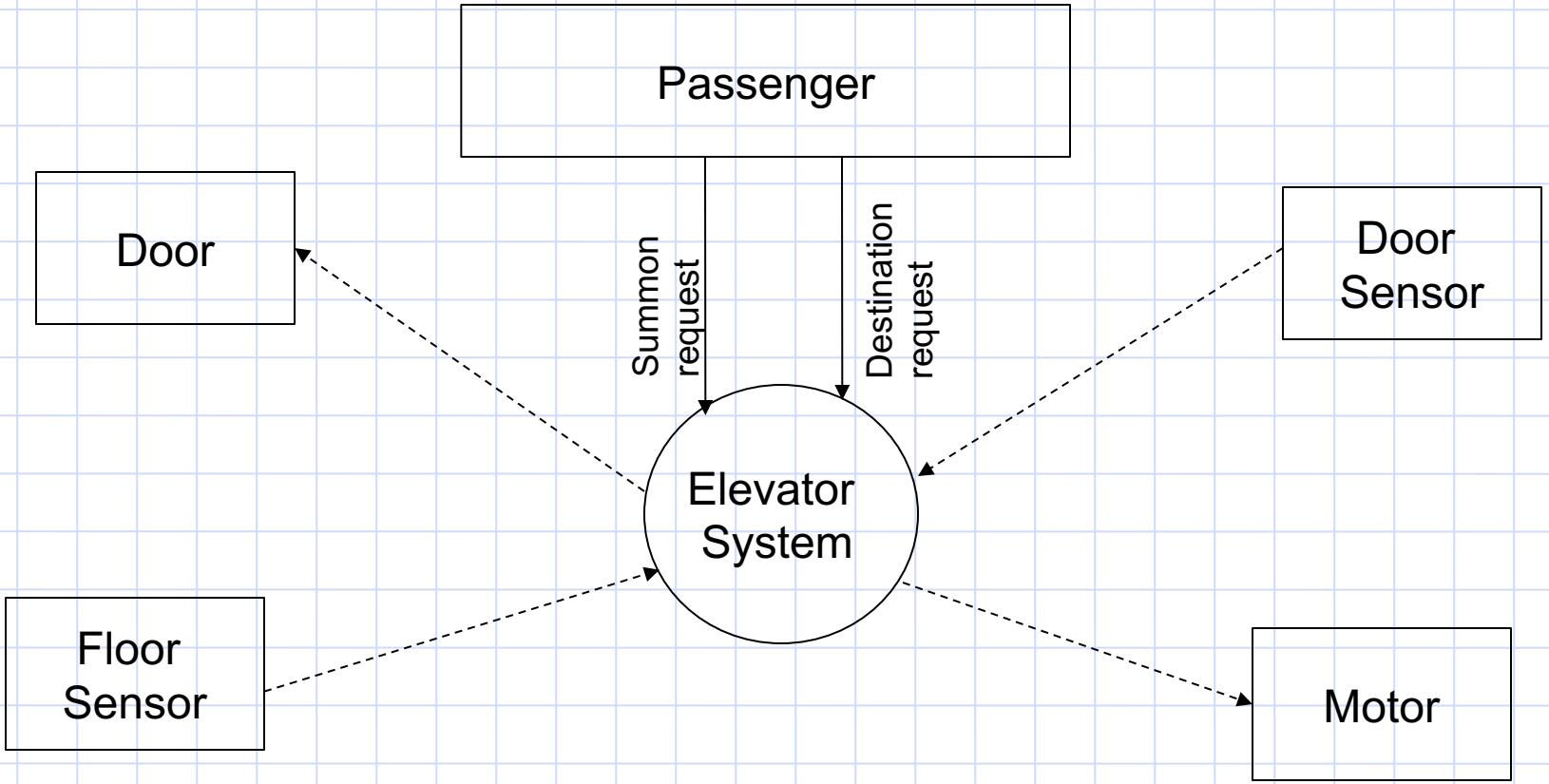
First step: developing the the system's block diagram



A Partial Event List

- Passenger Summons the elevator.
- Floor sensor captures the arrival of the elevator at the floor.
- Passenger opens the door.
- Passenger requests for a destination.
- Elevator motor receives signal and direction to move the elevator to the destination.

Simplified Context Diagram for An Elevator System



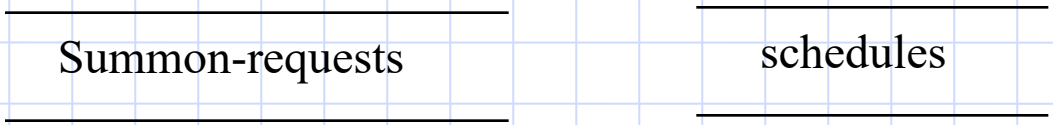
Lower Levels of Analysis

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:
    - Press open-door button
    - Read schedule
- Add data stores, as needed.
- Add the description of processes in the process-specifications.

Closer Look at Data Stores

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



Flows from and into a atore

- 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.

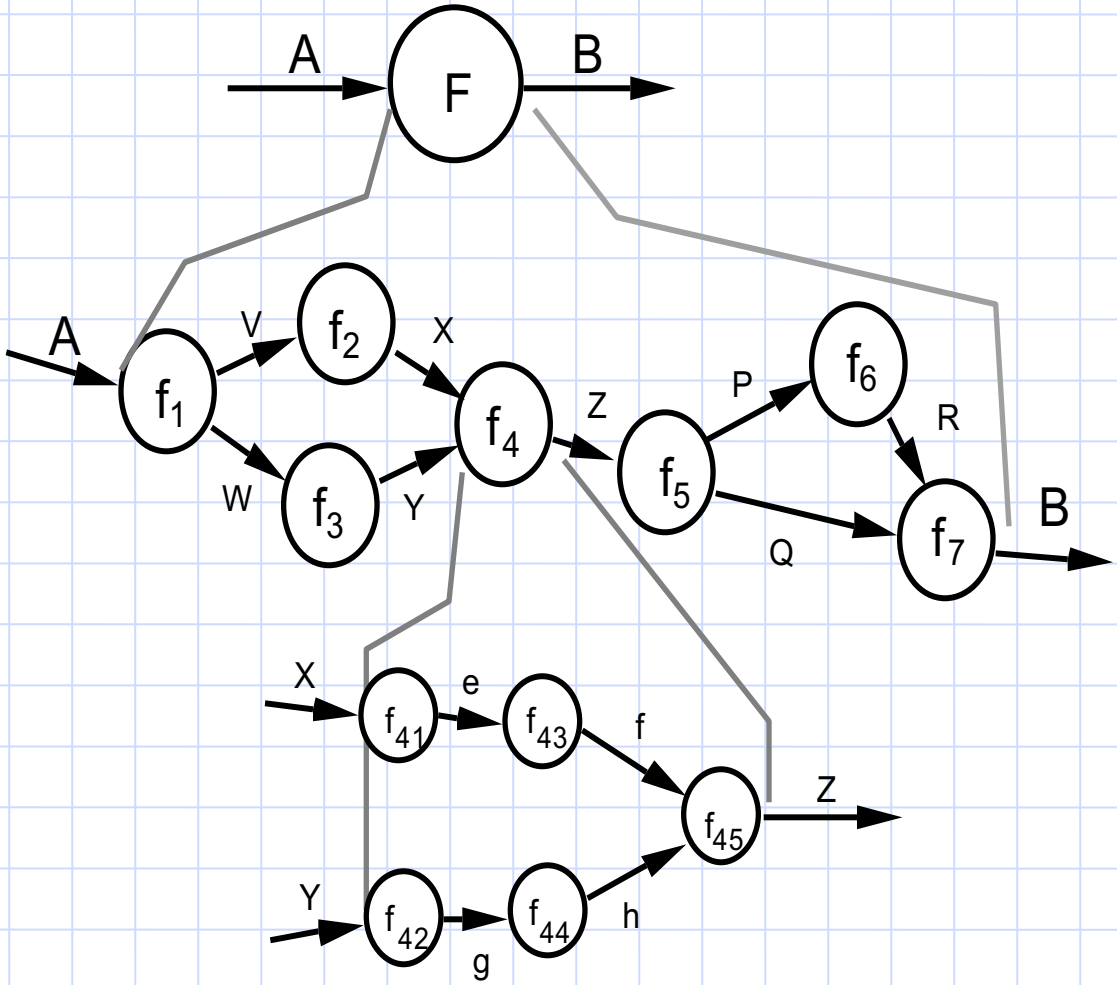
Decomposition… (continued)

- 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.

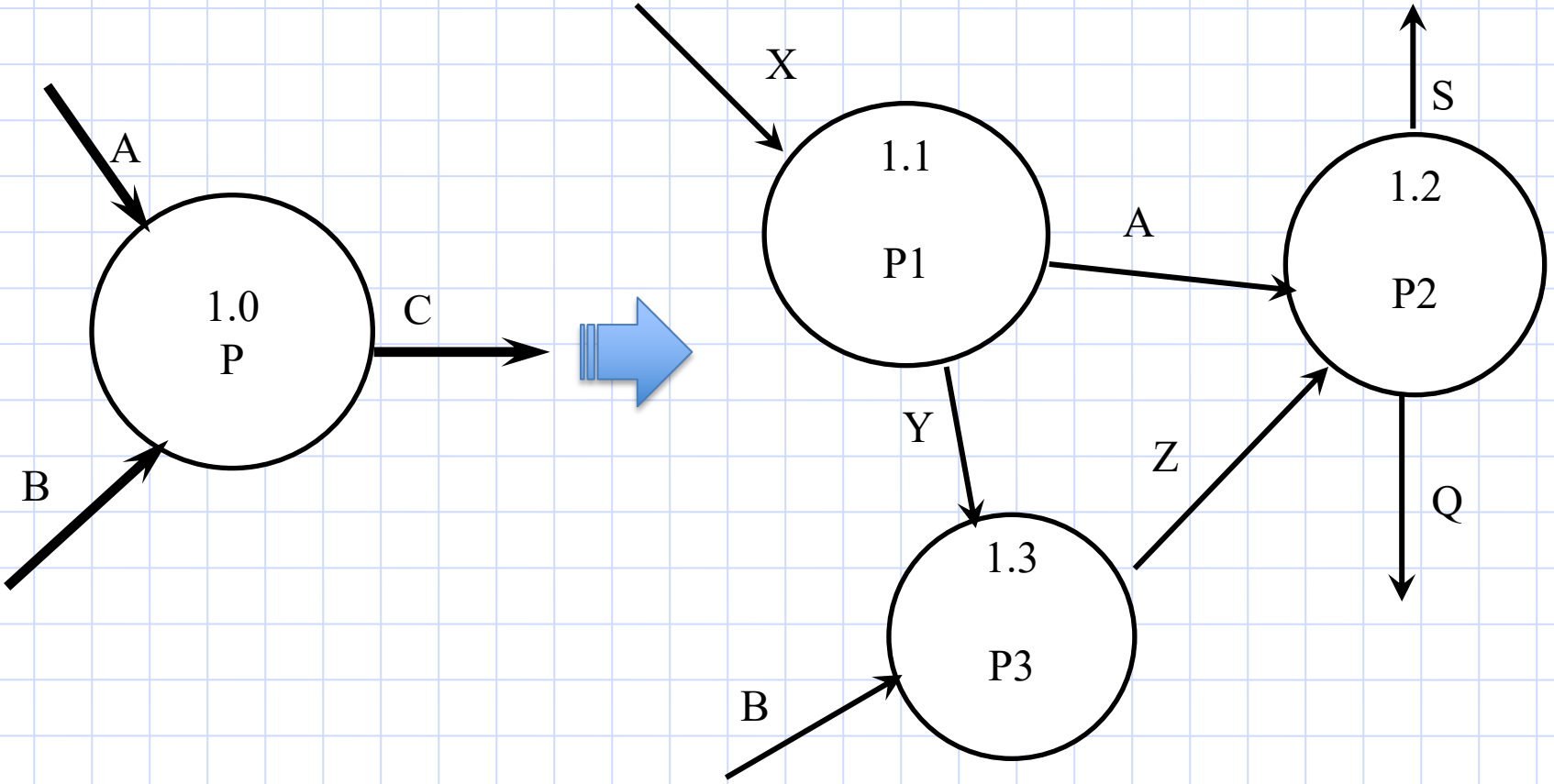
Balancing to Lower levels

- 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).

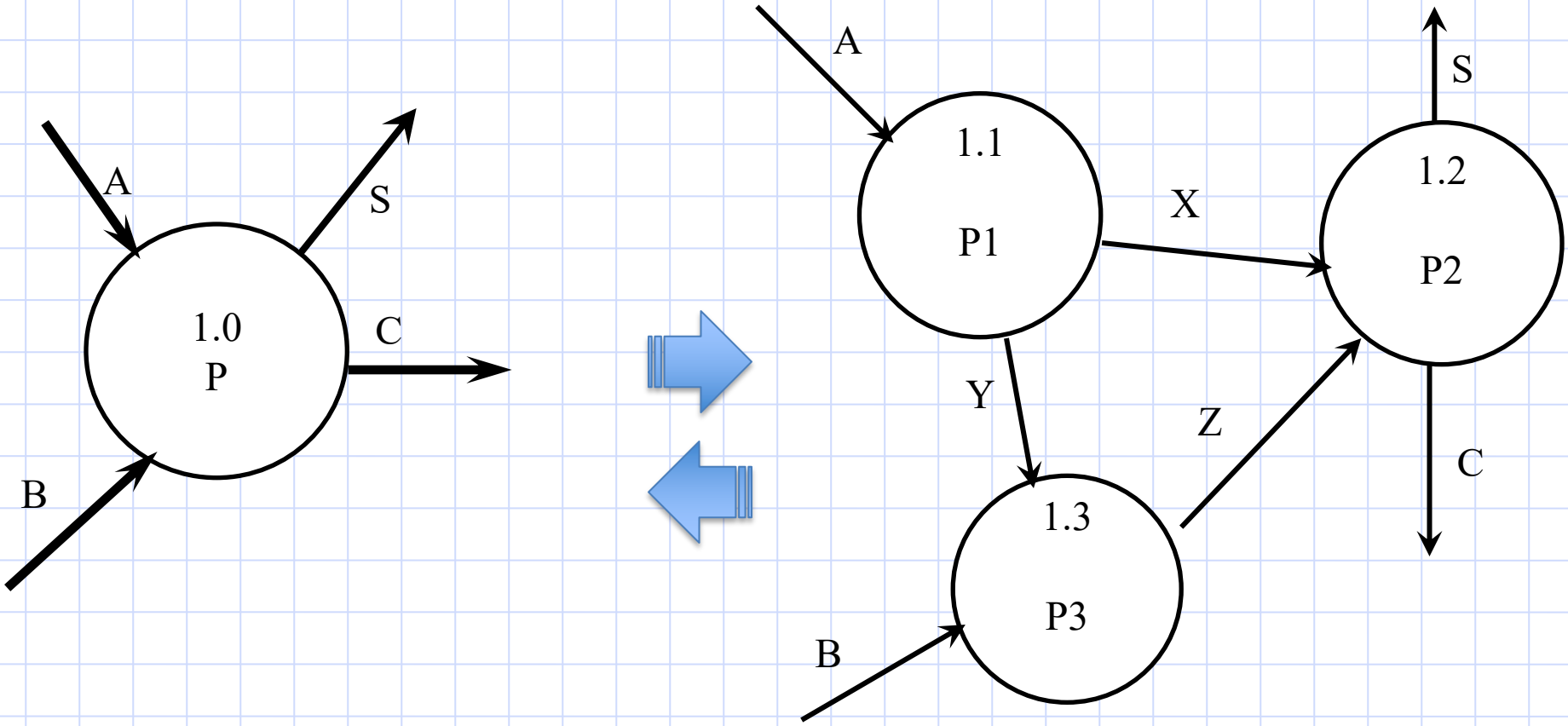
Data Flow Hierarchy



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.

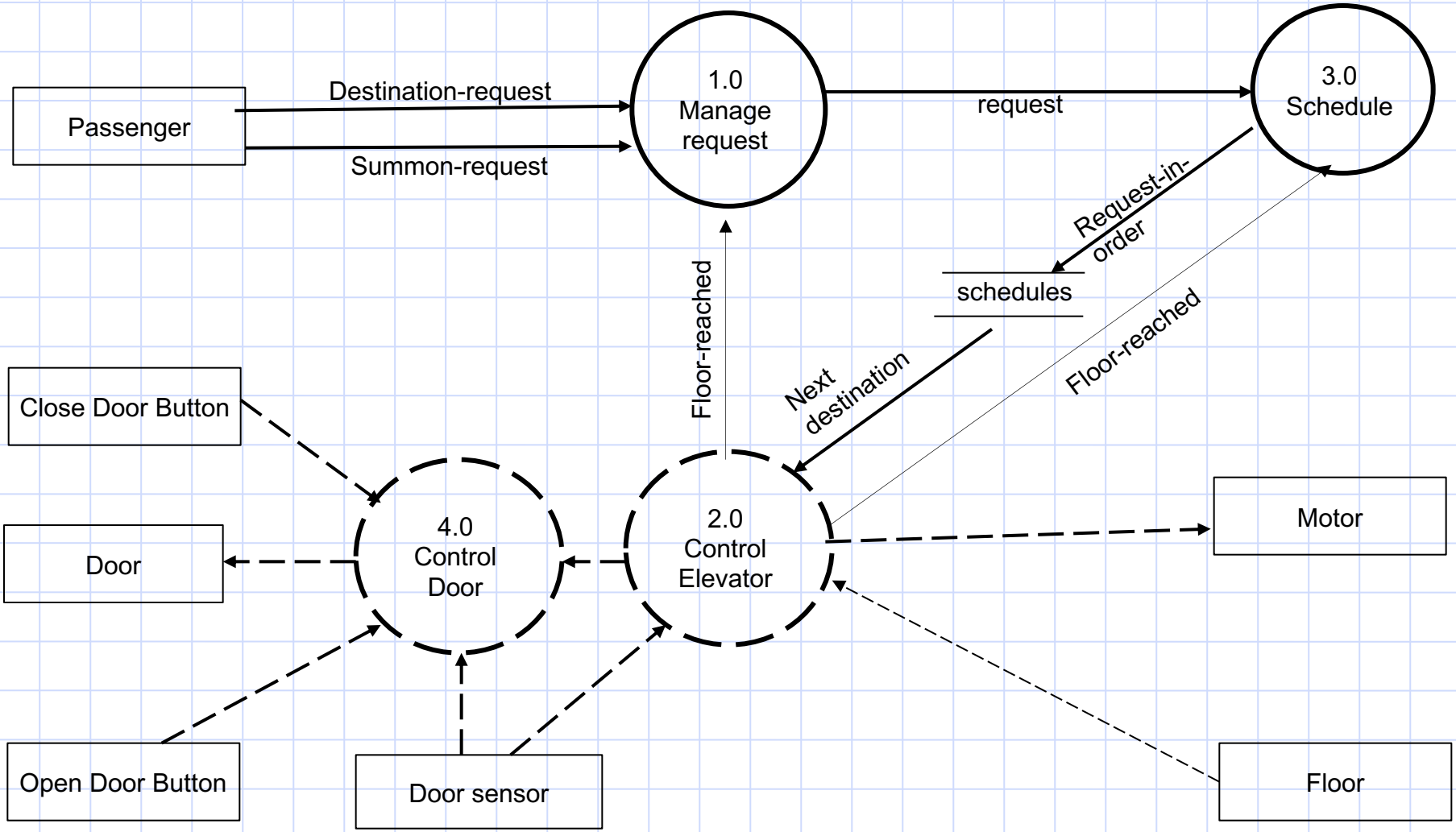
Logically Consistent

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

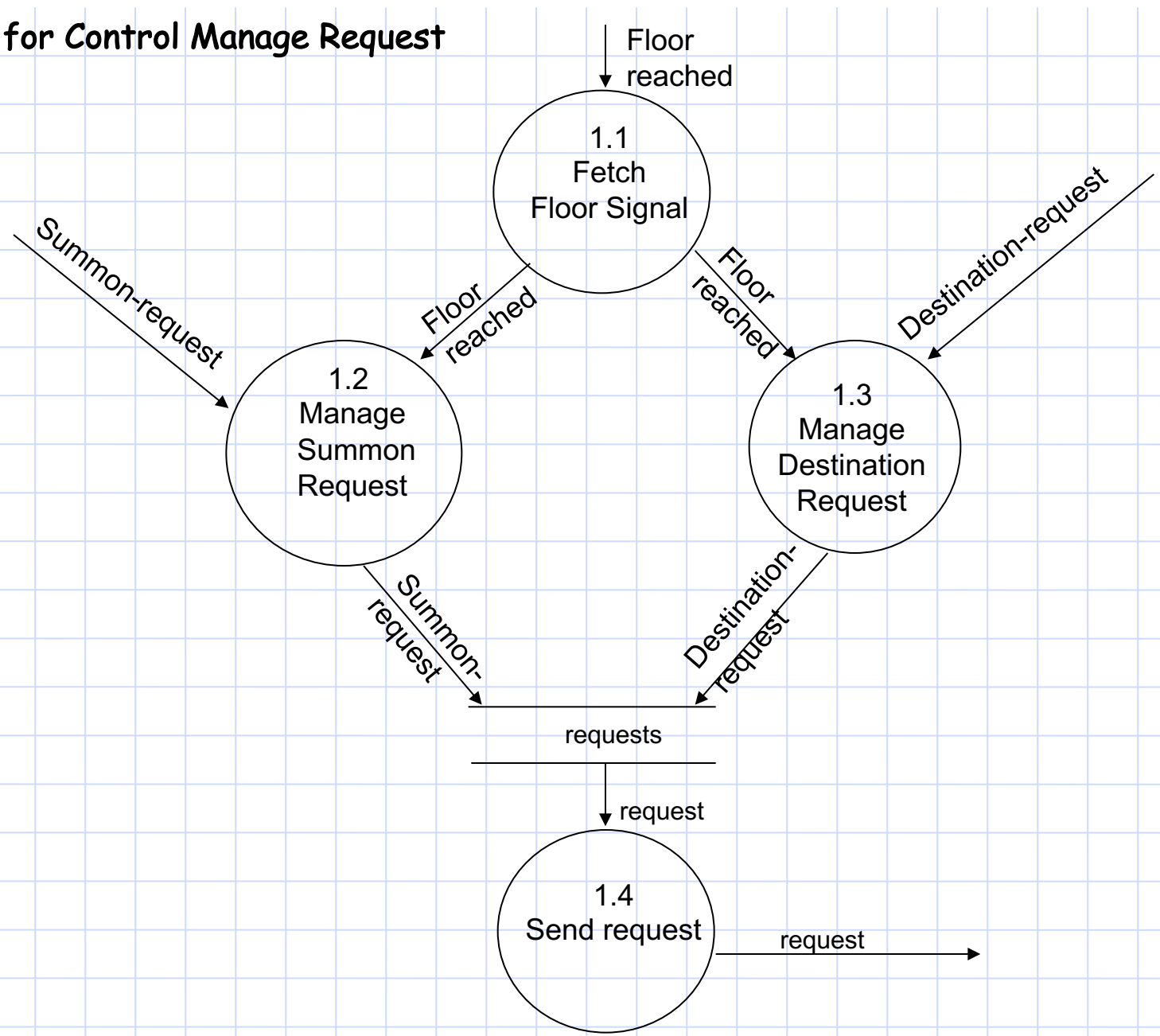
Data Stores in Leveling

- 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.

Simplified Level-1 for Elevator System



Level-2 for Control Manage Request

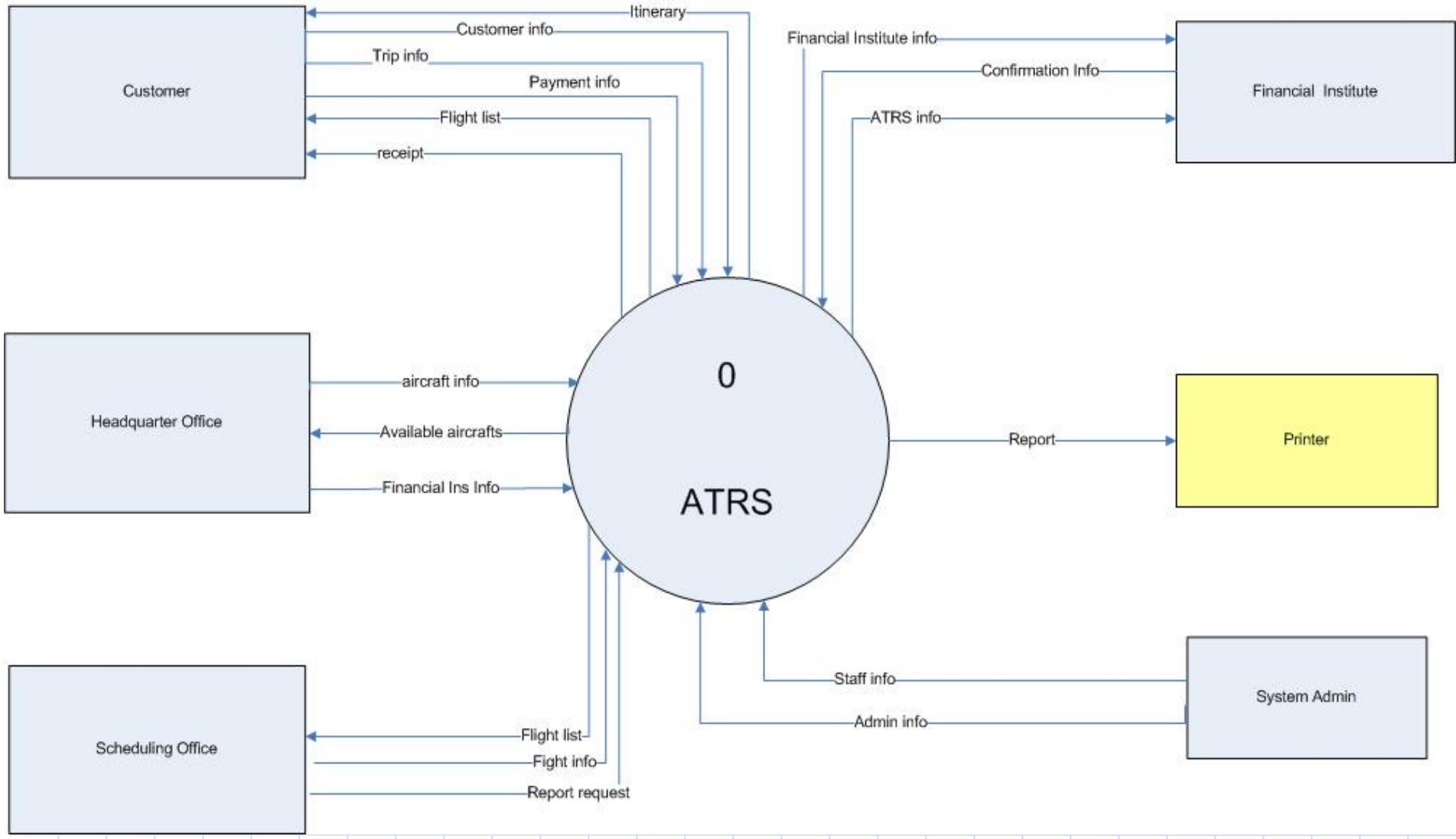


Another Example For
An Information/Transaction System

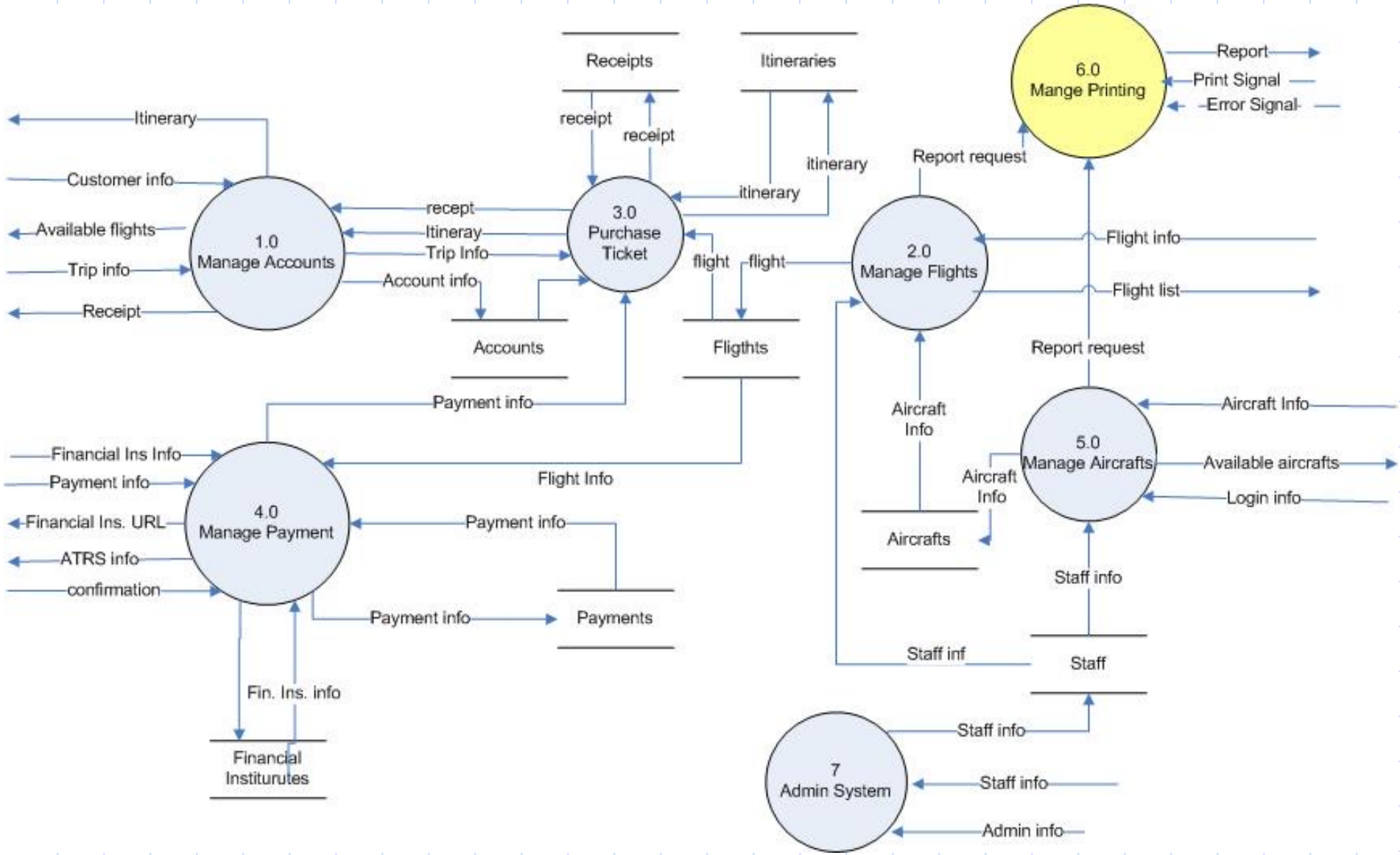
Airline Ticket Reservation System

- Given the following initial requirements provided by the customers, your are supposed conduct a functional and data requirements analysis, for an Airline Ticket Reservation System (ATRS):
- This application should:
  - *Allow **customer**s to sign up for an account, and manage their profile*
  - *Allow the customers to browse available flights from a source to a destination for an specified date*
  - *Allow customers to purchase their tickets, using available credits in their own account, PayPal, or common credit cards (MasterCard, Visa, etc.) and view their itineraries.*
  - *The **flight-scheduling** office should be able to browse the list of available aircrafts and mange the flights (add, remove, modify…). Each aircraft may have many flights*
  - *The **headquarter office** should be able to maintain the list of company's aircrafts.*
  - *Hidden requirement?*

Context Diagram:



Data Flow Diagram:



Levels Required

- small systems: 1-2 levels
- Medium size systems: 2 or 3 levels
- Large systems: 3 or more levels

Brief Introduction to Data Modeling

From DFD to ERD

Why Another Model

- Data Stores in a DFD show the places that data rests. But do not show the "natural structure of data"
  - Does not show the definition and relationships within the data
- Data Models are used to develop these descriptions

Brief Overview/Review of An Entity Relationship Diagram

What is ERD?

- Data Models use two main constructs:
  - Data Entities
  - Relationships
- These models are known as Entity Relationship Diagrams (ERD) are uses as a means of quickly obtaining, with minimum effort, a good sense of the structure of system's database.

What is an Entity

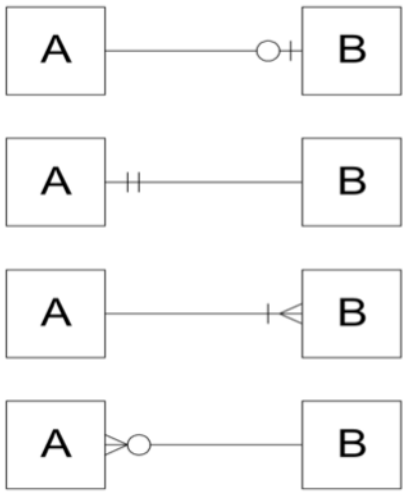
- Entities are abstract concepts within the data model. Each representing one or more instances.
- Each entity is represented by a box and a **Singular** name.
  - each row of the table representing an instance of that entity.

Data Modeling Process

- Each data store represents an entity
- Identify whether the terminator information is required to be accessed (if not done already in developing DFDs).
- Analyze each entity
  - Is it a composite-entity?
  - If yes, decompose it to more entities
- Identify the relationship among entities.
- Identify the type of relationship.
- If necessary, make corrections. Notice, that many to many relationship is not implementable in relational database systems
- Identify the multiplicity among the entities.
- Identify the optionality among the entities relationships

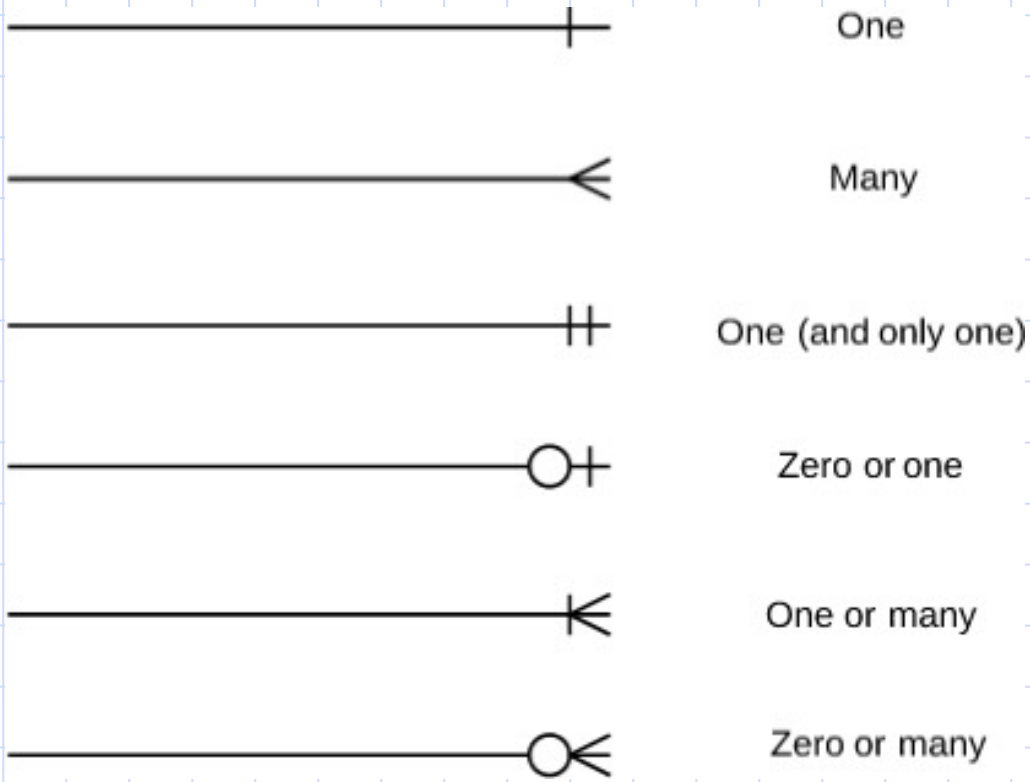
Optionality and Cardinality/Multiplicity

- Symbols at the ends of the relationship indicate the optionality and the cardinality of each relationship.
  - "Optionality" expresses whether the relationship is optional or mandatory.
  - "Cardinality" expresses the maximum INT of relationships.



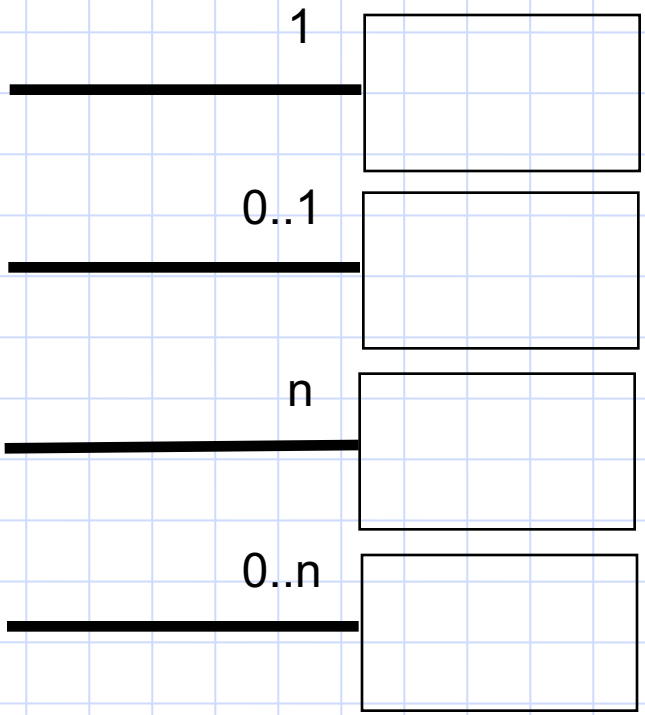
Optionality and Cardinality/Multiplicity

- Martin Notations



Cardinality – UML Notation

- Relationship Cardinality Notations Using UML Notation:



Mandatory 1 cardinality

Optional 0 or 1

Many cardinality

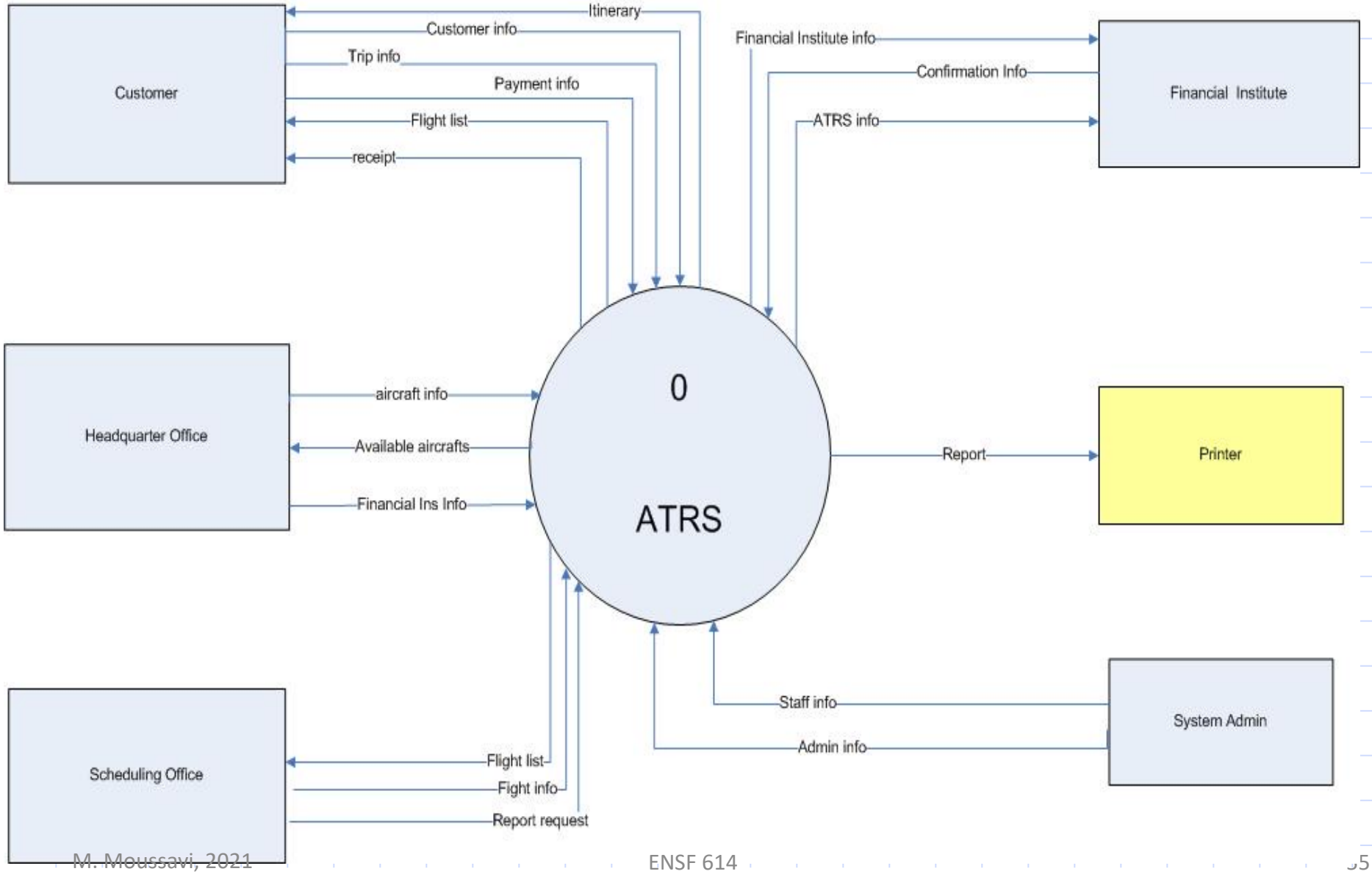
Optional Many

Example

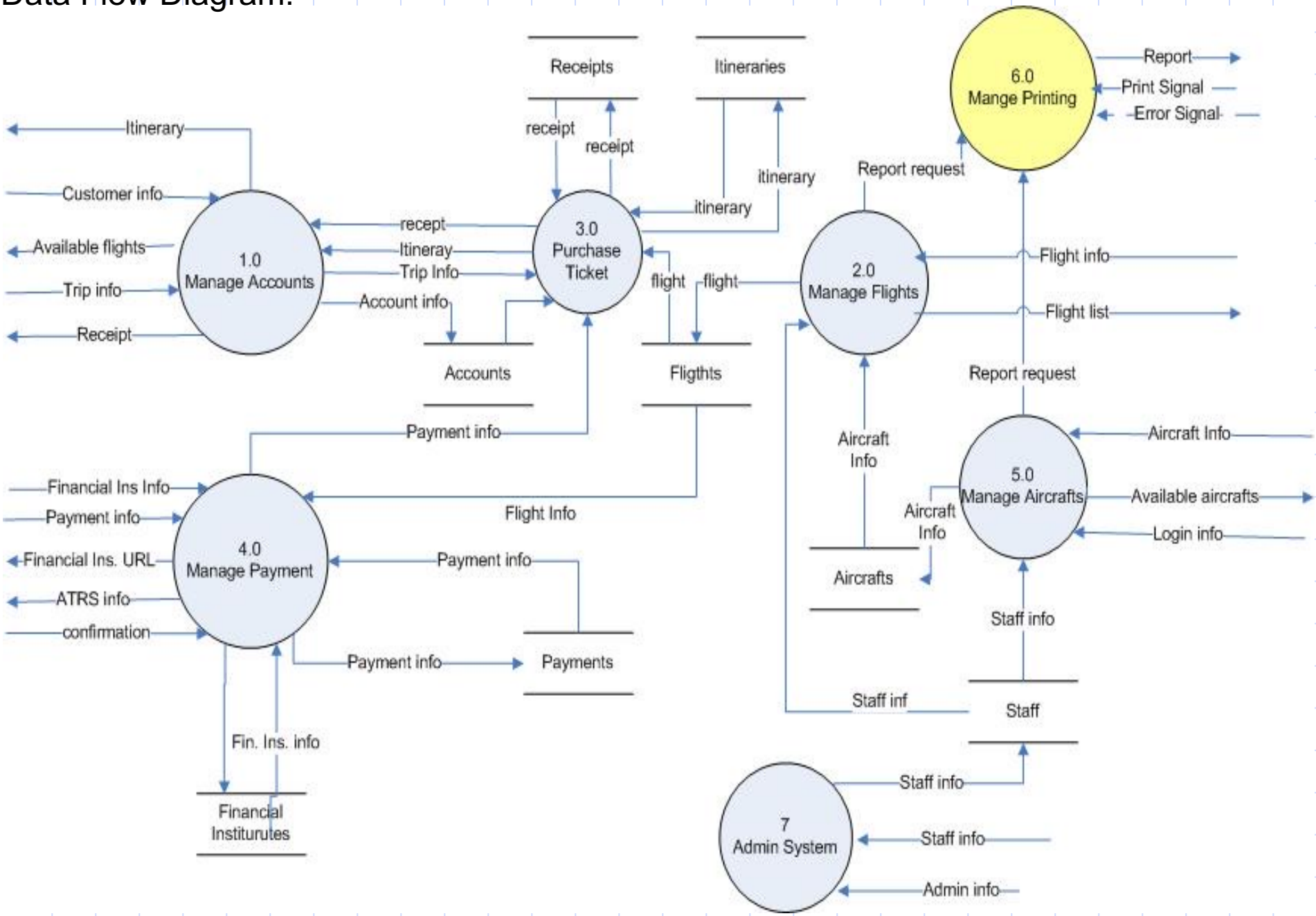
Airline Ticket Reservation System

- Given the following initial requirements provided by the customers, your are supposed conduct a functional and data requirements analysis, for an Airline Ticket Reservation System (ATRS):
- This application should:
  - *Allow customers to sign up for an account, and manage their profile*
  - *Allow the customers to browse available flights from a source to a destination for an specified date*
  - *Allow customers to purchase their tickets, using available credits in their own account, PayPal, or common credit cards (MasterCard, Visa, etc.) and view their itineraries.*
  - *The flight-scheduling office should be able to browse the list of available aircrafts and mange the flights (add, remove, modify…). Each aircraft may have many flights*
  - *The headquarter office should be able to maintain the list of company's aircrafts.*

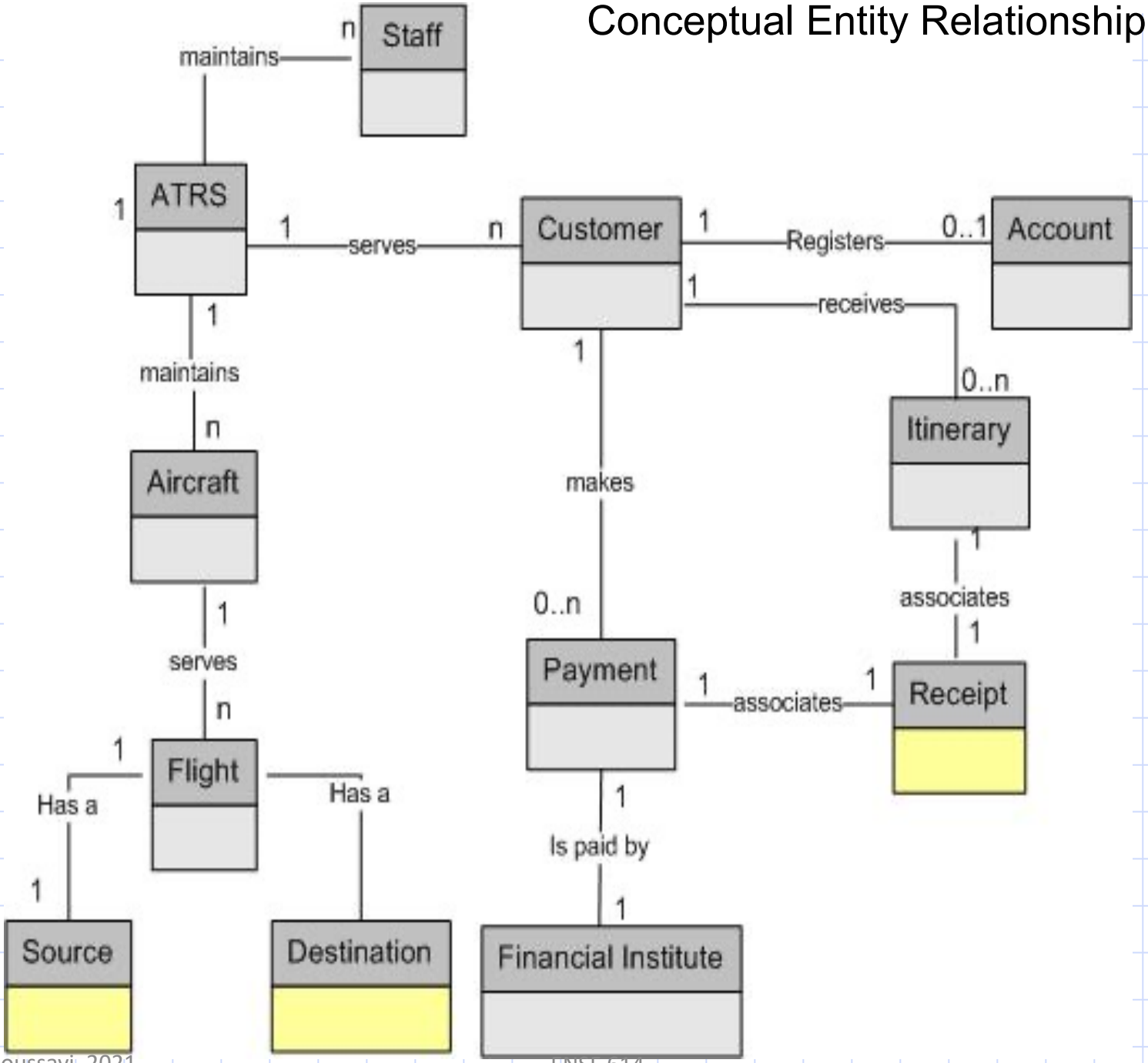
Context Diagram:



Data Flow Diagram:



Conceptual Entity Relationship Diagram



Data Dictionary

Data Dictionary

- A listing of all of the data elements, organized alphabetically.
- Defines each data element in a precise, rigorous manner.
- Describes the:
  - Meaning of all data flows and data stores.
  - Composition of data flows.
  - Composition of stored data.
- Specifies the values and units of data
- Helps define the relationship between data stores (in conjunction with the ERD).
- Understandable to both client and systems analyst.
- The level of detail at analysis stage will be much less than at design stage.
- At design stage, the level of detail must be sufficient to create code.

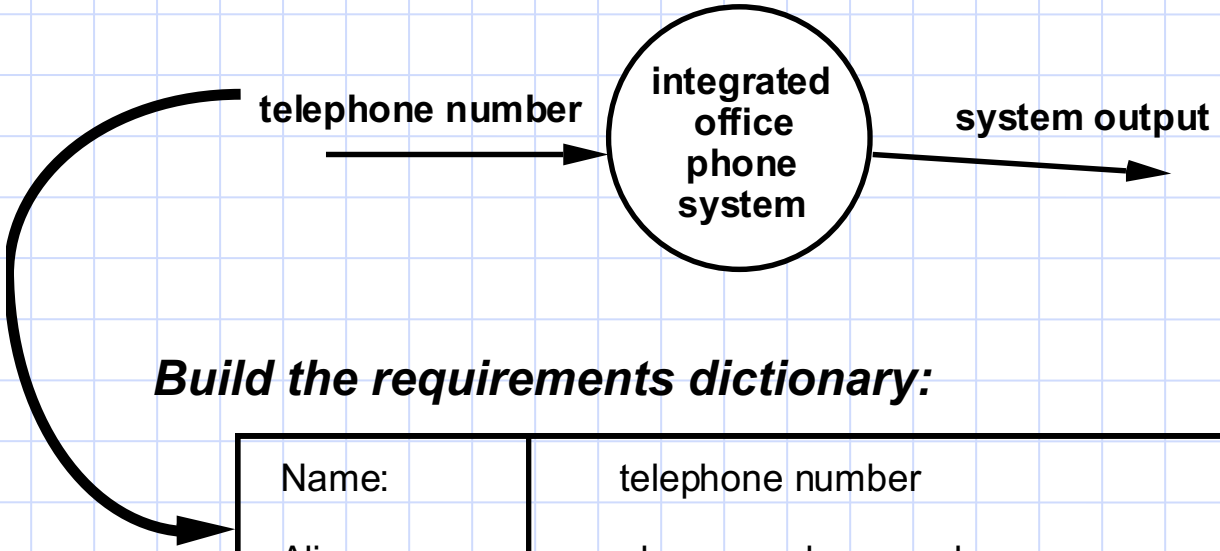
Building a Data Dictionary

Name:	the primary name of the composite data item
Aliases:	other names for the data item
Where used:	data transforms (processes) that use the composite data item
How used:	the role of the data item (input, output, temporary storage, etc.
Description:	a notation for representing content (presented on next slide)
Format:	specific information about data types, pre-set values (if known)

Data Dictionary Notation

Notation	*Meaning*
=	is composed of
+	and
[|]	either-or
$\{\ \}^n$	n repetitions of
(...)	optional data
* ... text ...*	delimits a comment

Data Dictionary Example



Build the requirements dictionary:

Name:	telephone number
Aliases:	phone number, number
Where/How used:	read-phone-number (input) display-phone-number (output) analyze-long-distance-calls (input)
Description:	telephone no. = [local extension | outside no. | 0] outside no. = 9 + [service code | domestic no.] service code = [211 | 411 | 611 | 911] domestic no. = ((0) + area code) + local number area code = *three numeral designator*
Format:	alphanumeric data

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

Data Dictionary for Elevator System

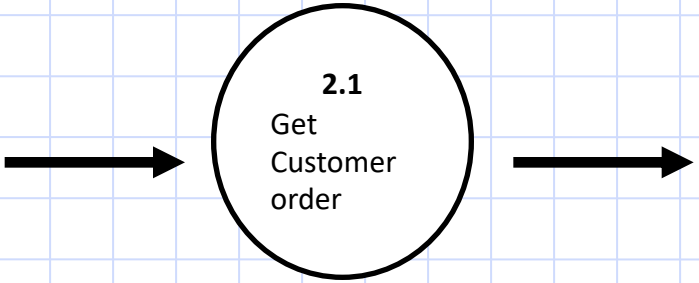
- Destination request = Floor number
- Direction = [up | down]
- Floor number = 1{legal digits}4
- Floor reached = signal
- Motor control = [on | off] + direction
- Requests = summon request + destination requests.
- Summon request = Floor number

What is a Process Spec (PSPEC)

PSPEC

- During the system analysis the PSPEC can be used to describe the internals of each process.
  - inner workings of a process represented in a data flow diagram.
  - the input to a process and the algorithm
  - restrictions and limitations imposed on the process (function) performance
    - For complex processes you may add a state machine or even a flowchart to help.
- Process specs are normally provided for the low level processes (2^{nd} or 3^{rd} level)
- Process specifications are not needed for:
  - Simple input and/or output processes.
  - simple data validation.
  - Processes for which prewritten code already exists.

Process Specs Format During Analysis



- Process < Number> - Process Label
  - Preconditions:
    - What is required to invoke the process
  - Post condition:
    - What it does…

Process 2.1: Get Customer Order

Description:

- This process provides a client side process to deliver the customer order to the web and database server.

Preconditions:

- Needs valid customer information and valid order items

Post conditions:

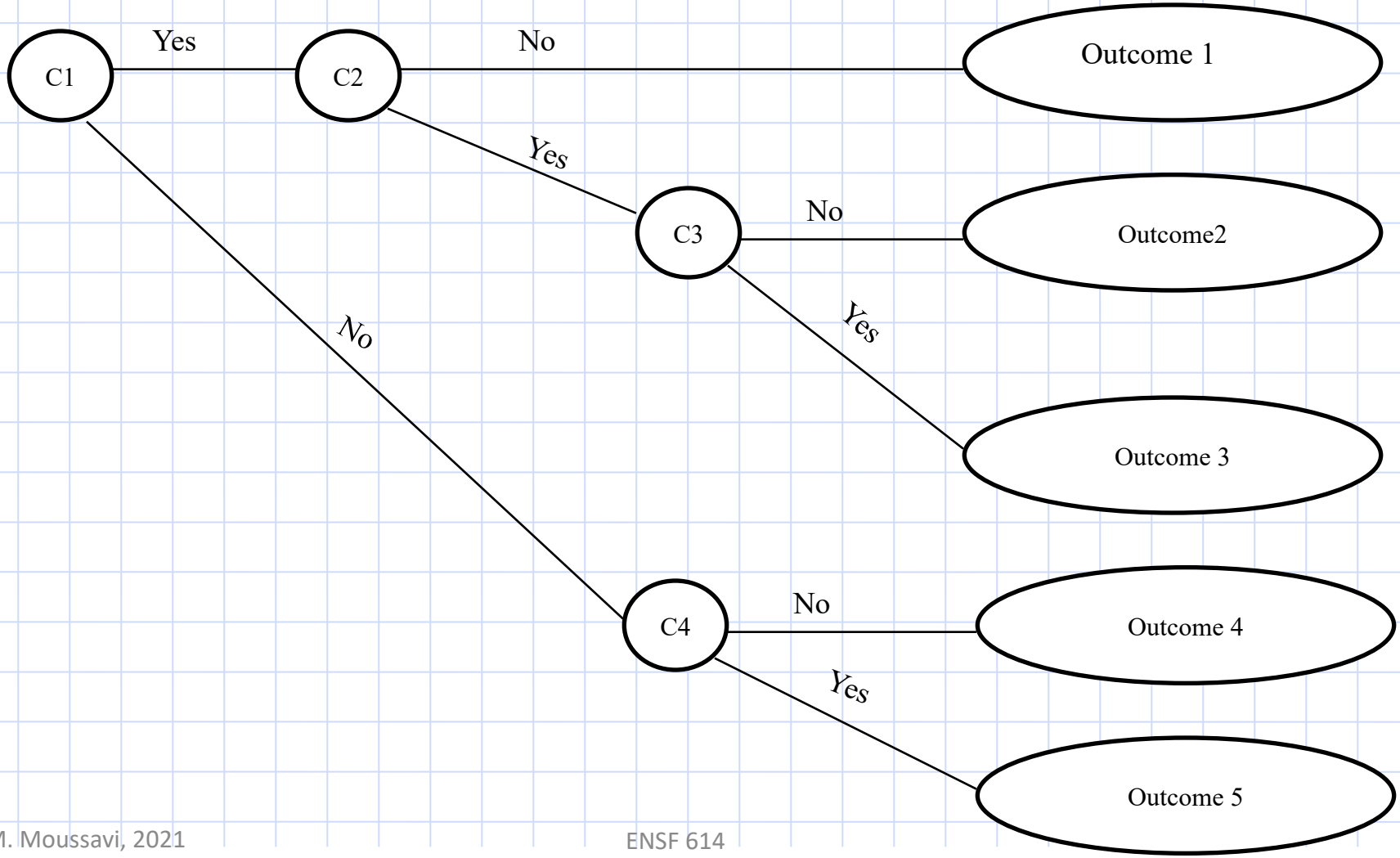
- store the order

Supporting Artifact

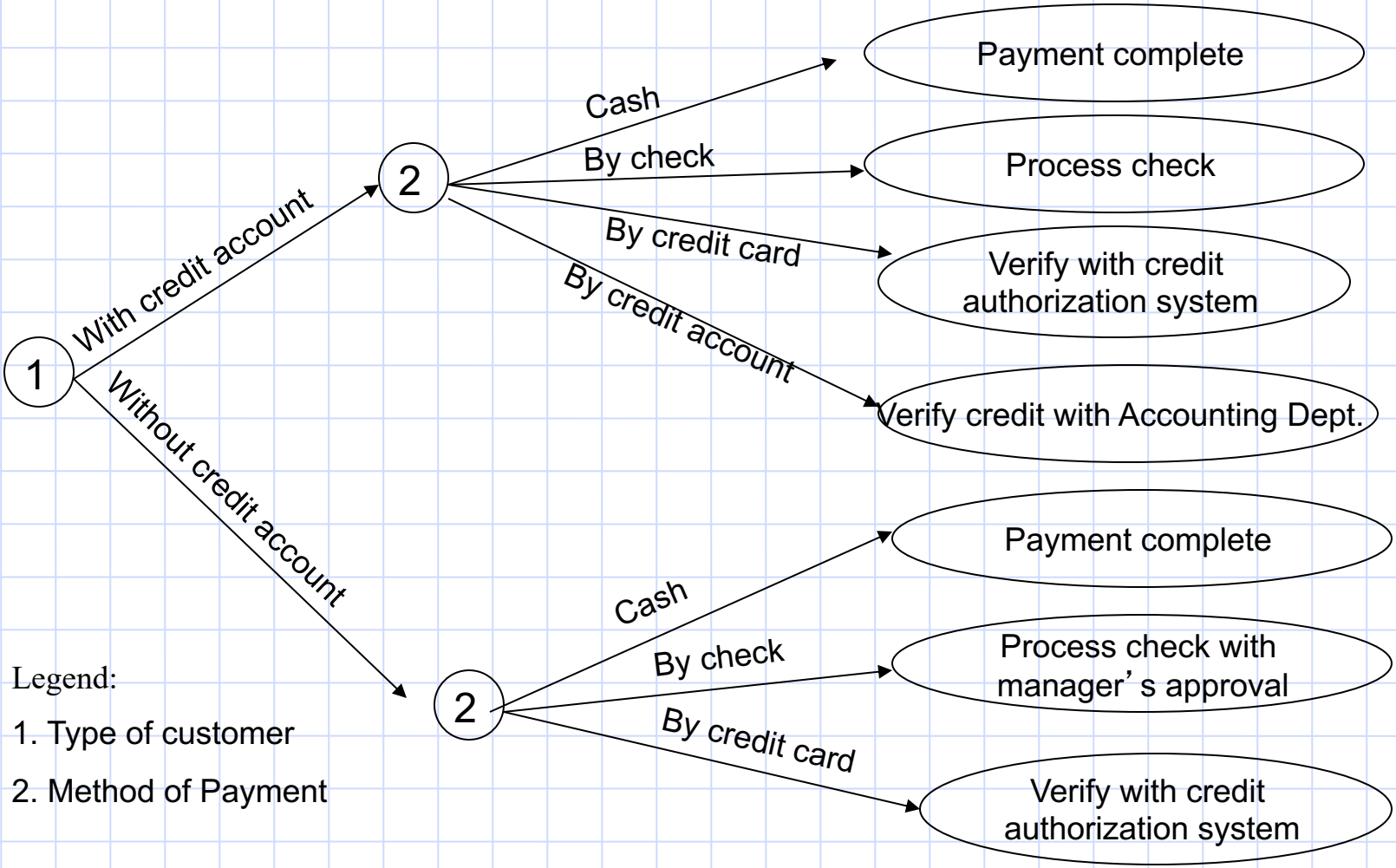
- Process specs can be supported by :
  - State Transition Diagrams (STDs)
  - Activity Diagram
  - Structure Chart
  - Decision Tables.
  - Decision trees.
  - Structured English
  - Data Dictionary
- Use a combination of tools based on:
  - Client / analyst preference.
  - Nature of the process.

Decision Trees

- A graphical representation of a decision situation using:
  - Decision points represented by circles, actions by ovals, and connections between decision points and actions by lines with values for conditions.



An example for a decision tree for the process of make Payments



Decision Tables

- Decision tables might be used for complex processes
- A matrix representing the logic of a complex decision.
- Inputs as columns, actions (outputs) as rows.
- Example: a decision table that shows the conditions to serve cocktail in travel airline.

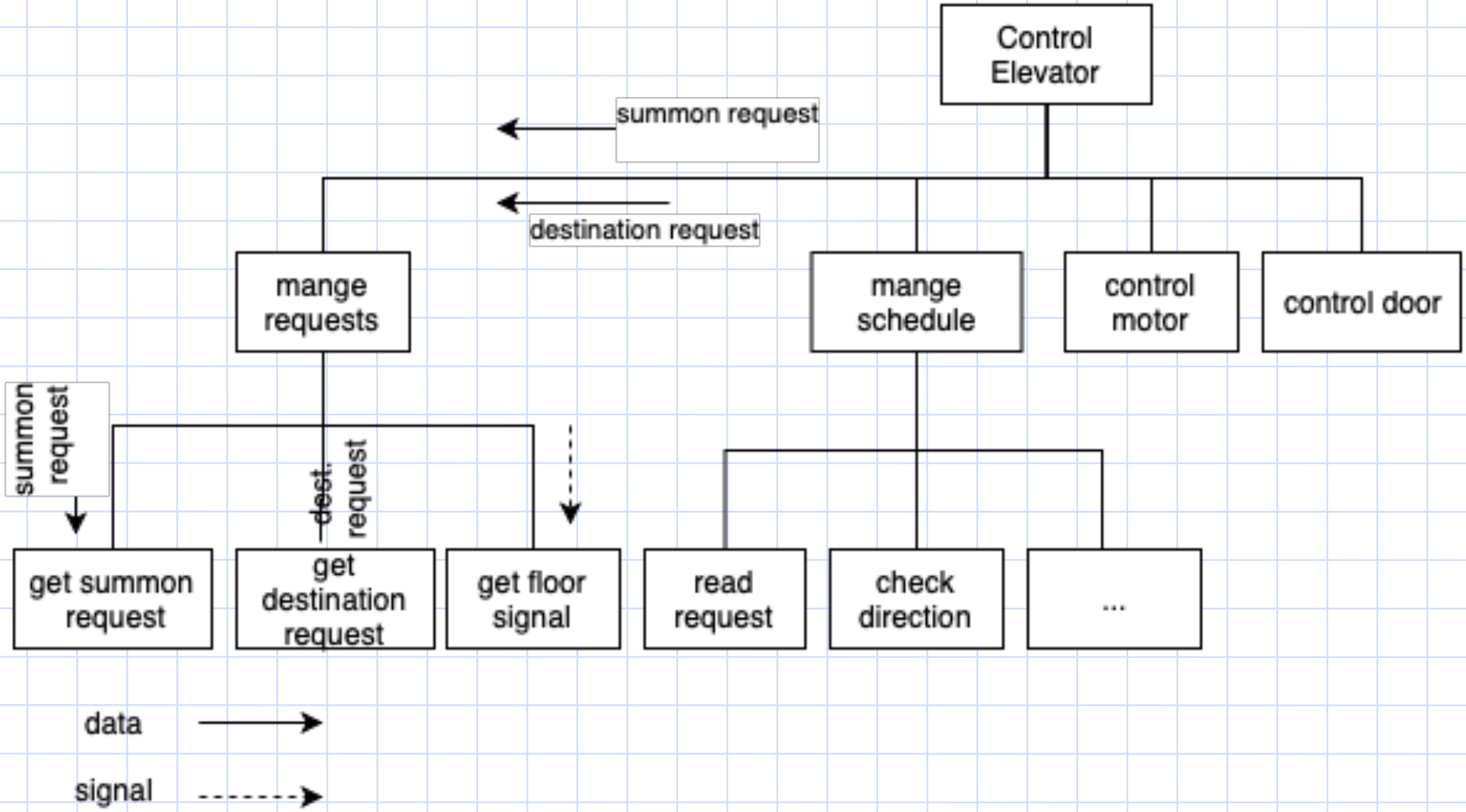
conditions	Domestic?	Y	Y	Y	Y	N	N	N	N
	≥half full?	Y	Y	N	N	Y	Y	N	N
	≥$350/seat	Y	N	Y	N	Y	N	Y	N
outcomes	Serve cocktails?	✓	✓	✓	?	✓	?	?	?
	Free cocktails?					✓			

Structured English

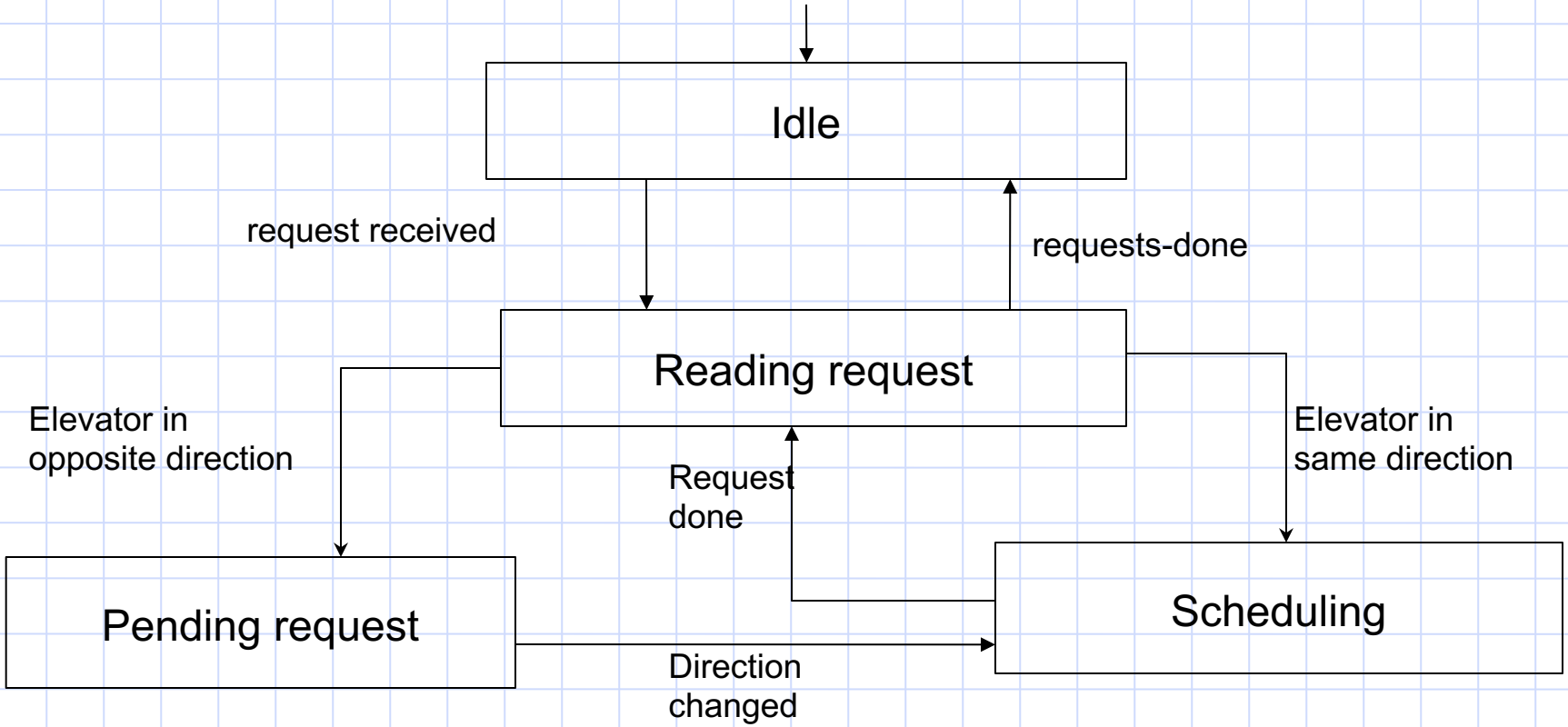
- Modified form of the English language used to specify the logic of information systems processes.
- A subset of English which includes:
  - Verbs such as READ, WRITE, ADD, SUBTRACT, etc.
  - Noun phrases to represent data and data structures (defined in the data dictionary).
- Does not include adverbs and adjectives.
- Usually 40 to 50 verbs.
- No standard version

Structure Chart (SC)

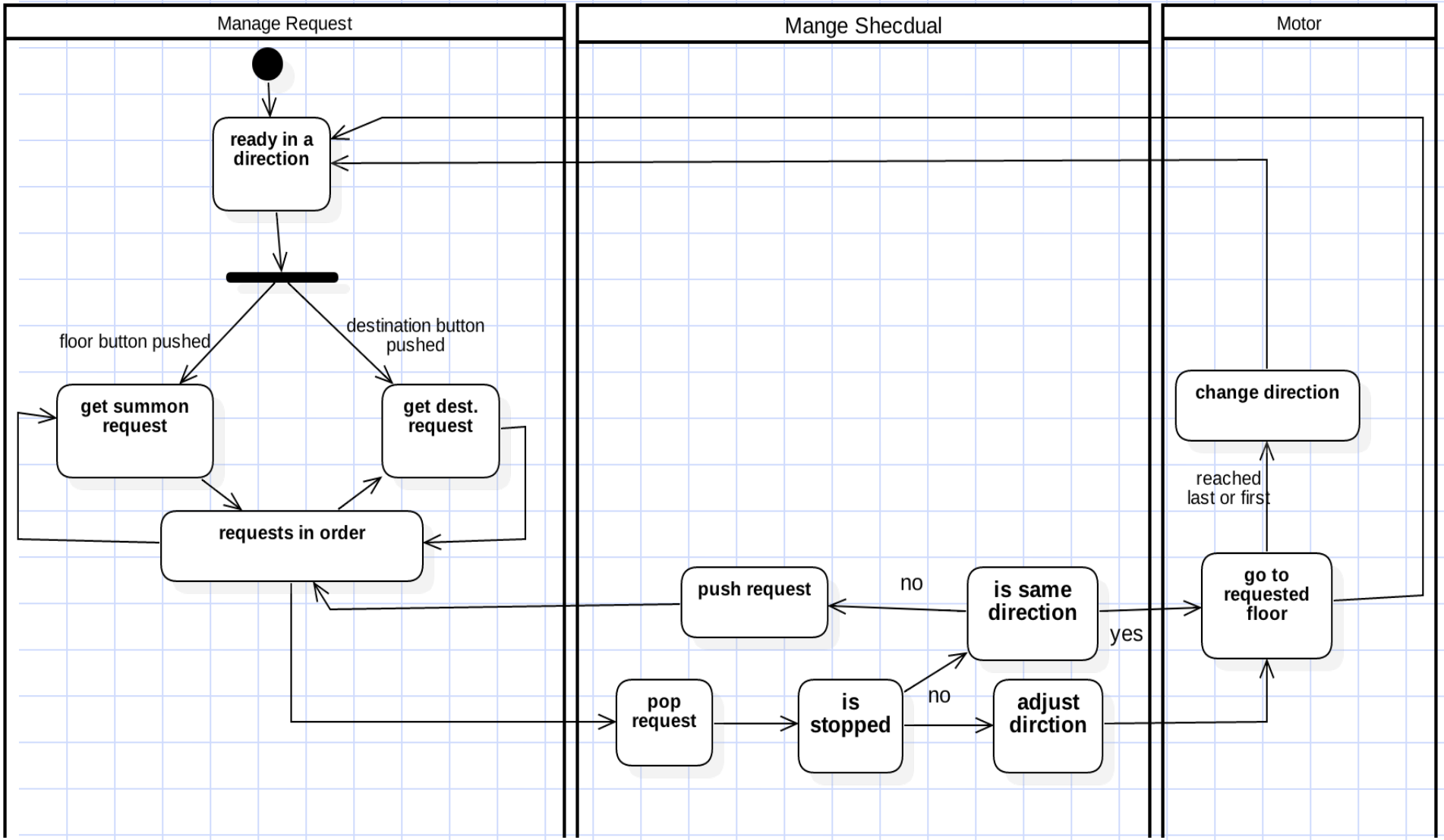
- In software engineering the SC is a diagram that shows the breakdown of a software at its lowest level (collection of tasks, modules, or functions). Here is an example of a structure chart:



STD for Schedule Process in an Elevator System



Activity Diagram for Elevator System



Structured English for Schedule Process

- BEGIN
- with summon/destination request received
- DO WHILE request exists
- IF elevator in-direction
- Move elevator to destination
- Clear request
- ELSE
- Pending the request
- ENDIF
- END DO
- END