

Basic Software Engineering

Prof. Vibhuti Patel, Assistant Professor
Parul Institute of Computer Application





CHAPTER - 3

Software Design



DFD (Data Flow Diagram)

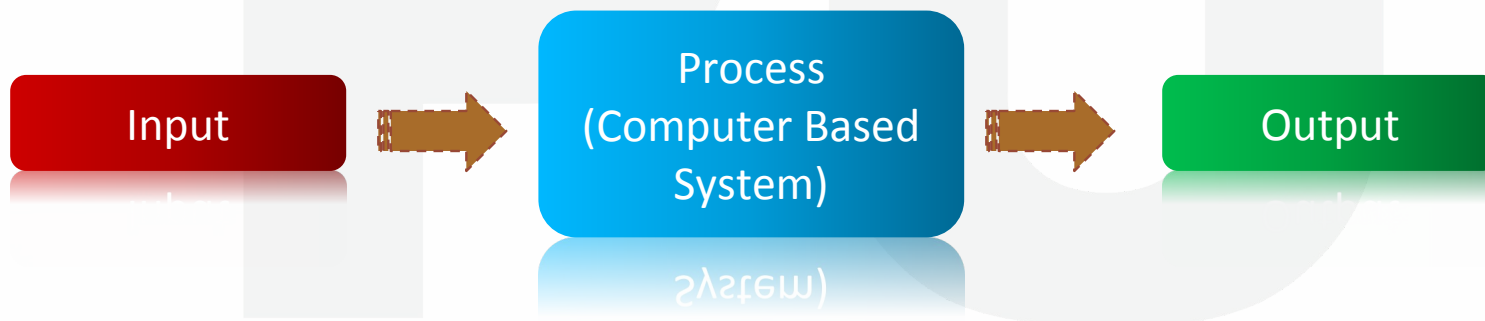
- A Data Flow Diagram (DFD) is a way of representing a flow of a data of a process or a system (usually an information system).
- The DFD also provides information about the outputs and inputs of each entity and the process itself.
- A data flow diagram has no control flow, there are no decision rules and no loops.



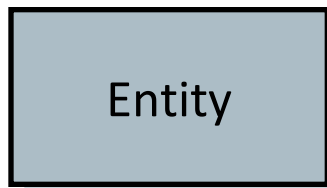
DFD (Data Flow Diagram)

- DFD takes an **input-process-output** view of a system. Data flow into the software & transformed by processing elements & resultant data flow out of the software.

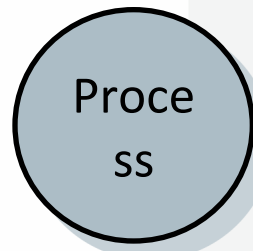
Example: Every computer-based system is an information transform.



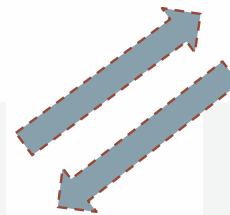
DFD Notations



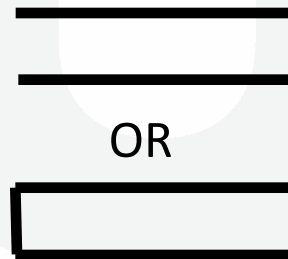
External Entity



Process



Data Flow



Data Store



DFD Notations

Entity

External Entity: Entities are sources and destinations of information data.

Entities are represented by rectangles with their respective names.

Example: a person, a device, a sensor



DFD Notations

Proce
ss

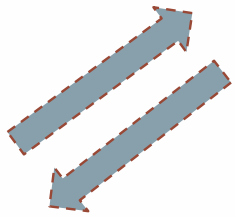
Process: A data transformer (changes input to output).

Activities and action taken on the data are represented by Circle or Round-edged rectangles.

Example: computer system, calculate area



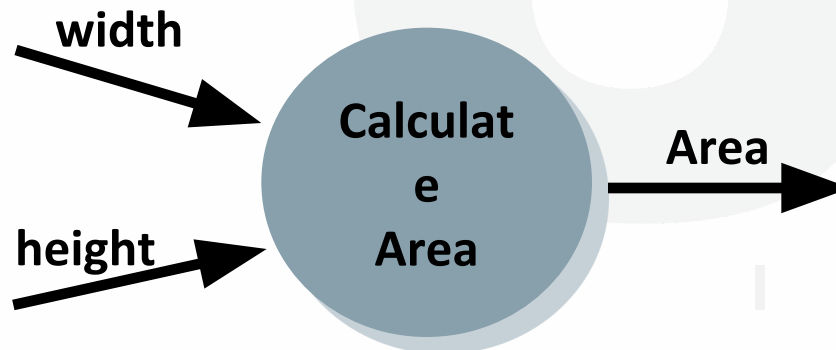
DFD Notations



Data Flow: Data flows through a system, beginning as input & transformed into output.

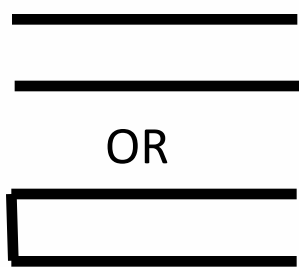
Movement of data is shown by pointed arrows.

Example: Area of Rectangle





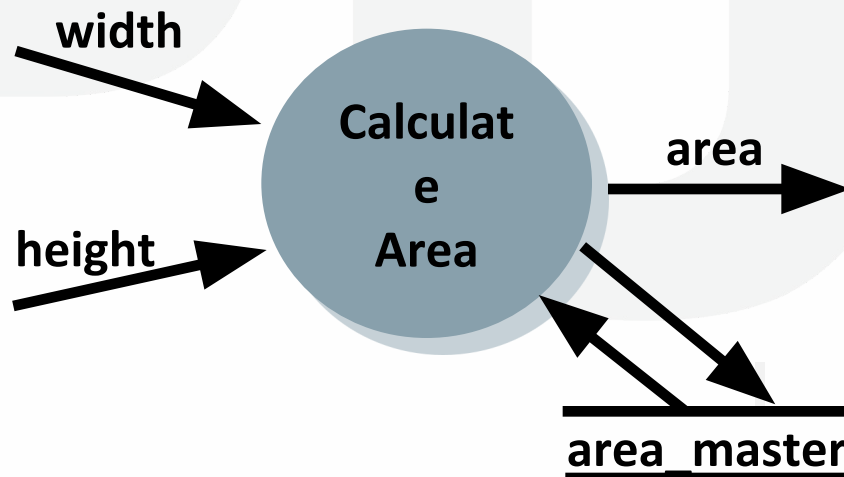
DFD Notations



Data Store: There are two variants of data storage - it can either be represented as a rectangle with absence of both smaller sides or as an open-sided rectangle with only one side missing.

Data is often stored for later use.

Example: Area of Rectangle



Data Flow Diagram 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 (also called level 0)
- Always show external entities at level 0
- Always label data flow arrows
- Do not represent procedural logic



Levels of DFD

Context Level DFD (0 Level / Zero Level):

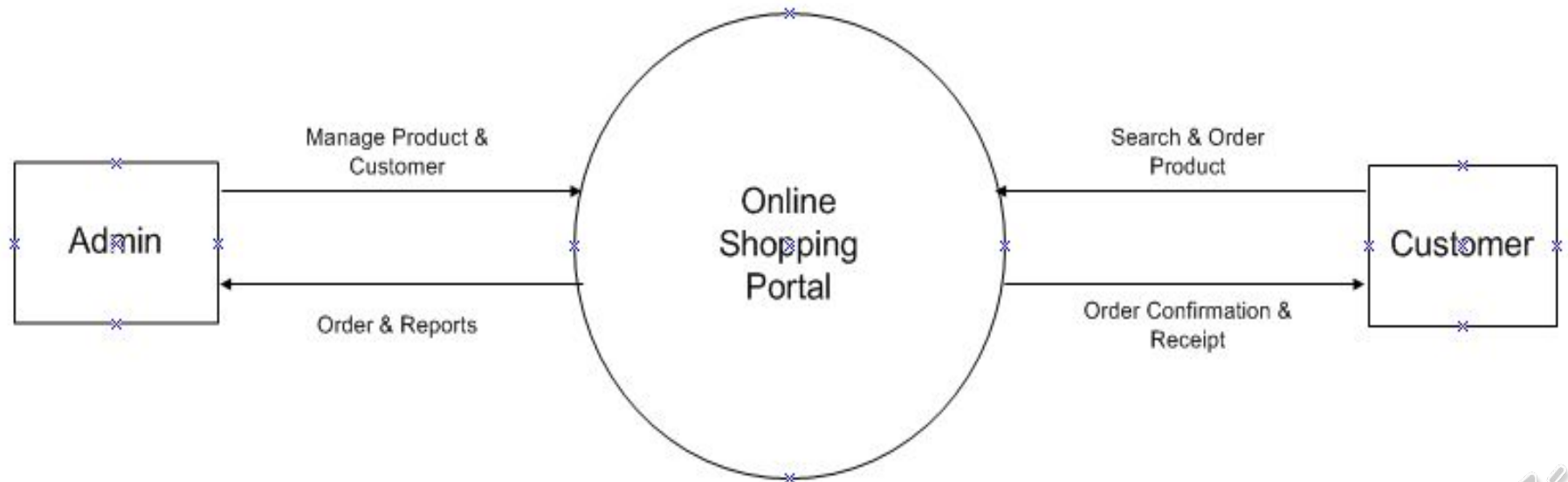
- A **context** diagram is a **top level** (also known as "**Level 0**") data flow diagram.
- It only contains one process node ("Process 0") that generalizes the function of the entire system in relationship to external entities.
- A "context level" DFD can be used to show the interaction between a system and outside entities; it can also show the internal data flows within a system. This version is also called a context diagram.



Levels of DFD

Context Level DFD (0 Level / Zero Level):

Example: Online shopping Portal



Levels of DFD

Level 1 DFD:

- A level 1 DFD notates each of the main sub-processes that together form the complete system.
- We can think of a level 1 DFD as an “exploded view” of the context diagram.



Levels of DFD

Constructing Level 1 DFD:

- If no context diagram exists, first create one before attempting to construct the level 1 DFD.
- Identify processes & external entities
- Draw the data-flows between the external entities and processes.
- Identify data stores by establishing where documents / data needs to be held within the system. Add the data stores to the diagram, labelling them with their local name or description.



Levels of DFD

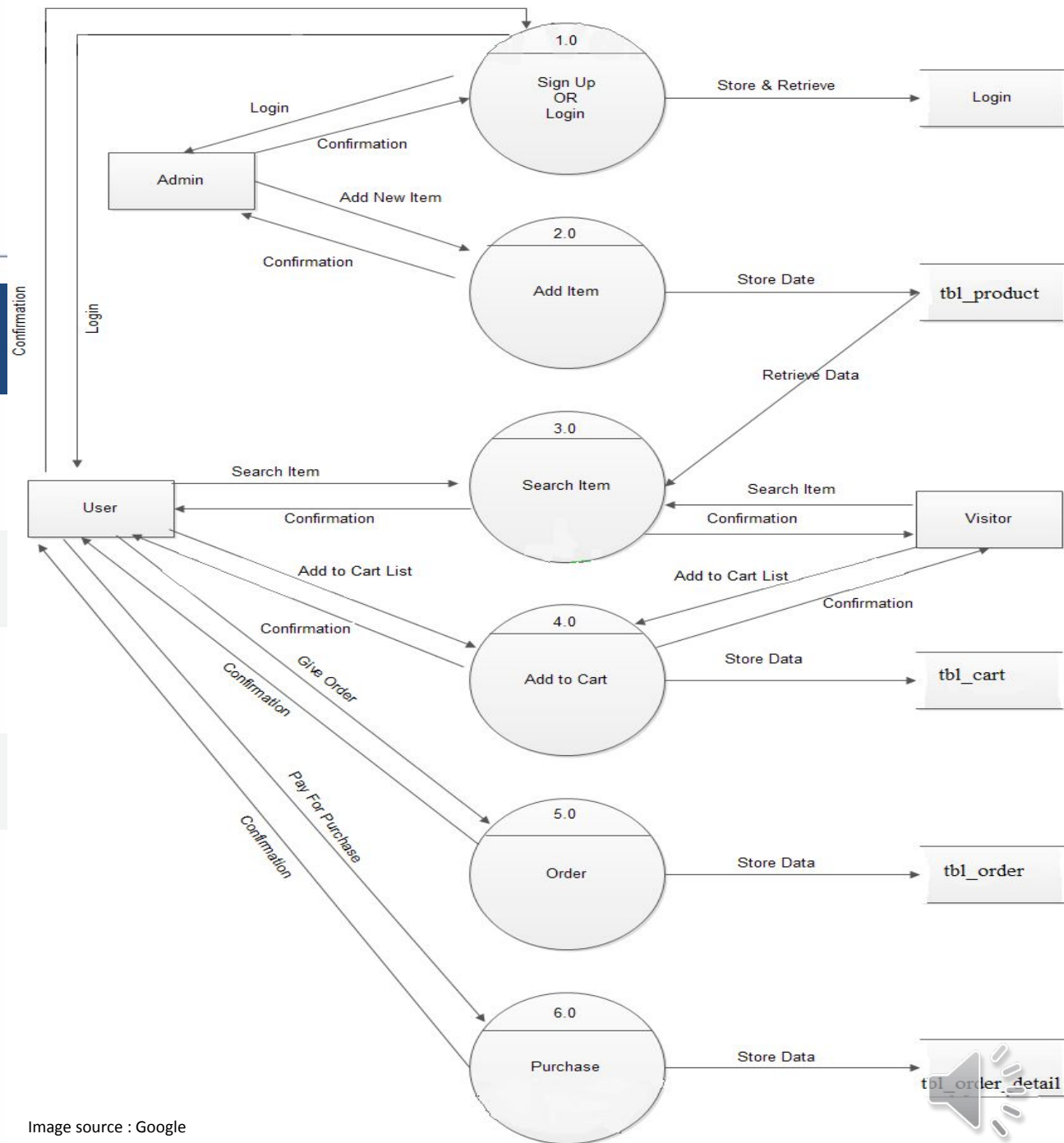
Constructing Level 1 DFD:

- Add data-flows flowing between processes and data stores within the system. Each data store must have at least one input data-flow and one output data-flow
- Check diagram. Each process should have an input and an output. Each data store should have an input.



Levels of DFD

Level 1 DFD Example:



Levels of DFD

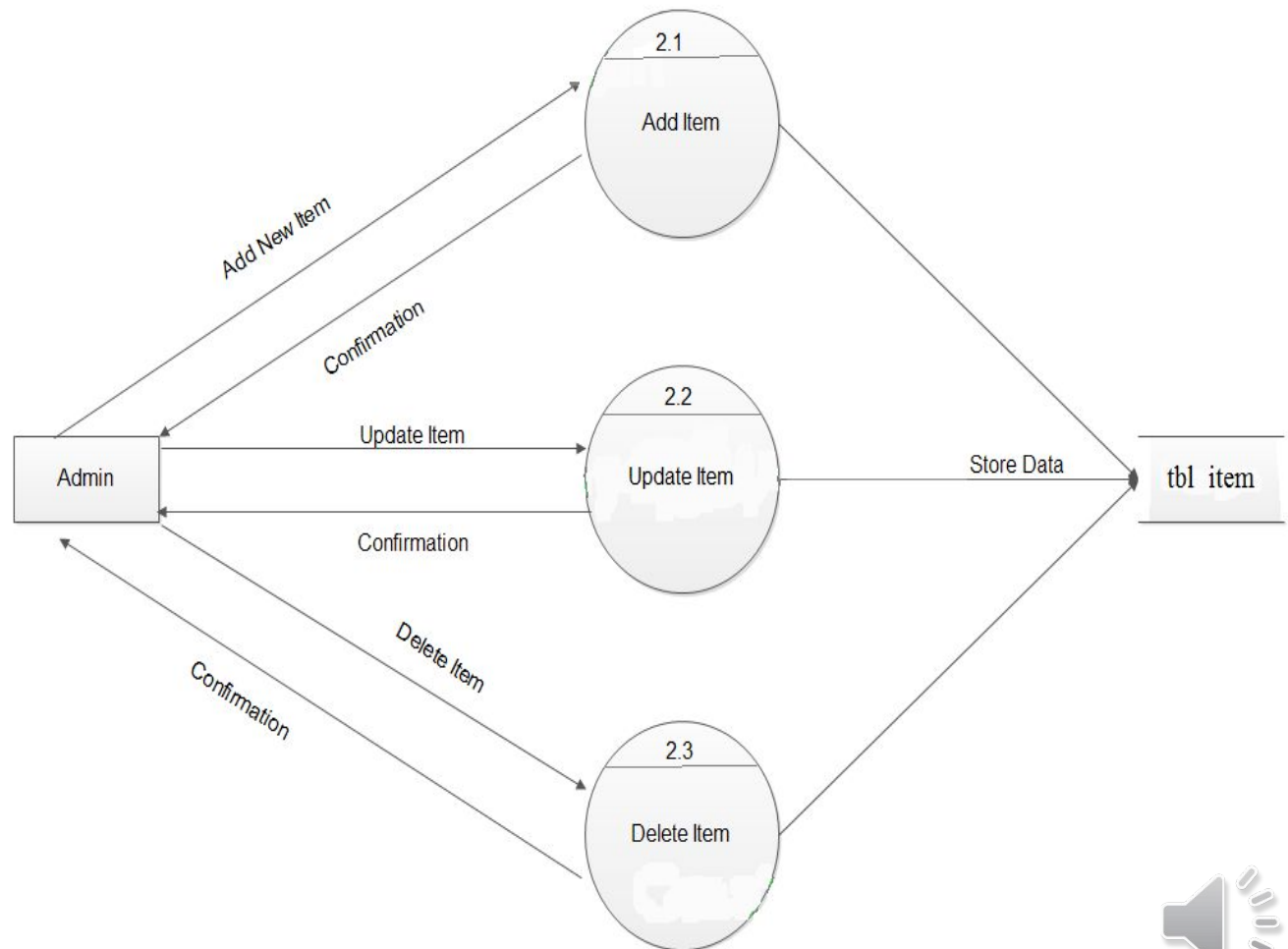
Level 2 and so on

- A level 2 DFD explodes more summarized processes and shows another level of complexity within them.
- A level 3 or 4 DFD shows even more components opened up to show their inner details.



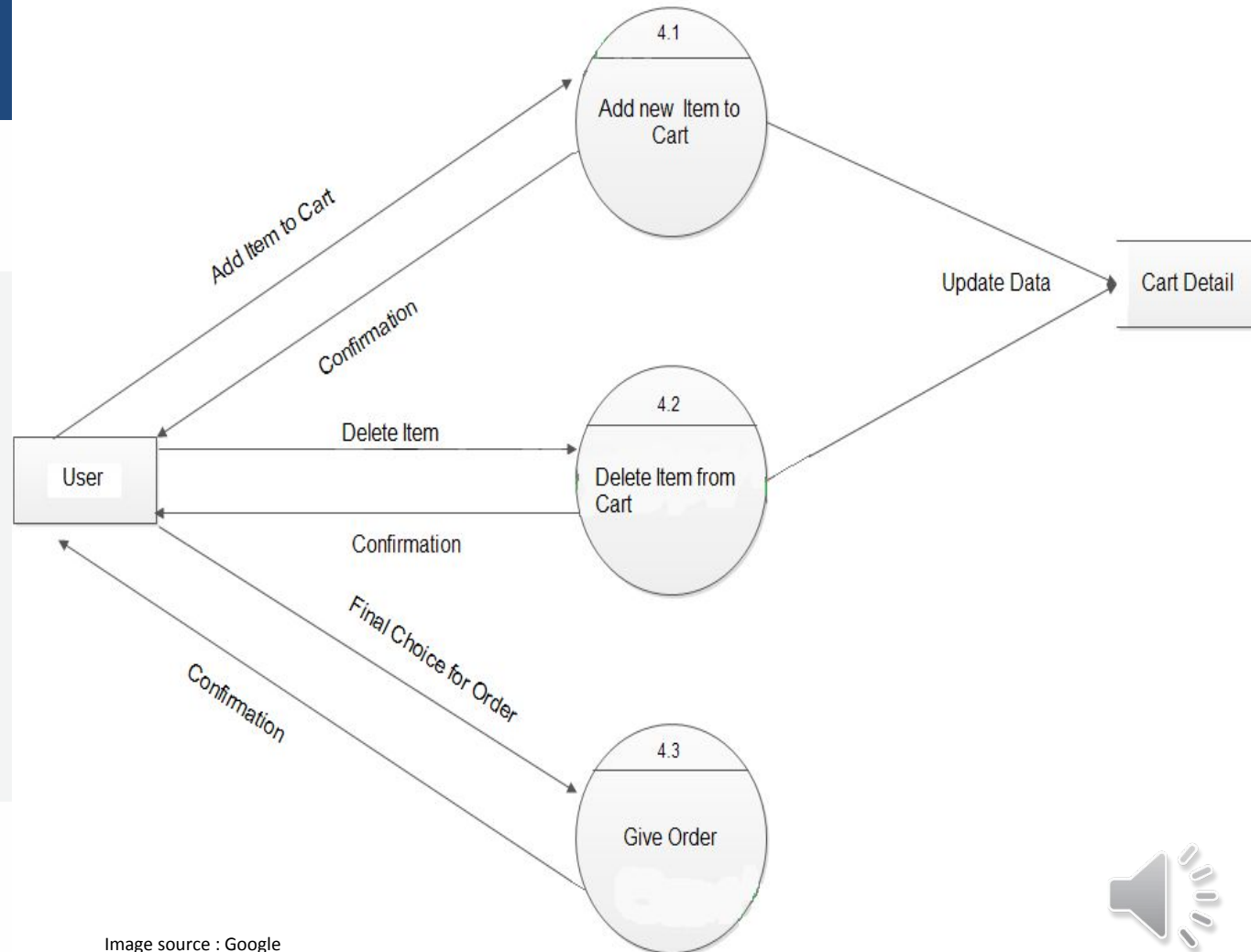
Levels of DFD

Level 2 DFD Example:



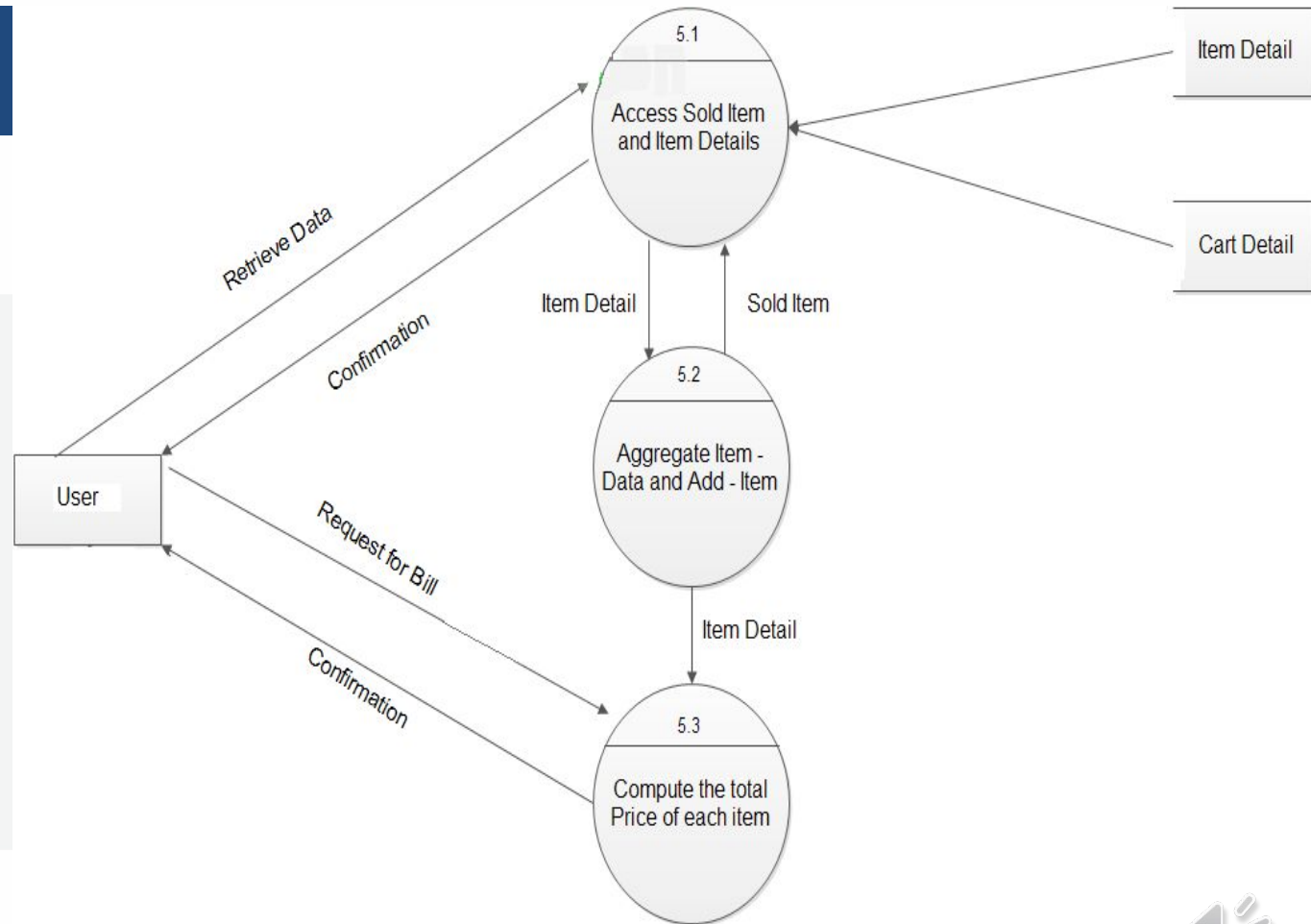
Levels of DFD

Level 2 DFD Example:



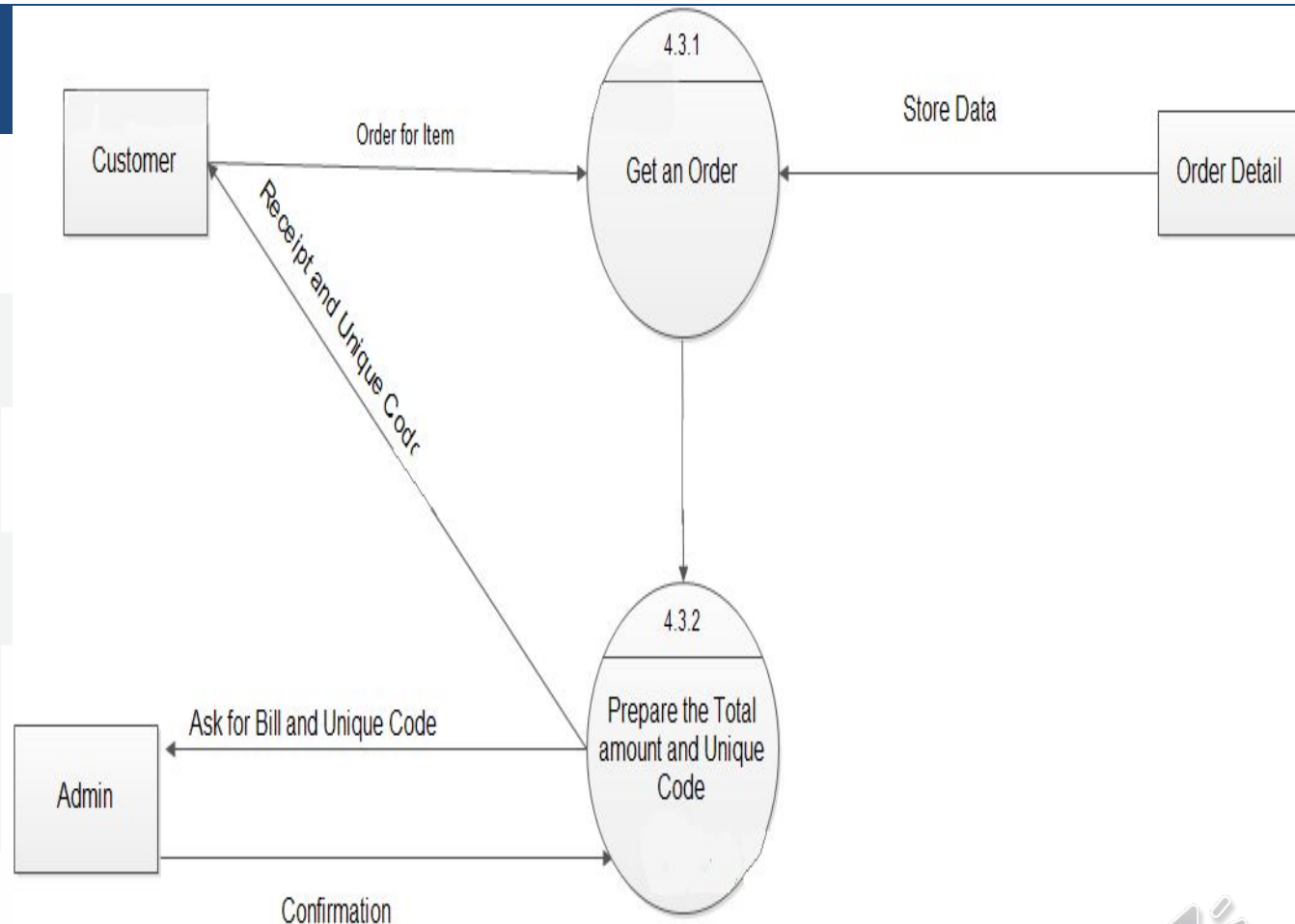
Levels of DFD

Level 2 DFD Example:



Levels of DFD

Level 3 DFD Example:



What is Data Dictionary?

- It is an integral part of a database.
- It holds information about the database and the data that it stores.
- A data dictionary is a “**virtual database**” containing metadata (data about data).



What is Meta Data?

Metadata is defined as data providing information about one or more aspects of the data, such as:

- Time and date of creation
- Authorization of the data
- Attribute size
- Purpose of the data



Data Dictionary provides information about database

- Tables
- Columns
- Constrains
- Relationship to other variables
- Variable Format
- Data Type
- And more

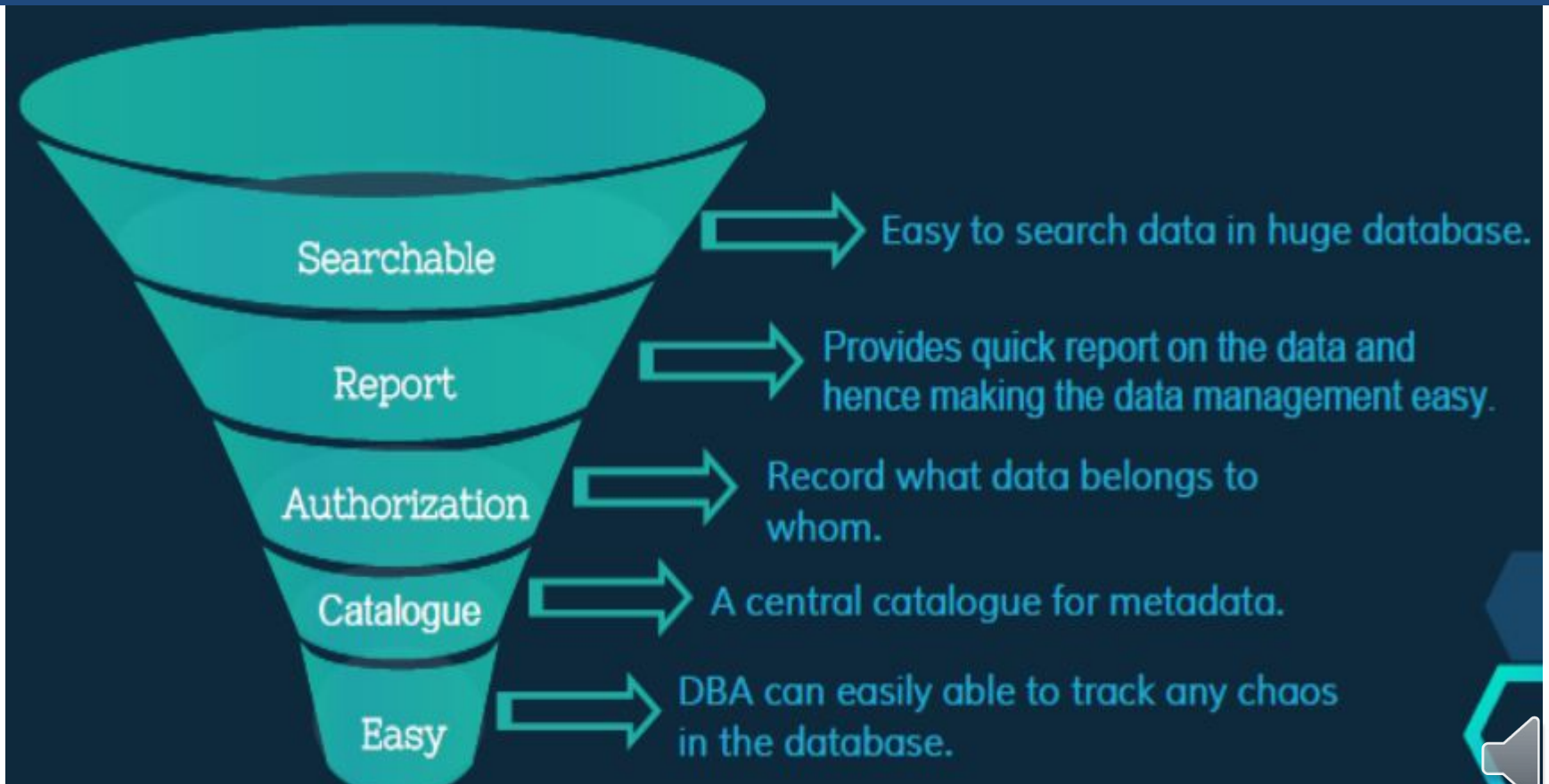


Importance

- Avoid Duplication
- Make maintenance straight forward
- To locate the error in the system



Why Data Dictionary?



Data Dictionary Format

- There exists no standard format for creating a data dictionary.
- Meta-data differs from table to table.
- The only prerequisite for a data dictionary is that it should be easily searchable.



Example - 1

- **Table:** Employees
- Holds data about organization's current and past employees. Each row represents one person.





Example - 1

Column	Data Type	Key	Constrain	Description	Example
emp_id	int	PK	AI	Primary key of table	1
first_name	varchar(20)		Not Null	First name of employee	Vibhuti
last_name	varchar(20)		Not Null	Last name of employee	Patel
nin	varchar(15)		Not Null	National Identification Number	1234567890
position	varchar(15)		Not Null	Current Postion	Assitant Professor
department	varchar(20)		Not Null	Department	BCA
gender	char(1)		Not Null	M=Male, F=Female	F
contactno	varchar(13)		Not Null	Contact Number with country code	+91 1234567890
doj	date		Not Null	Date of joining	01-01-2019



Example - 2

- **Table:** Students
- Holds data about university's students. Each row represents one person.





Example - 2

Column	Data Type	Key	Constrain	Description	Example
student_id	int	PK	AI	Primary key of table	180511201703
first_name	varchar(20)		Not Null	First name of employee	James
last_name	varchar(20)		Not Null	Last name of employee	Bond
contactno	varchar(13)		Not Null	Contact Number with country code	+91 1234567890
email	varchar(30)		Not Null	Student email id	abc@gmail.com
gender	char(1)		Not Null	M=Male, F=Female	M
department	varchar(20)		Not Null	Department	BCA
dob	date		Not Null	Date of birth	01-01-1994
address	varchar(50)		Not Null	Student address	Gagan Society
city	varchar(20)		Not Null	City	Vadodara
pincode	int		Not Null	Pincode	387000
state	varchar(20)		Not Null	State	Gujarat
counrty	varchar(20)		Not Null	Country	India



UML Overview

- UML is a standard language for **specifying, visualizing, constructing and documenting** the artifacts of software systems.
- UML was created by **Object Management Group** and UML 1.0 specification draft was proposed to the OMG in January 1997.
- The Unified Modeling Language (UML) is a general-purpose, developmental, modeling language in the field of software engineering, that is intended to provide a **standard way to visualize** the design of a system.



Introduction to UML

- According to the OMG specification, "The Unified Modeling Language (UML) is a graphical language for:
 - visualizing
 - specifying
 - constructing
 - documenting the artifacts of a software



Why UML?

- Use graphical notation: more clearly than natural language (imprecise) and code (too detailed).
- Help acquire an overall view of a system.
- UML is not dependent on any one language or technology.
- UML moves us from fragmentation to standardization.



Types of UML Diagrams

- Use Case Diagram
- Class Diagram
- Activity Diagram
- Sequence Diagram



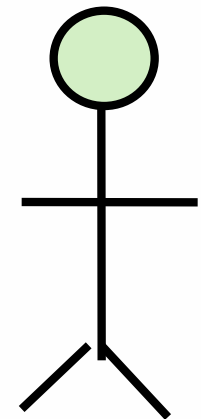
Use Case Diagram

- A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.



Notation of Use Case Diagram

- Actor: An Actor is outside or external the system. It can be a:
 - Human
 - Peripheral device (hardware)
 - External system or subsystem
 - Time or time-based event
- Represented by stick figure.



Patient



Notation of Use Case Diagram

Use Cases: A use case (Attribute) is a summary of scenarios for a single task or goal.

- A use case represents a class of functionality provided by the system as an event flow.
- An actor is who or what initiates the events involved in the task of the use case.
- Represented by oval.

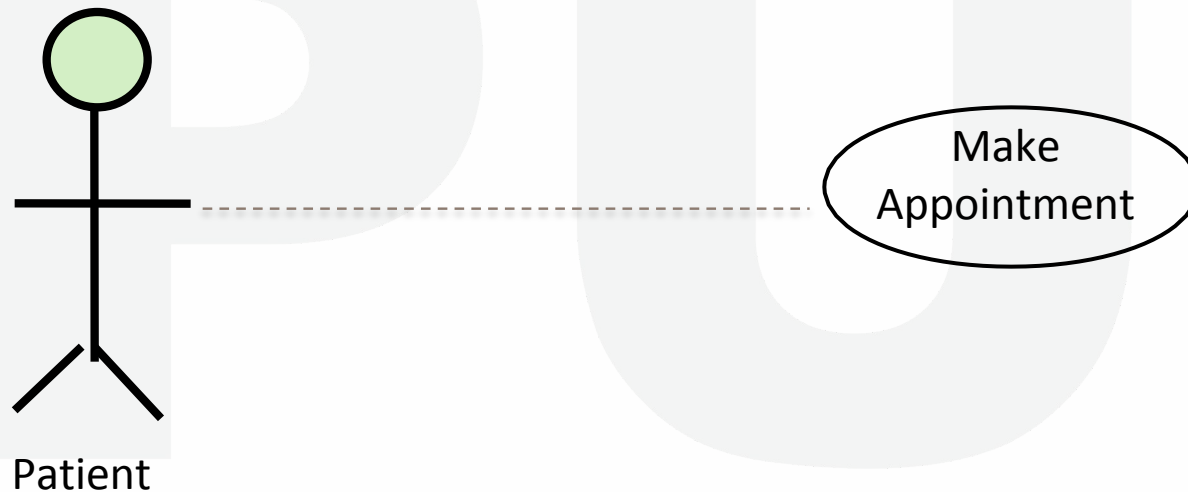
Make
Appointment



Notation of Use Case Diagram

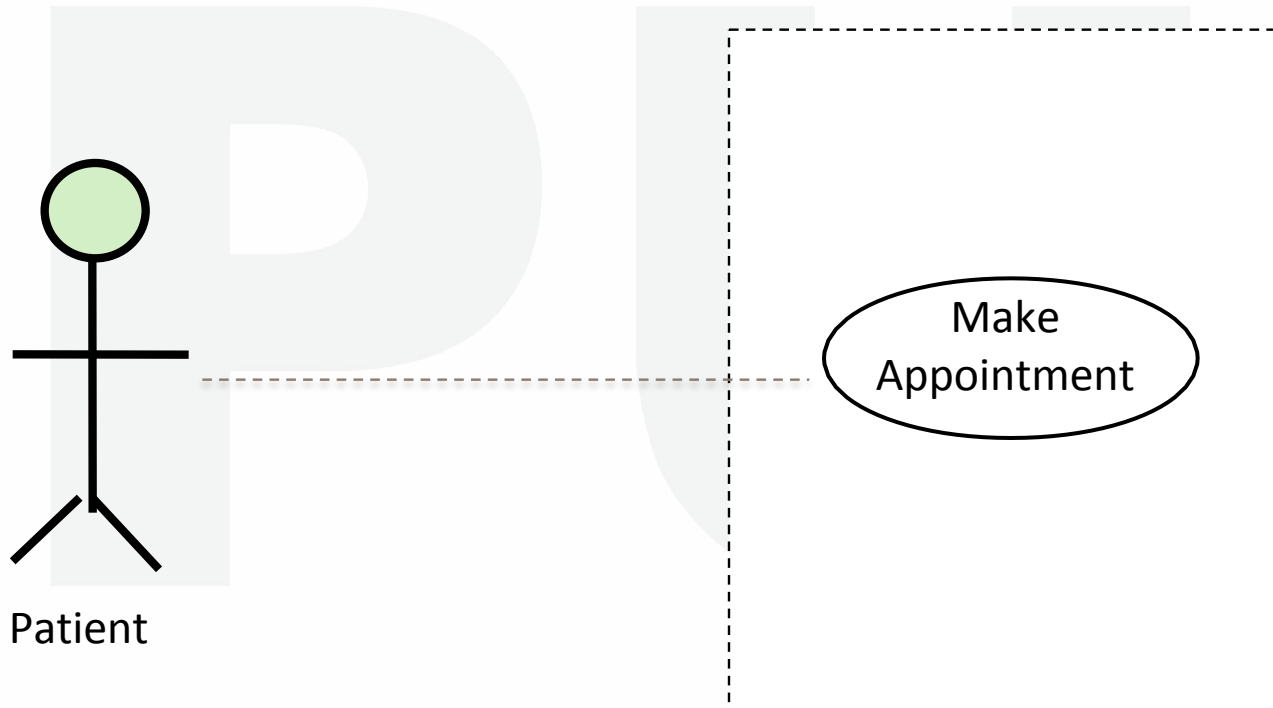
Relationships: Represent communication between actor and use case

- Represented by line or double-headed arrow line
- Also called association relationship

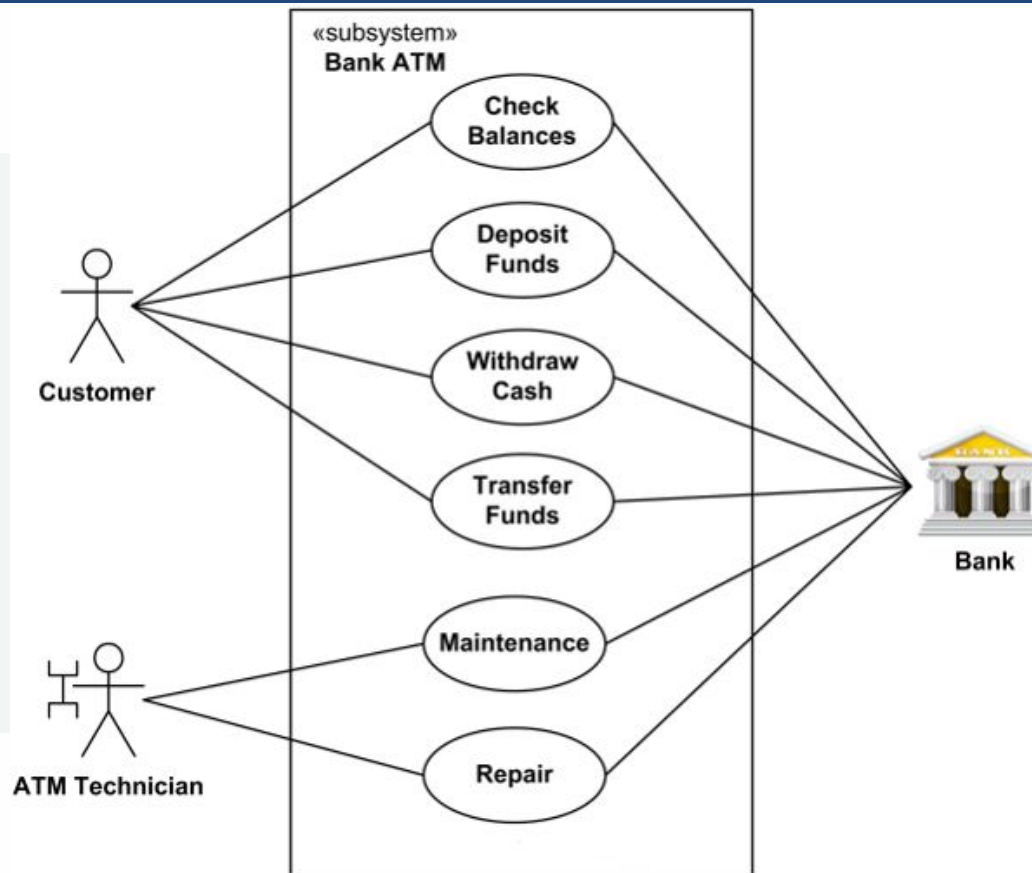


Notation of Use Case Diagram

Boundary: A boundary rectangle is placed around the perimeter of the system to show how the actors communicate with the system.



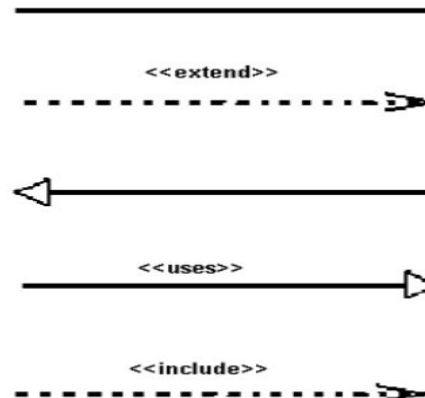
Example 1: ATM System



Use Case Relationships

Relationships: There are several Use Case relationships:

- Association
- Extend
- Generalization
- Uses
- Include



Use Case Relationships

Extend Relationships:

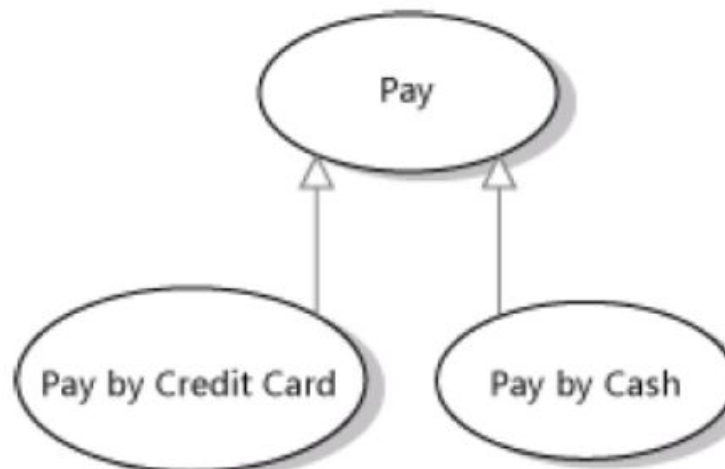
- The extended relationship is used to indicate that use case completely consists of the behavior of another use case at one or specific point.
- It is shown as a dotted line with an arrow point and labeled <<extend>>.



Use Case Relationships

General Relationships:

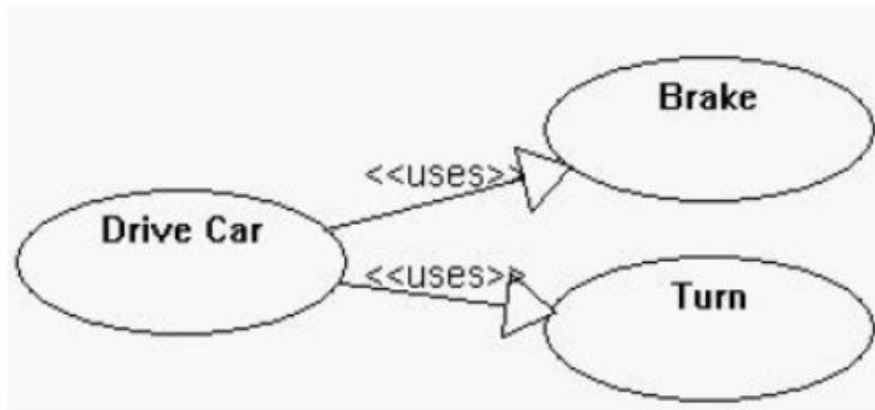
- Generalization is a relationship between a general use case and a more specific use case that inherits and extends features to it.
- It is shown as a solid line with a hollow arrow point.



Use Case Relationships

Uses Relationships:

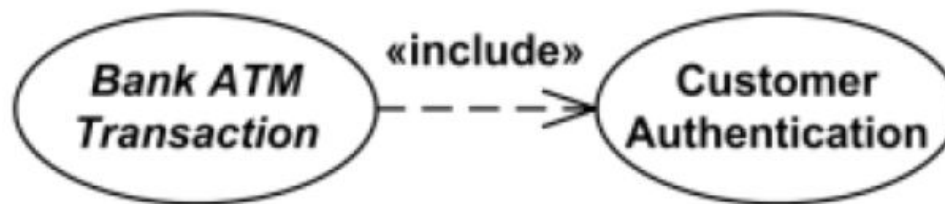
- When a use case uses another process, the relationship can be shown with the uses relationship.
- This is shown as a solid line with a hollow arrow point and the <<uses>> keyword.



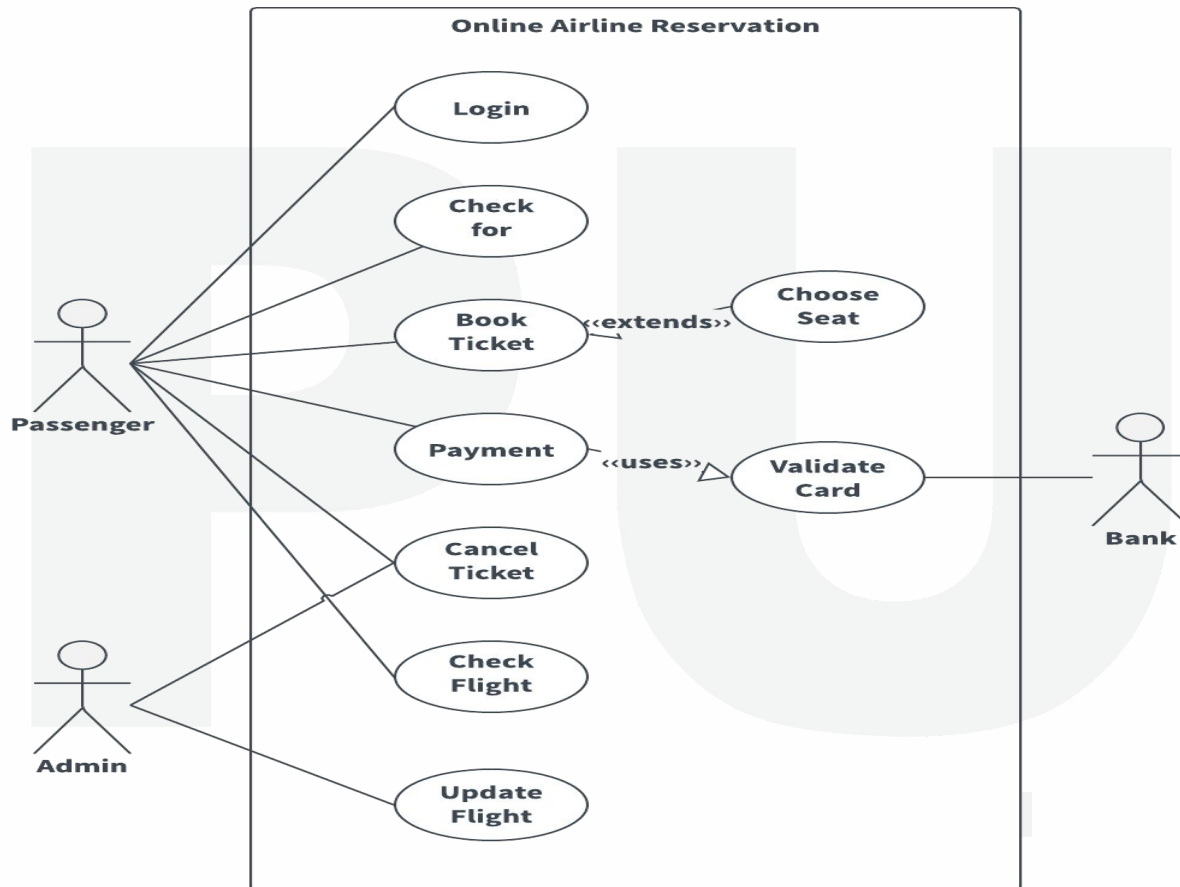
Use Case Relationships

Include Relationships:

- Include relationships insert additional behavior into a base use case.
- They are shown as a dotted line with an open arrow and the key word <<include>>



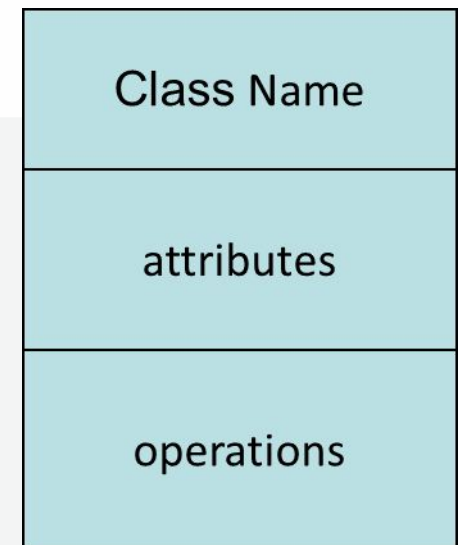
Example 2: Online Airline Reservation





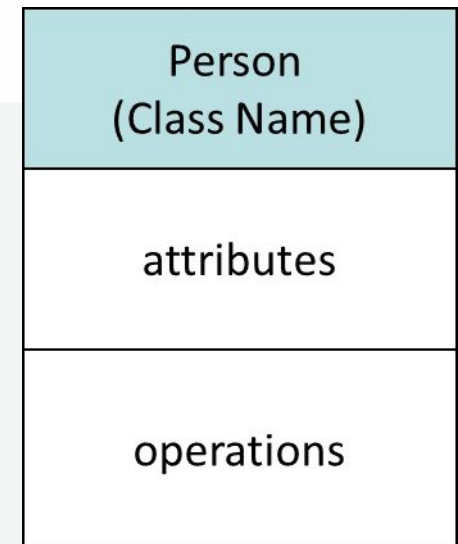
Class Diagram

- A class is a description of a set of objects that share the same attributes, operations, relationships.
- Graphically, a class is rendered as a rectangle, usually including its name, attributes, and operations in separate, designated compartments.



Class Name

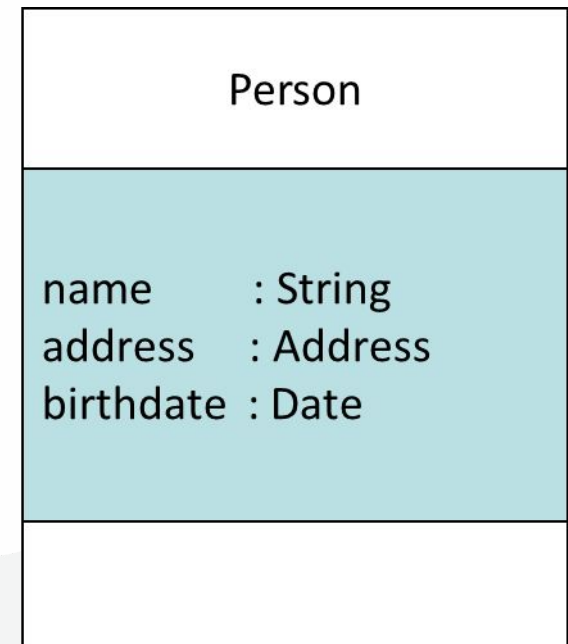
- The name of the class is the only required tag in the graphical representation of a class. It always appears in the top-most compartment.





Class Attributes

- An attribute is a named property of a class that describes the object being modeled.
- In the class diagram, attributes appear in the second compartment just below the name-compartment.





Class Attributes (Cont'd)

- Attributes are usually listed in the form:
attributeName : Type
- A *derived* attribute is one that can be computed from other attributes, but doesn't actually exist.
- For example, a Person's age can be computed from his birth date. A derived attribute is designated by a preceding '/' as in:

/ age : Date

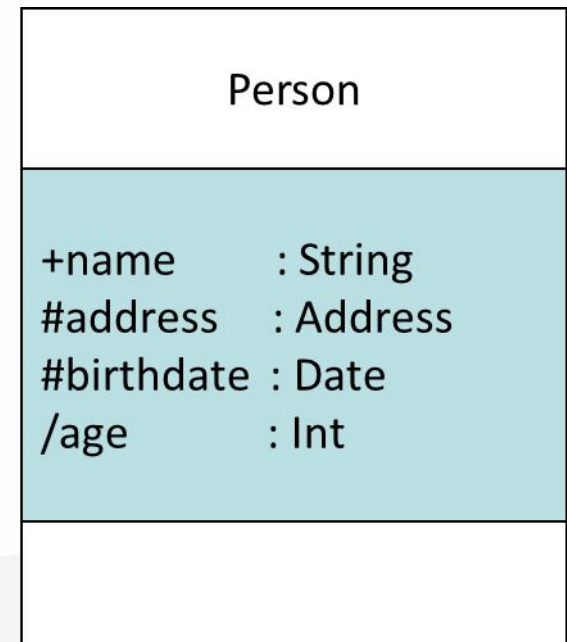
Person	
name	: String
address	: Address
birthdate	: Date
/age	: Int



Class Attributes (Cont'd)

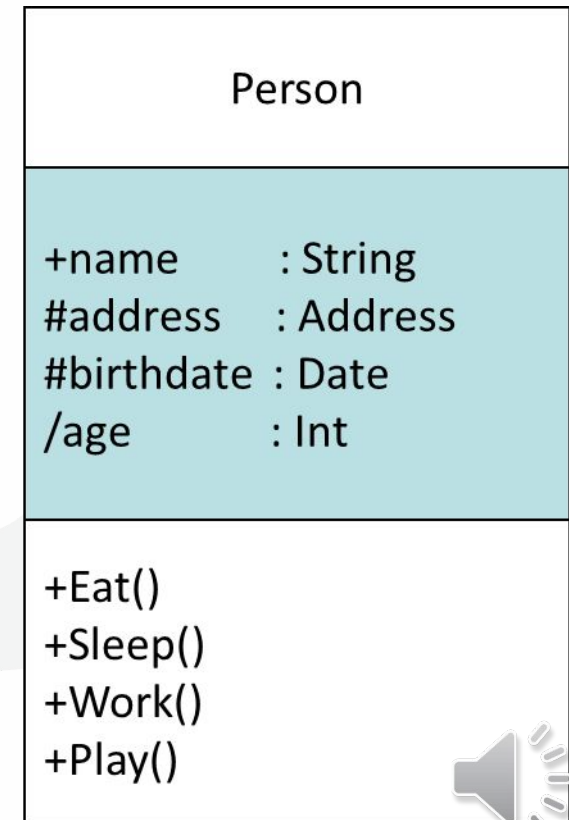
- Attributes can be:

- + public
- # protected
- - private
- / derived



Class Operations

- Operations describe the class behavior and appear in the third compartment.



Relationships

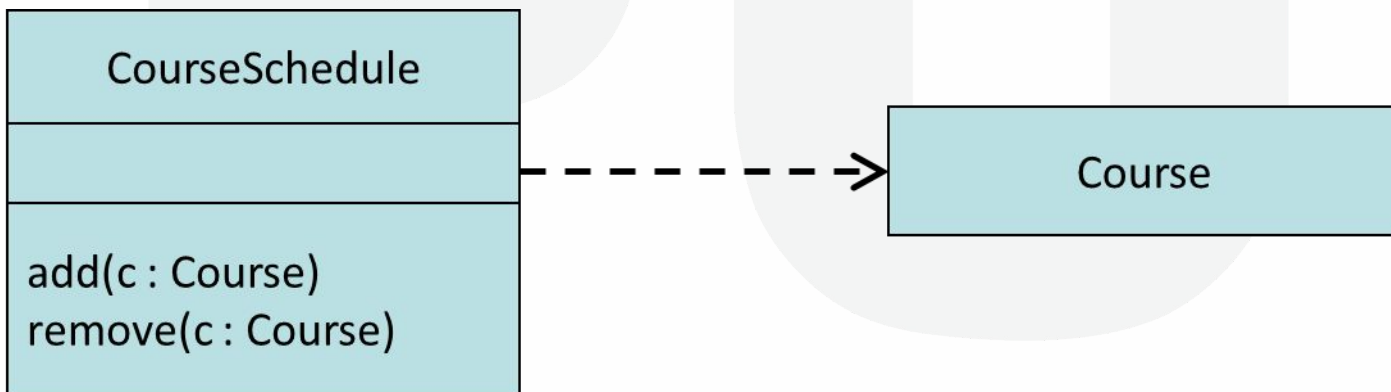
- There are three kinds of relationships in UML:

- dependencies
- generalizations
- associations



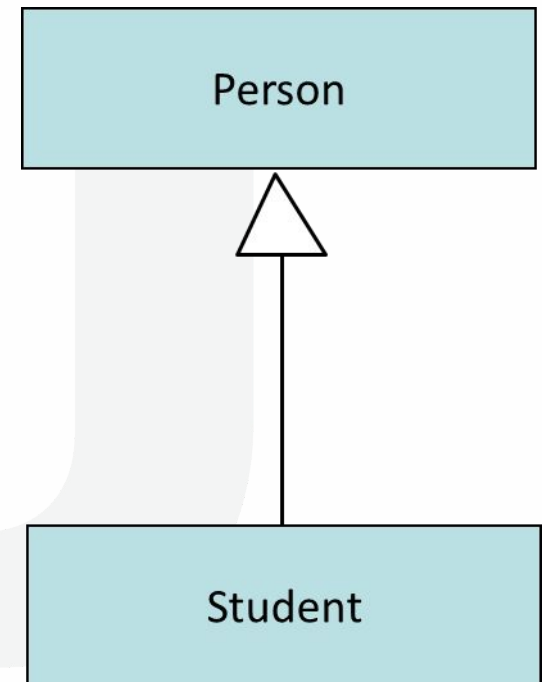
Dependency Relationship

- A dependency indicates a semantic relationship between two or more elements.
- The dependency from CourseSchedule to Course exists because Course is used in both the **add** and **remove** operations of CourseSchedule.



Generalization Relationship

- A generalization connects a subclass to its superclass.
- It denotes an inheritance of attributes and behavior from the superclass to the subclass and indicates a specialization in the subclass of the more general superclass.



Association Relationship

- If two classes in a model need to communicate with each other, there must be link between them.
- An association denotes that link.



Association Relationship (Cont'd)

- We can indicate the multiplicity of an association by adding multiplicity adornments to the line denoting the association.
- The example indicates that a Student has one or more Instructors:



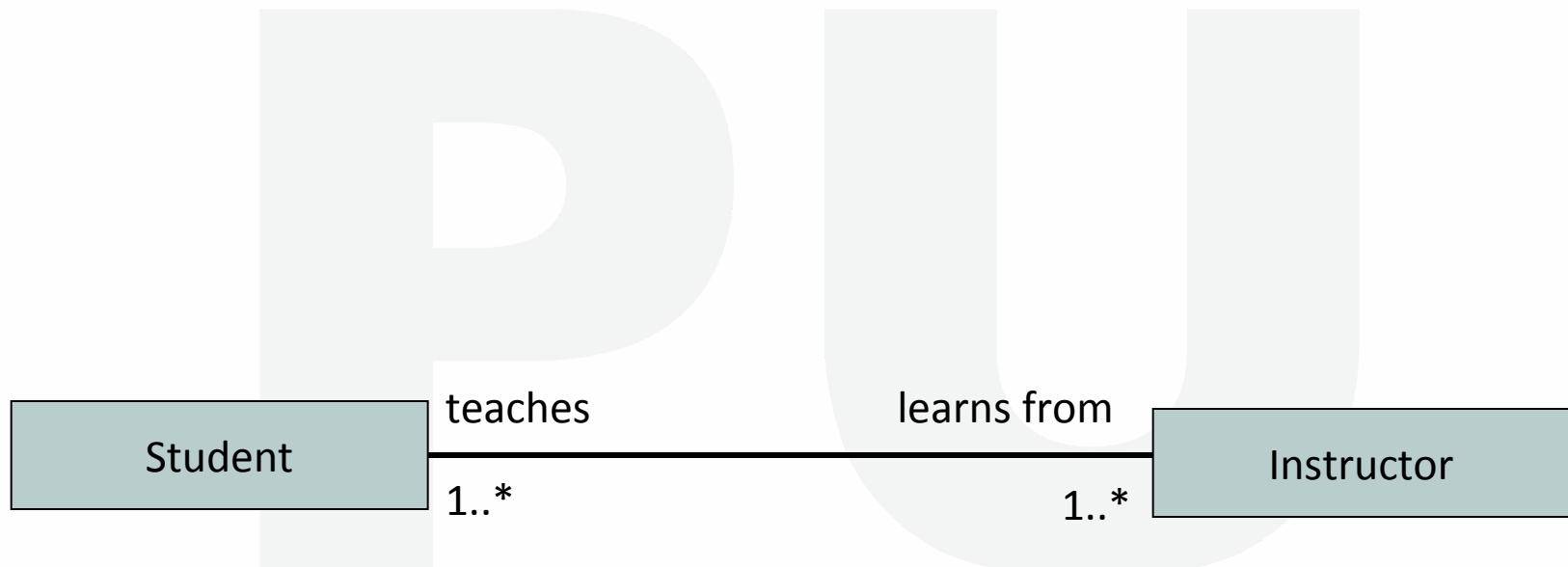
Association Relationship (Cont'd)

- The example indicates that every Instructor has one or more Students:



Association Relationship (Cont'd)

- We can also indicate the behavior of an object in an association (i.e., the role of an object) using rolenames.

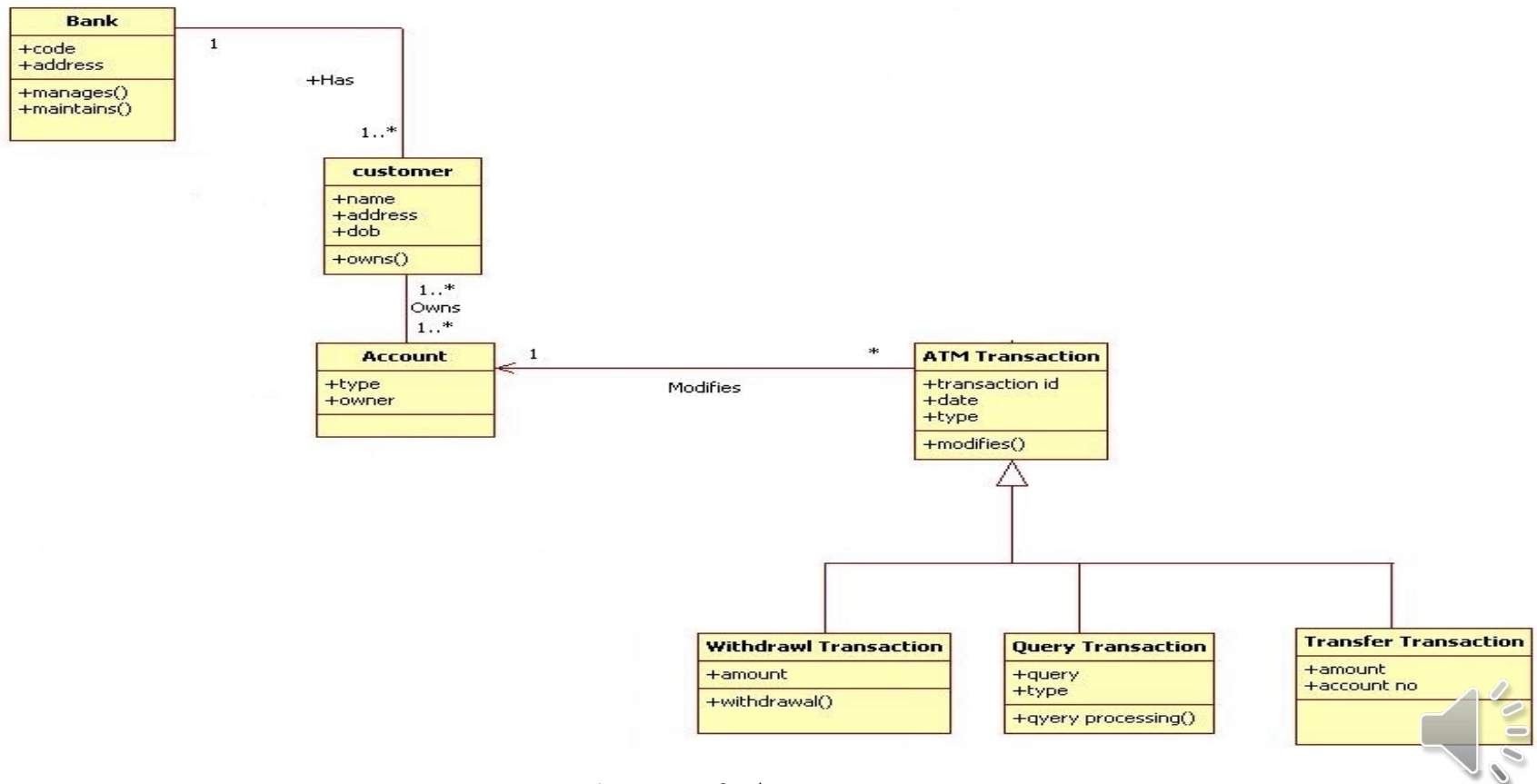


Association Relationship (Cont'd)

Symbol	Meaning
1	One and only one
0..1	Zero or one
M..N	From M to N (natural language)
*	From zero to any positive integer
0..*	From zero to any positive integer
1..*	From one to any positive integer



Example 1: ATM System



Activity Diagram

- Activity diagram is basically a flowchart to represent the flow from one activity to another activity.
- The activity can be described as an operation of the system.







Basic Component of Activity Diagram

- **Action:** A step in the activity wherein the users or software perform a given task. Actions are symbolized with round-edged rectangles.
- **Decision node:** A conditional branch in the flow that is represented by a diamond. It includes a single input and two or more outputs.
- **Control flows:** Another name for the connectors that show the flow between steps in the diagram.
- **Start node:** Symbolizes the beginning of the activity. The start node is represented by a black circle.
- **End node:** Represents the final step in the activity. The end node is represented by an outlined black circle.





Activity Diagram Symbols

Symbol	Name	Description
	Start symbol	Represents the beginning of a process or workflow in an activity diagram. It can be used by itself or with a note symbol that explains the starting point.
	Activity symbol	Indicates the activities that make up a modeled process. These symbols, which include short descriptions within the shape, are the main building blocks of an activity diagram.

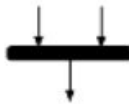


Activity Diagram Symbols



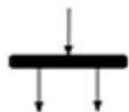
Connector
symbol

Shows the directional flow, or control flow, of the activity. An incoming arrow starts a step of an activity; once the step is completed, the flow continues with the outgoing arrow.



Joint symbol/
Synchronization
bar

Combines two concurrent activities and re-introduces them to a flow where only one activity occurs at a time. Represented with a thick vertical or horizontal line.



Fork symbol

Splits a single activity flow into two concurrent activities. Symbolized with multiple arrowed lines from a join.



Activity Diagram Symbols



Decision symbol

Represents a decision and always has at least two paths branching out with condition text to allow users to view options. This symbol represents the branching or merging of various flows with the symbol acting as a frame or container.



Note symbol

Allows the diagram creators or collaborators to communicate additional messages that don't fit within the diagram itself. Leave notes for added clarity and specification.



Activity Diagram Symbols



Send signal
symbol

Indicates that a signal is being sent
to a receiving activity.



Receive signal
symbol

Demonstrates the acceptance of an
event. After the event is received,
the flow that comes from this
action is completed.



Shallow history
pseudostate
symbol

Represents a transition that invokes
the last active state.



Option loop
symbol

Allows the creator to model a
repetitive sequence within the
option loop symbol.



Activity Diagram Symbols



Flow final
symbol

Represents the end of a specific process flow. This symbol shouldn't represent the end of all flows in an activity; in that instance, you would use the end symbol. The flow final symbol should be placed at the end of a process in a single activity flow.

[Condition]

Condition text

Placed next to a decision marker to let you know under what condition an activity flow should split off in that direction.

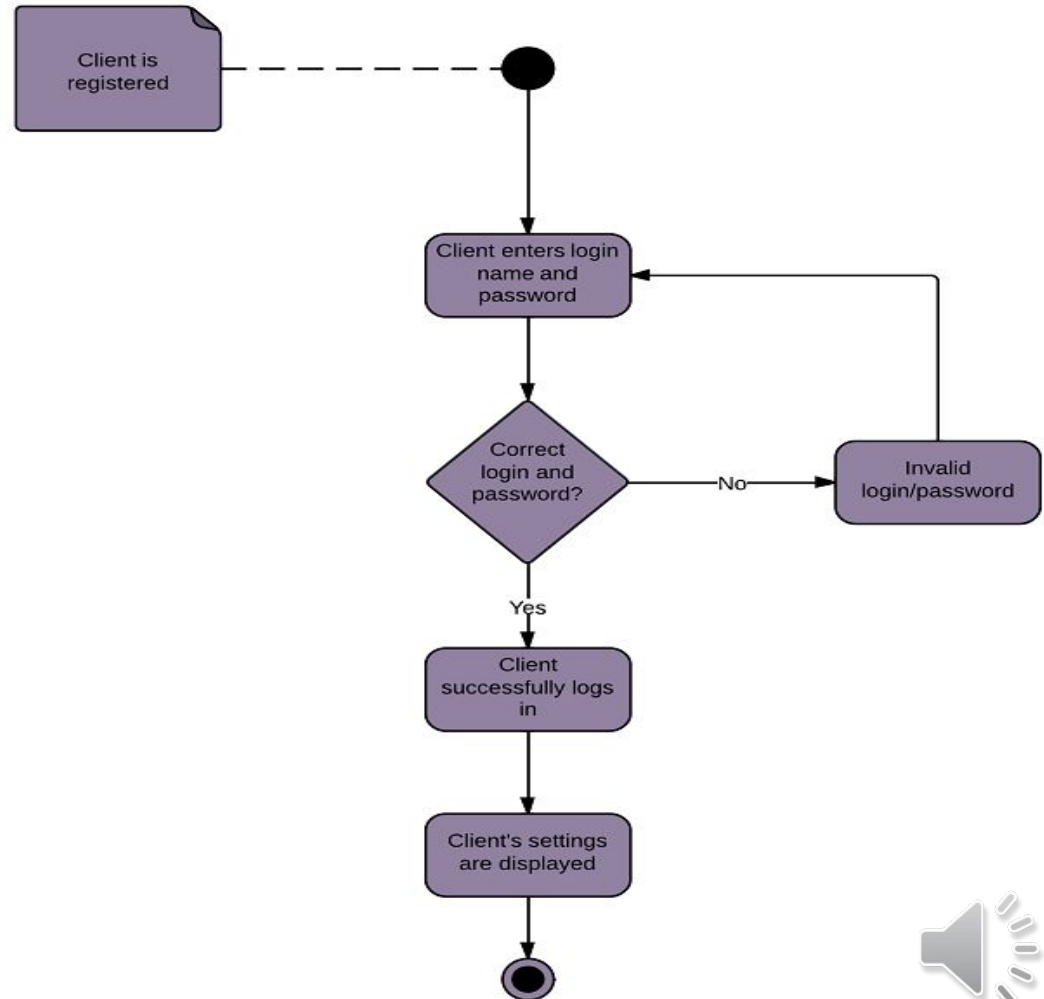


End symbol

Marks the end state of an activity and represents the completion of all flows of a process.

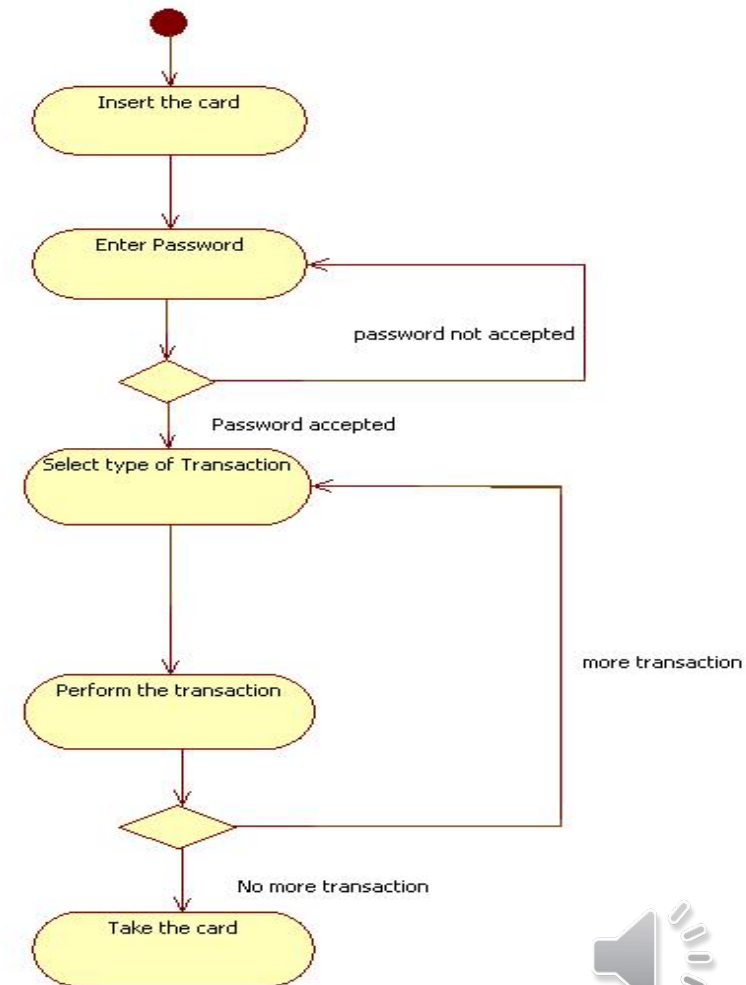


Example 1: Login



Example 2: ATM System

P



Sequence Diagram

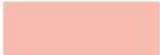

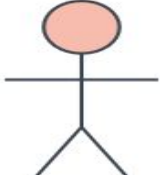
- A sequence diagram simply depicts interaction between objects in a sequential order.

PU





Symbols & Components of Sequence Diagram

Symbol	Name	Description
	Object symbol	Represents a class or object in UML. The object symbol demonstrates how an object will behave in the context of the system. Class attributes should not be listed in this shape.
	Activation box	Represents the time needed for an object to complete a task. The longer the task will take, the longer the activation box becomes.
	Actor symbol	Shows entities that interact with or are external to the system.

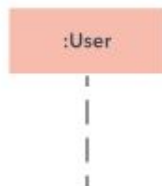


Symbols & Components of Sequence Diagram



Package
symbol

Used in UML 2.0 notation to contain interactive elements of the diagram. Also known as a frame, this rectangular shape has a small inner rectangle for labeling the diagram.



Lifeline
symbol

Represents the passage of time as it extends downward. This dashed vertical line shows the sequential events that occur to an object during the charted process. Lifelines may begin with a labeled rectangle shape or an actor symbol.



Symbols & Components of Sequence Diagram



Option
loop
symbol

Used to model if/then scenarios, i.e., a circumstance that will only occur under certain conditions.





Alternative
symbol

Symbolizes a choice (that is usually mutually exclusive) between two or more message sequences. To represent alternatives, use the labeled rectangle shape with a dashed line inside.









Symbols & Components of Sequence Diagram

Symbol	Name	Description
	Synchronous message symbol	Represented by a solid line with a solid arrowhead. This symbol is used when a sender must wait for a response to a message before it continues. The diagram should show both the call and the reply.
	Asynchronous message symbol	Represented by a solid line with a lined arrowhead. Asynchronous messages don't require a response before the sender continues. Only the call should be included in the diagram.



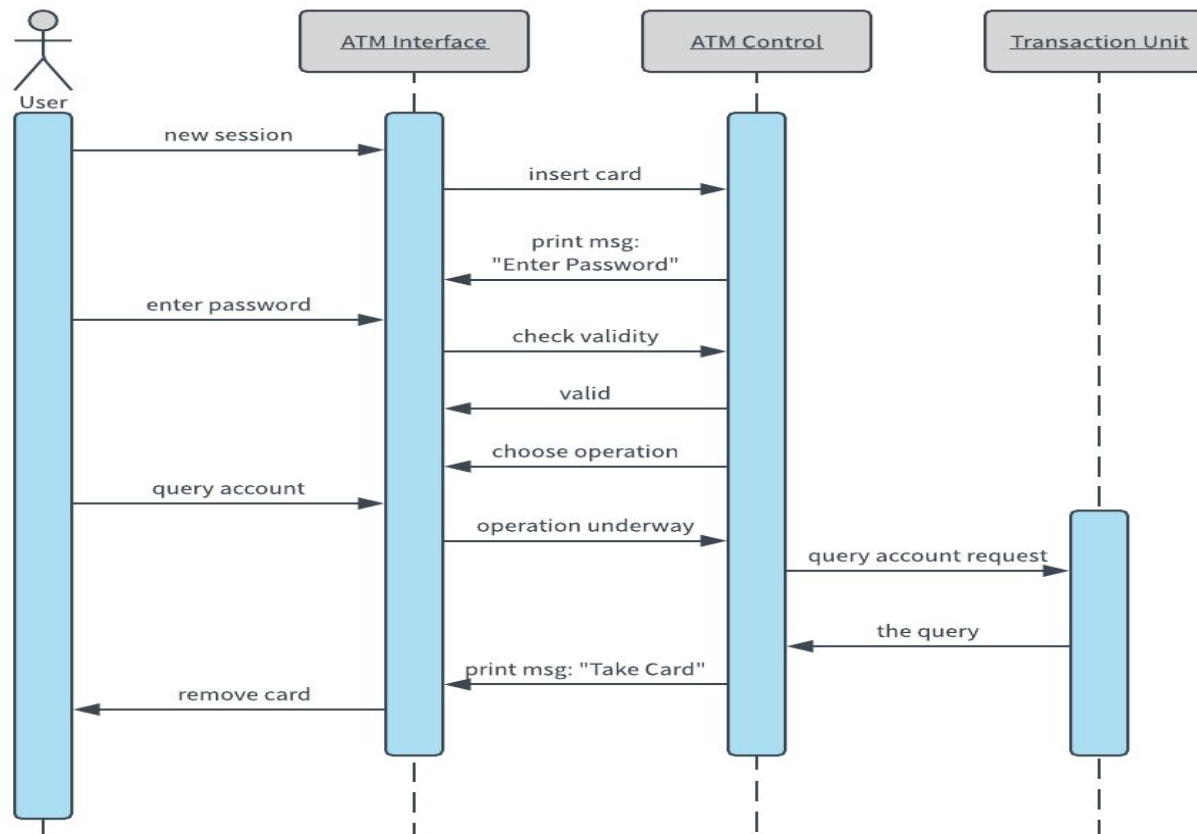


Symbols & Components of Sequence Diagram

	Asynchronous return message symbol	Represented by a dashed line with a lined arrowhead.
	Asynchronous create message symbol	Represented by a dashed line with a lined arrowhead. This message creates a new object.
	Reply message symbol	Represented by a dashed line with a lined arrowhead, these messages are replies to calls.
	Delete message symbol	Represented by a solid line with a solid arrowhead, followed by an X. This message destroys an object.



Example: ATM System



× ○ DIGITAL LEARNING CONTENT



Parul[®] University



www.paruluniversity.ac.in

