SOFTWARE ENGINEERING

(AUTONOMOUS)

H.L.Phalachandra

U
N
I
T
3

SYSTEM ANALYSIS & MODELLING

MODELLING

- We looked at some of the simple systematic procedural models like flow charts and Pseudo code as part of analysing and validating requirements
- Models are part of analysing and validating requirements. Interpreting these models **the same way by everyone** **every time** needs some semi formal or formal semantics we will discuss this a little later
- If we specify the requirements with the usage of more precise and formal semantics, we could have diagrams associated with some models to be generated using some tools. These tools also generate code, so these diagrams could be linked to the methods that generate them and thus operationalizing the construction process.
- We will look at general guidelines, approaches used while looking at models. We will also look at some of the more prominent models

ANALYSIS MODELS : THUMB RULES

- The analysis model should focus on requirements that are visible within the problem or business domain
  - The level of abstraction should be relatively high
- Each element of the analysis model should add to an overall understanding of software requirements and provide insight into the following
  - Information domain, function, and behavior of the system
- The model should delay the consideration of infrastructure and other non-functional models until the design phase
  - First complete the analysis of the problem domain
- The model should minimize coupling throughout the system
  - Reduce the level of interconnectedness among functions and classes
- The model should provide value to all stakeholders
- The model should be kept as simple as can be

DECOMPOSITION

- A logical first step in creating a model is to identify the essential components of the system being modeled. This analysis is called as decomposition

- The components identified by decomposition vary considerably, depending upon the view of the system

- From this high-level view, software models can be grouped into two categories
  - Structural models
  - Behavioral models

Modeling Types in RE

One approach of looking at different models

Types of Models

Models

DFDs,CFDs, SD etc

Dynamic models – describe system behavior

Static models – describe system structure

Class Models, User Interactions

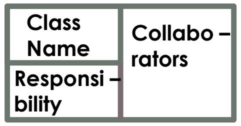
Modeling Notations

UML – used to depict use cases, state changes, and event sequences etc.

Integrated definition (IDEF) Languages
IDEF0 - functional modeling
IDEF1X - information modeling

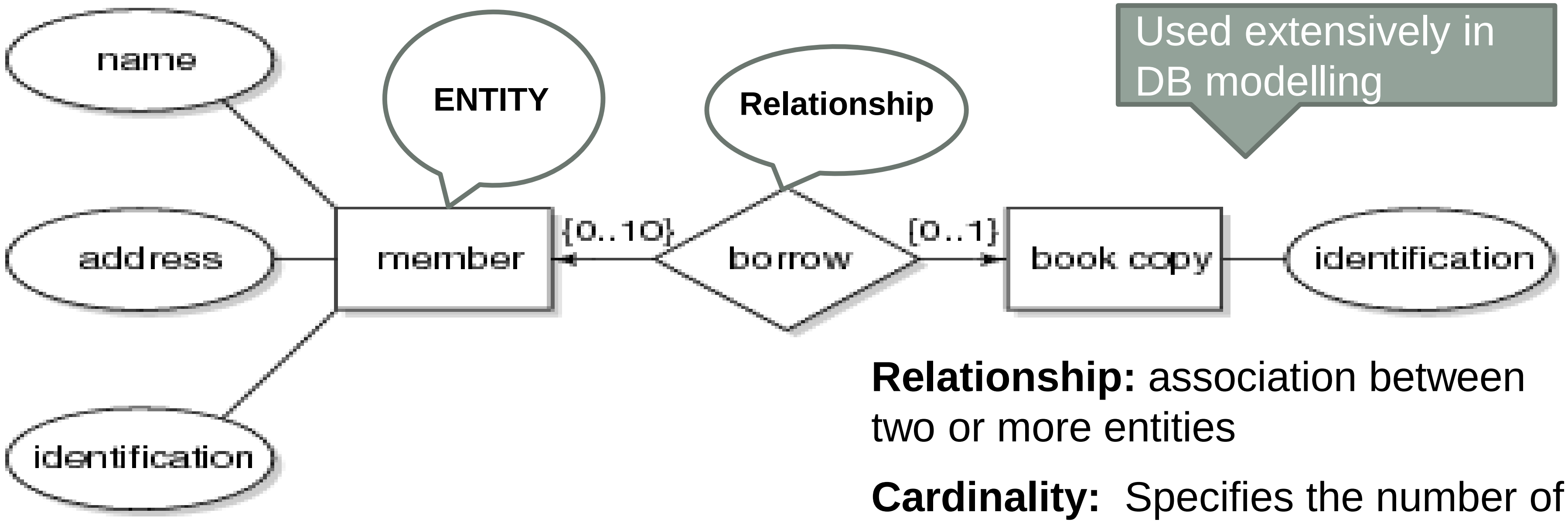
CLASSIC SYSTEM MODELING TECHNIQUES

- Some classic modeling techniques:
  - Entity-relationship modeling
    - Models Logical semantic structure of the object
    - UML class diagrams for object oriented system
  - Finite state machines
    - Models states and their transitions
    - Finite state machines for object oriented systems too.
  - Data flow diagrams
    - Models processes and set of data flows which connects these processes
  - CRC (Class-Responsibility-Collaborators) cards



- Object-oriented modeling: and variety of UML diagrams

ENTITY-RELATIONSHIP DIAGRAM



Relationship: association between two or more entities

Cardinality: Specifies the number of occurrences of one object that can be related to no of occurrences of another object

Modality: Necessary (1) or not (0)

Limitations : Assumes info content can readily be represented in a relational database else is Inadequate.

Entity: distinguishable object of some type

Entity type: type of a set of entities

Attribute: type of a set of attribute values

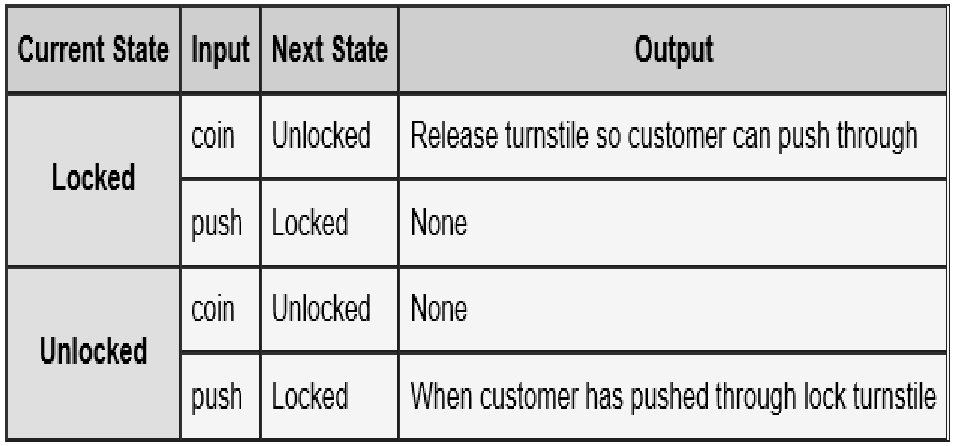
Attribute value: piece of information (partially) describing an entity

STATE TRANSITION DIAGRAM

- This represents system's responses to stimuli so are often used for modelling real-time systems.
- Each state models the behaviour of the system in response to external and internal events
- Could be a hierarchy of State Machines

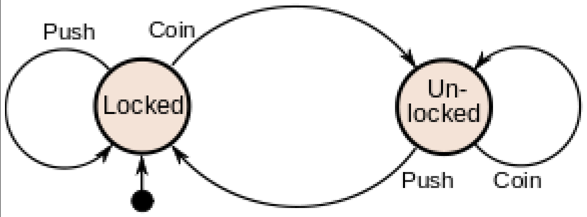
Turnstile

State transition table



Current State	Input	Next State	Output
Locked	coin	Unlocked	Release turnstile so customer can push through
	push	Locked	None
Unlocked	coin	Unlocked	None
	push	Locked	When customer has pushed through lock turnstile

State transition diagram



Eg. State transition diagram of a Turnstile

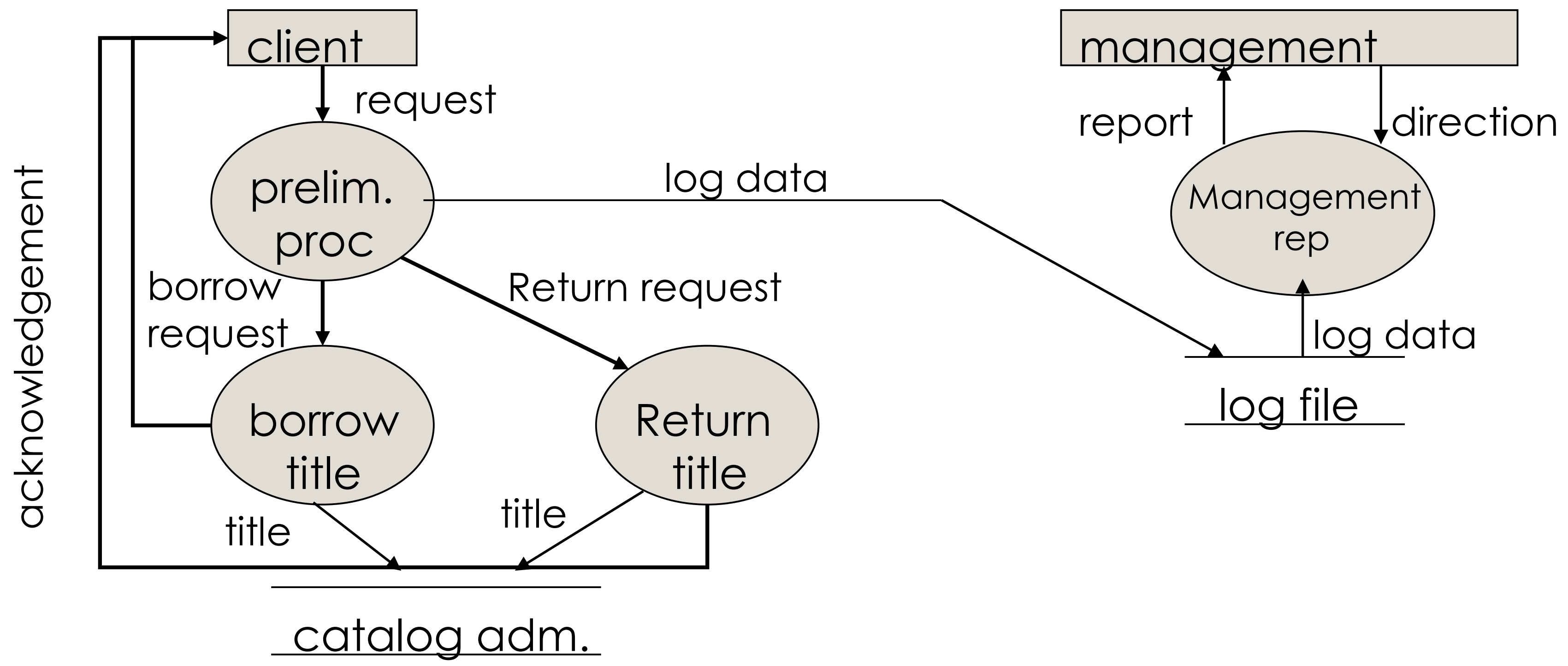
DATA FLOW DIAGRAMS – 1 (2)

DFD is a functional decomposition with respect to the flow of data. A module transforms (like a black box) some input stream to an output stream.

There are 4 types of data entities in the DFD

- External Entity
  - Source and Destination of a transaction.
  - These are located outside the domain considered in the DFD.
  - Depicted as Squares in the diagram
- Processes
  - Transforms the data
  - Depicted as circles
- Data Stores
  - These lie between processes and are places where data structures reside
  - Depicted between two parallel lines
- Data Flows
  - Paths where data structures travel between processes, entities and data stores
  - Depicted as an Arrow

EXAMPLE DATA FLOW DIAGRAM – 2 (2)



external entities	▭	data flows	→
processes	●	data stores	═

WHAT IS AN OBJECT?

Modeling viewpoint:

- It’s a model of part of the world
- It has an identity which distinguishes it from other objects in the world

  Object identity Would be a tag or address which identifies an object as they could be in different location in memory.

  - Could be instances of ADT (abstract data types)
  - Could contain
    - states as a set of variables of the ADTs,
    - operations to modify and inspect the state which would be the only way to access this object. So these operations act as interfaces to these objects. So object could be looked at as.

      Object = Identity + variables + operations

      Or Object = Identity + state + behavior

- It has properties which can be found by investigating the object

WHAT IS AN OBJECT?

- Philosophical viewpoint: existential abstractions
  - Everything is an object
  - Entities such as numbers, dates having eternal existence are referred to as values
- Software engineering viewpoint: data abstraction
- Implementation viewpoint: structure in memory
- Formal viewpoint: state machine

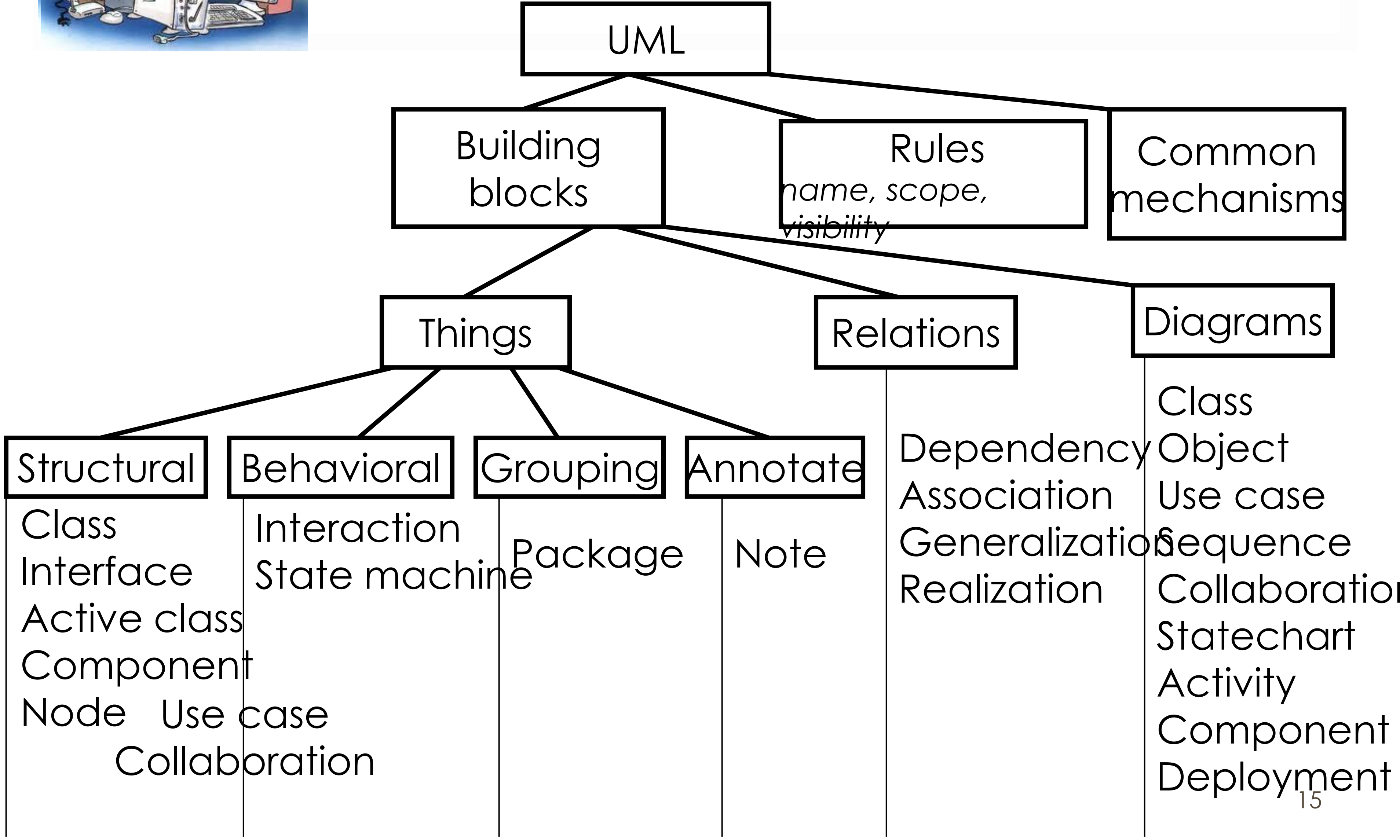
OBJECT ORIENTED MODEL

- Models the requirements elicited, for communication between the customers and system developers
- Objects and Classes are identified (attributes and methods are defined)
- Class hierarchy is defined
- Object to Object relationships (object connections) should be represented
- Object behaviour should be modelled
- All of these represented using Object diagrams

UNIFIED MODELING LANGUAGE (UML)

- **Controlled by OMG consortium:** Object Management Group
- **Used with Modeling Systems**
- **UML 2 has 13 diagram types**
  - Static diagrams depict static structure
  - Dynamic diagrams show what happens during execution
- **Most often used diagrams:**
  - class diagram: 75%
  - Use case diagram and communication diagram: 50%
  - Often loose semantics

CONCEPTUAL MODEL OF UML



Conceptual model of UML

Things are abstractions in the model.

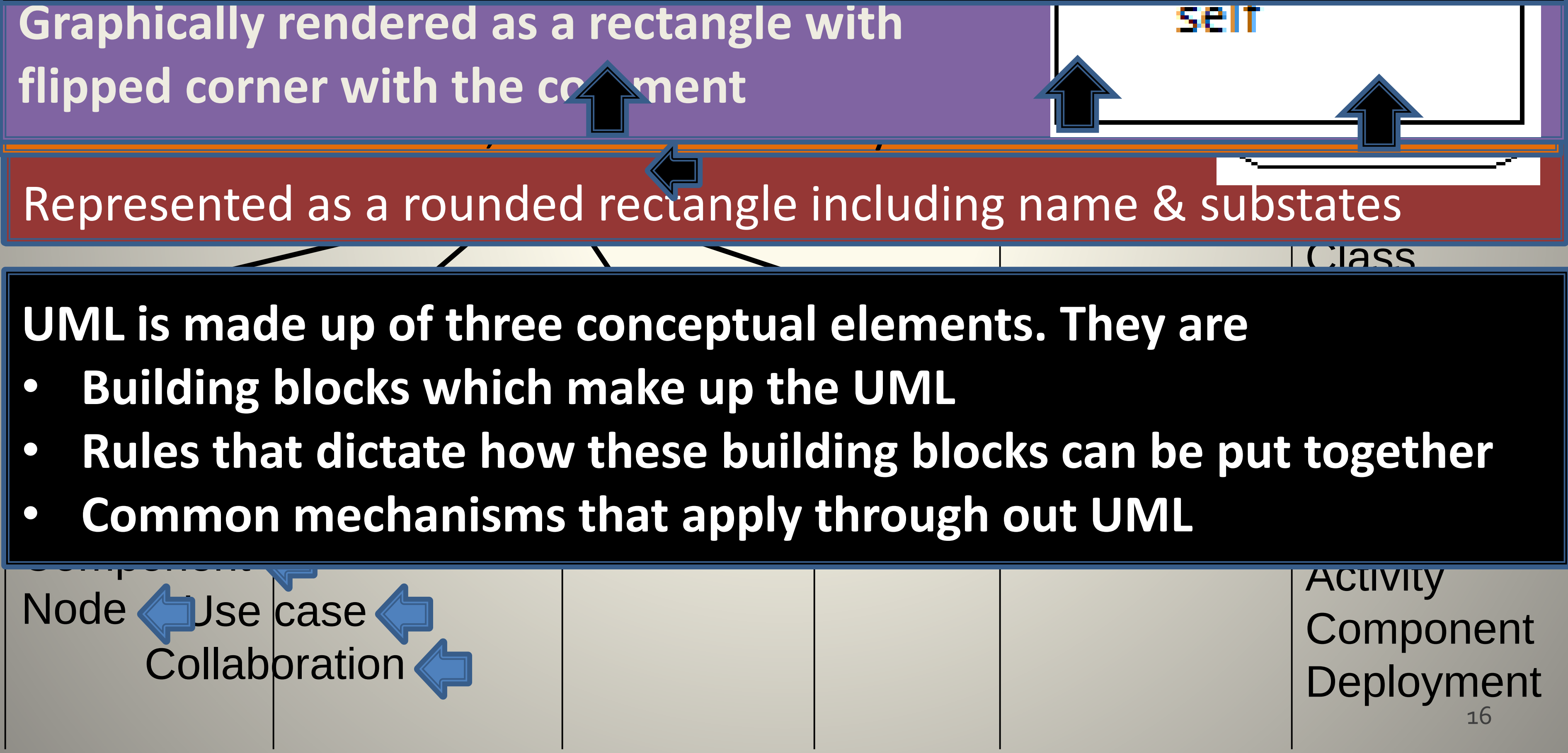
There are four kinds of things or abstraction

Graphically rendered as a rectangle with flipped corner with the comment

Represented as a rounded rectangle including name & substates

UML is made up of three conceptual elements. They are

- Building blocks which make up the UML
- Rules that dictate how these building blocks can be put together
- Common mechanisms that apply through out UML



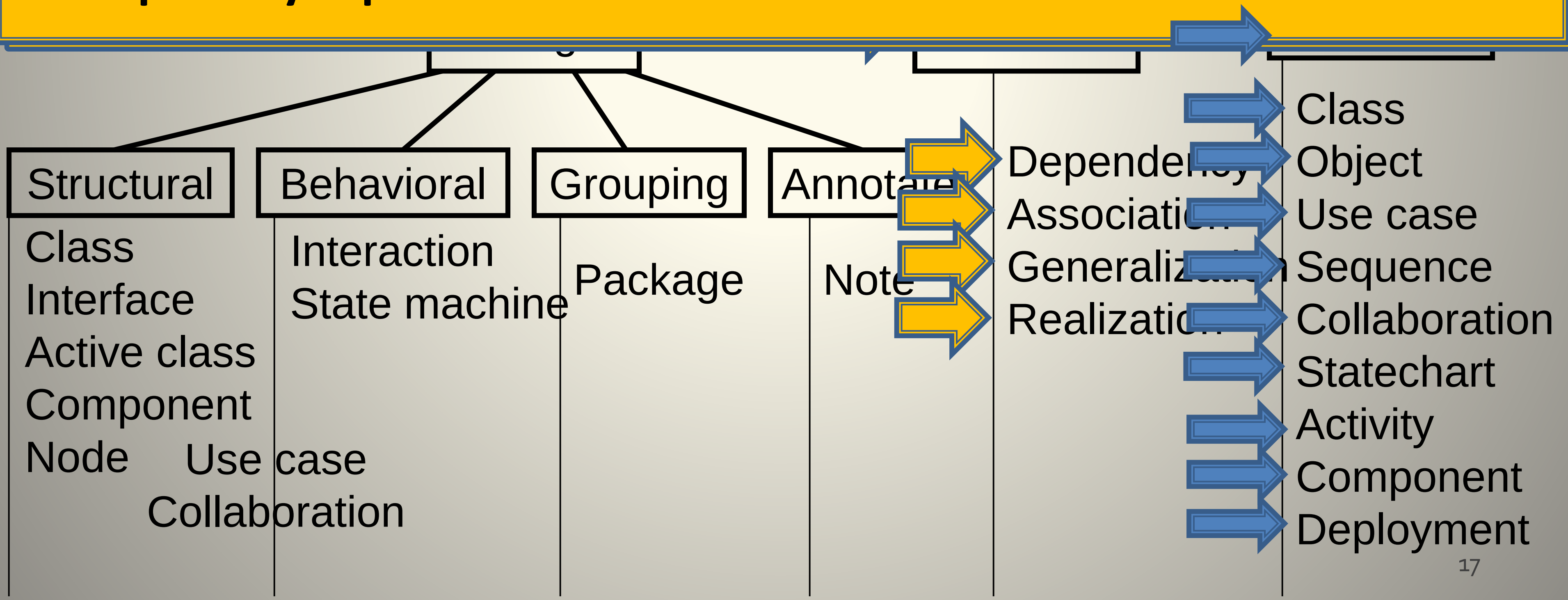
Conceptual model of UML

Deployment Diagram

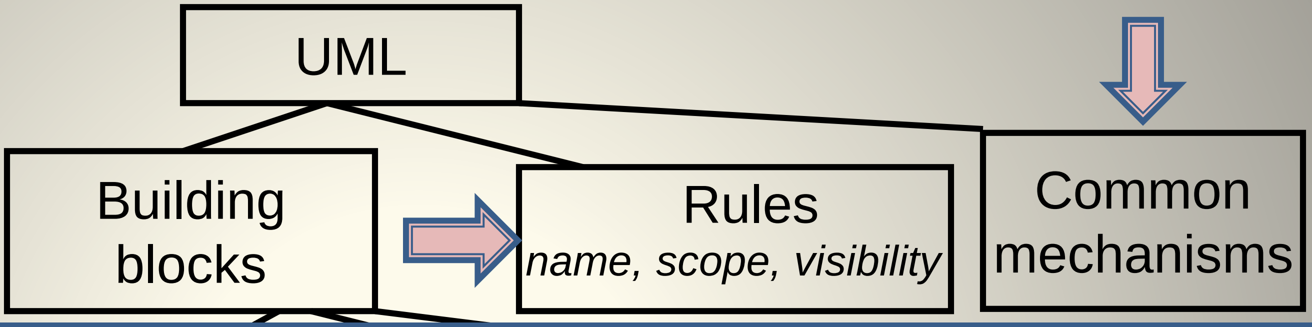
Shows the configuration of run-time processing nodes and the components that are deployed on them

internal structure

- **Graphically represented as a dotted line with hollow**



Conceptual model of UML

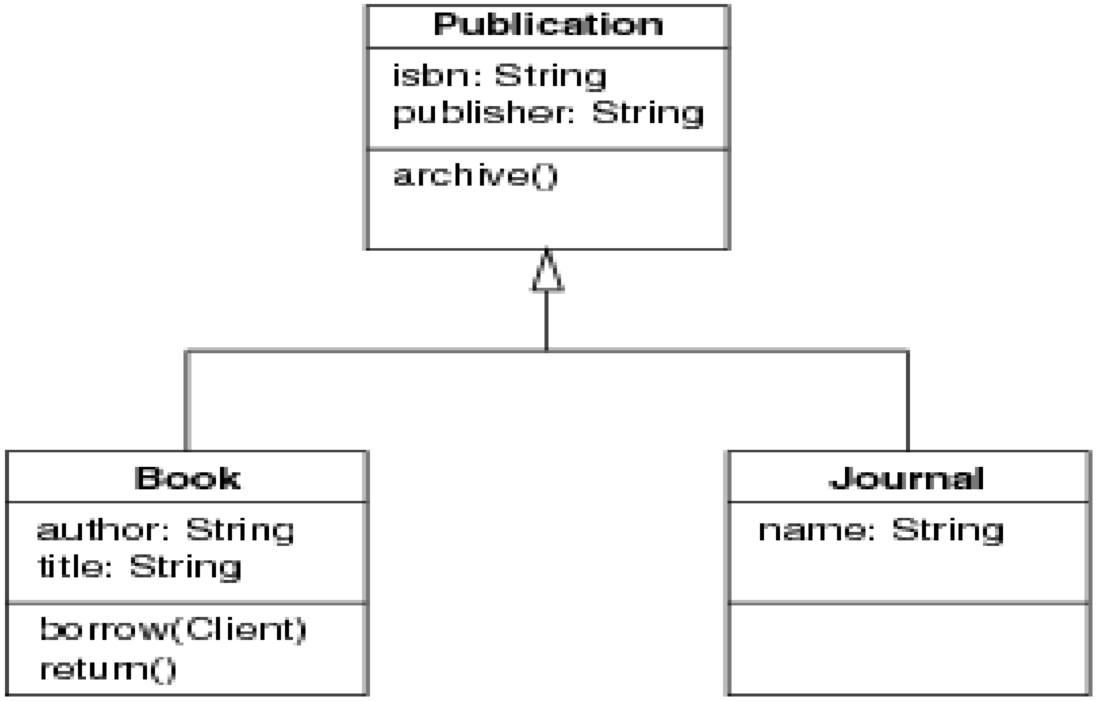


UML has four common mechanisms that apply consistently throughout the language

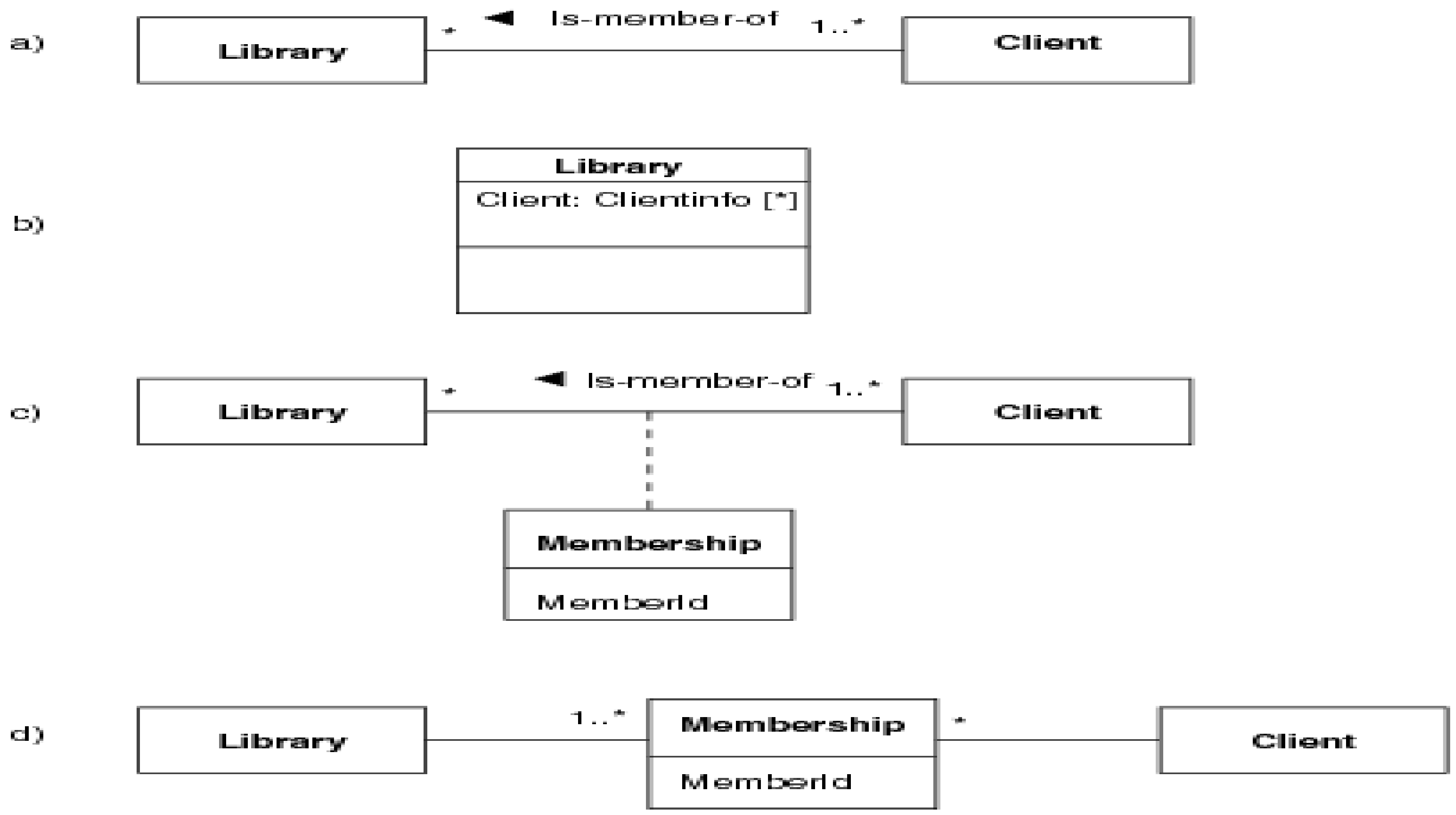
- **Specifications :** Every graphical notation has a textual statement of the syntax and semantics to state the systems details
- **Adornments :** UML elements have direct graphical notations for ease of drawing and can have other details rendered as graphical and textual adornments
- **Common Divisions :** There can be are divided views in modelling OO systems like you could define a Class and object, Interfaces and implementation etc.
- **Extensibility mechanism :** UML provides mechanisms like stereotypes (extends vocabulary), tagged values (extends properties) and constraints (extends semantics)

UML CLASS DIAGRAM

- Depicts the static structure of a system
- Most common example: subclass/superclass hierarchy
- Shows mutual properties between two or more entities (ER relationships, often called associations in OO)

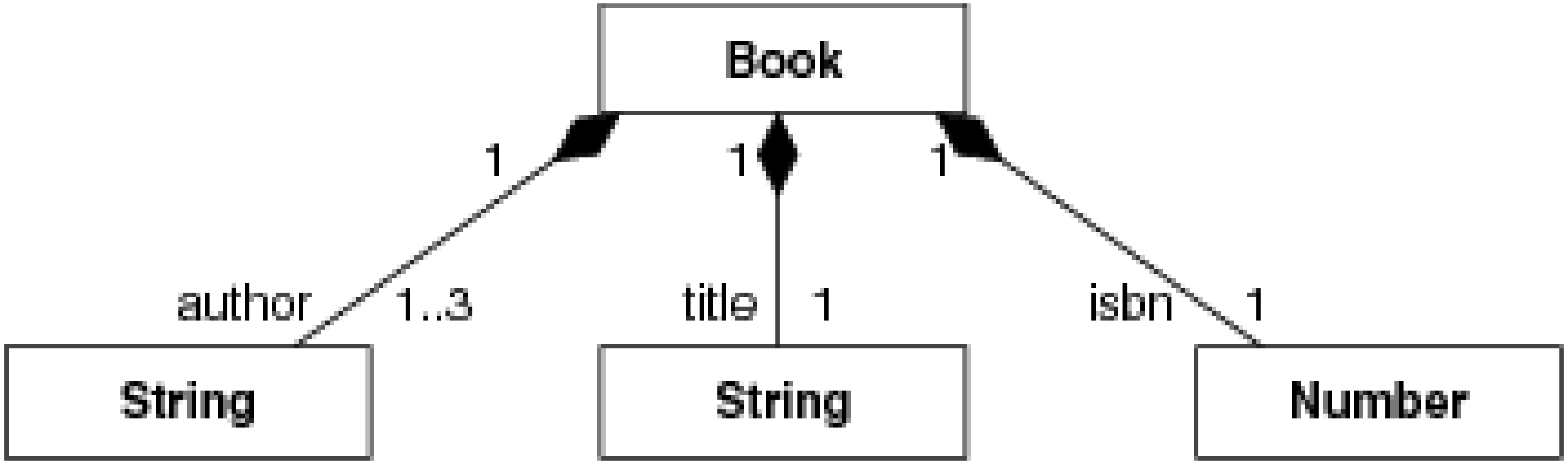


Eg: CLASS DIAGRAM (ASSOCIATION)

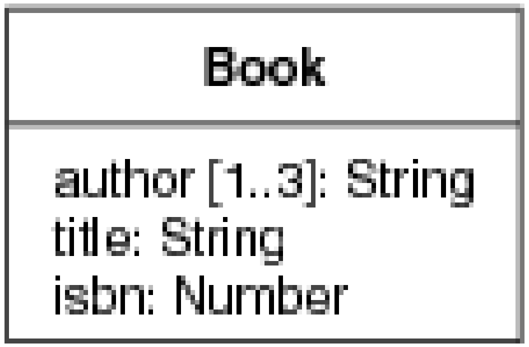


Eg CLASS DIAGRAM (COMPOSITION)

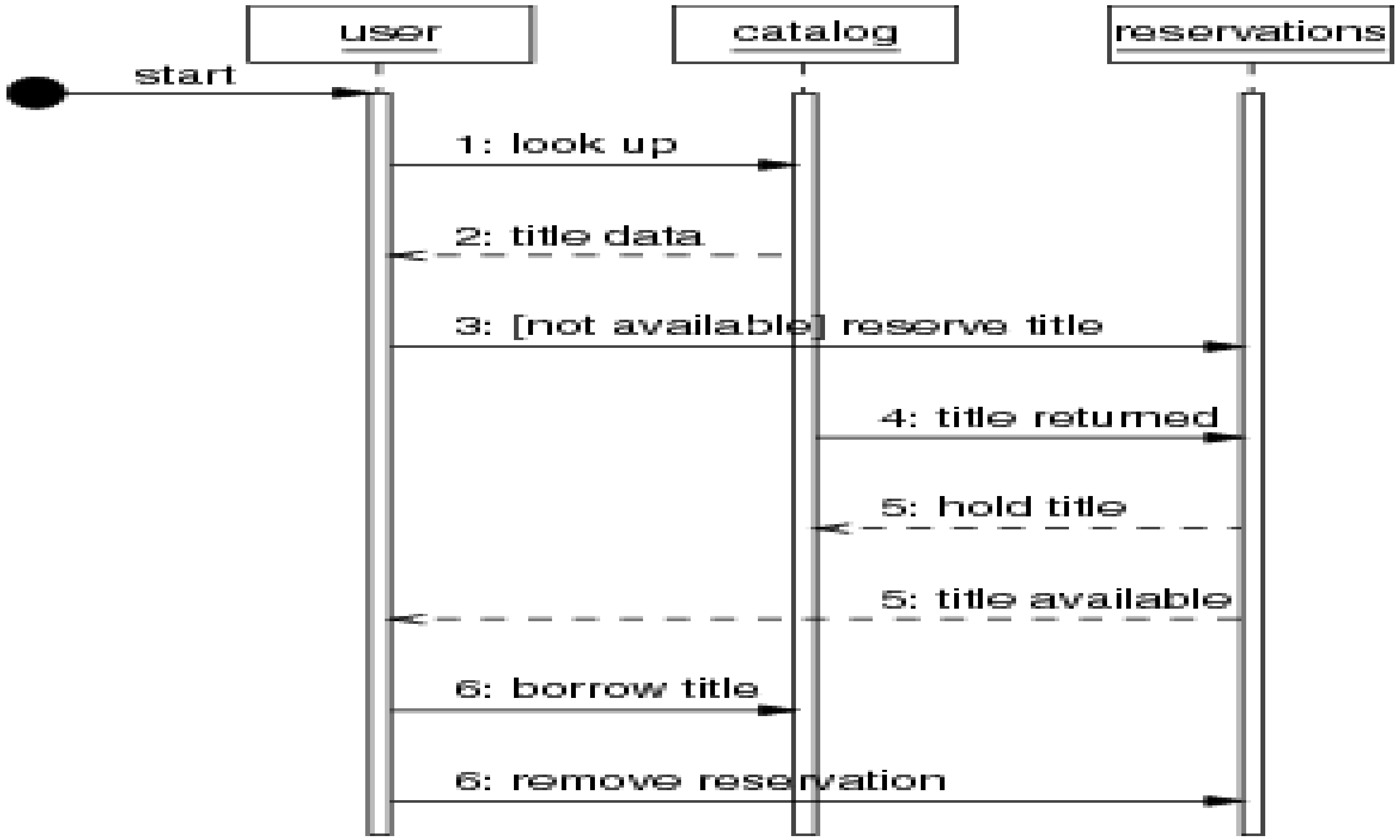
a)



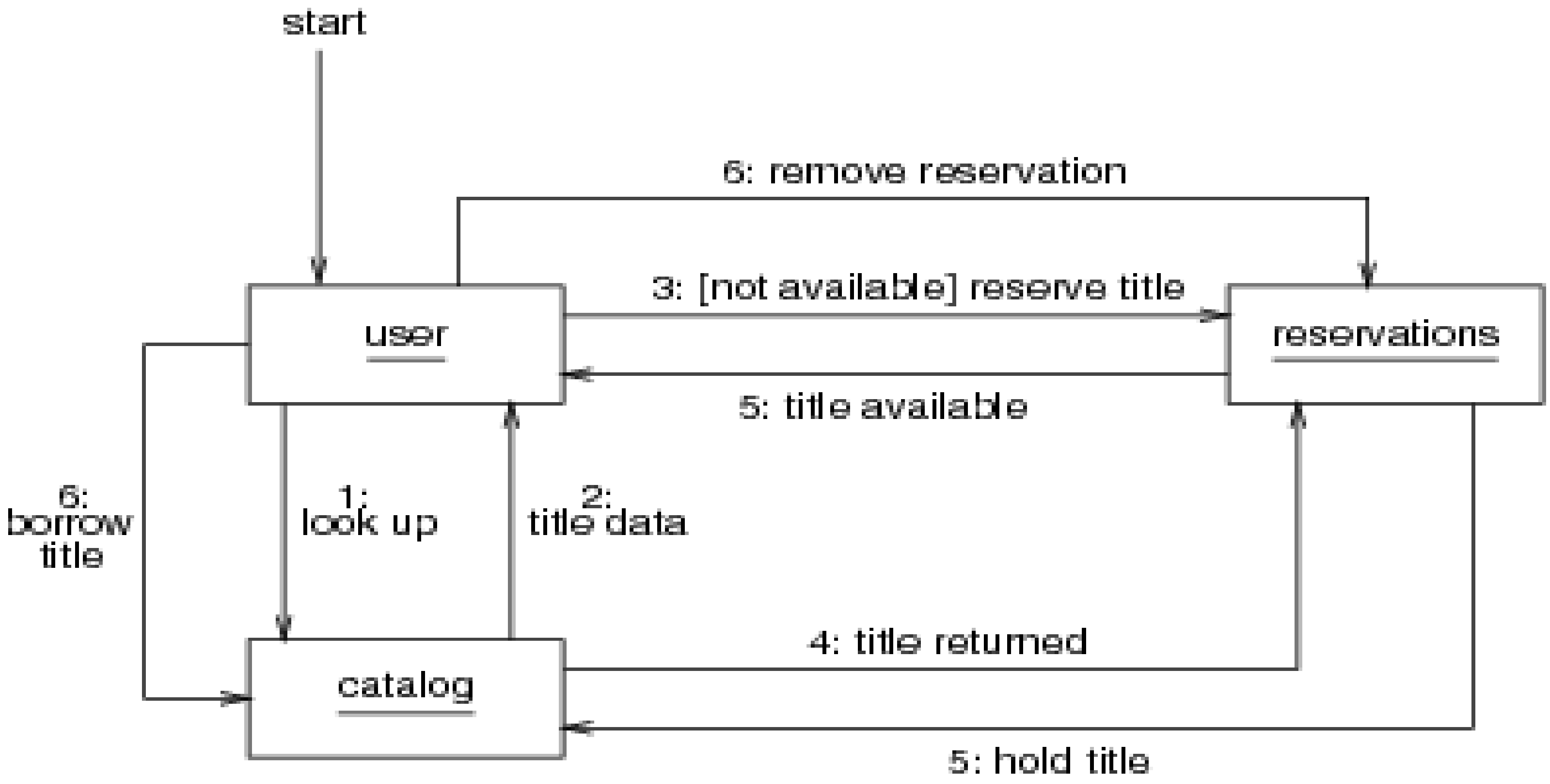
b)



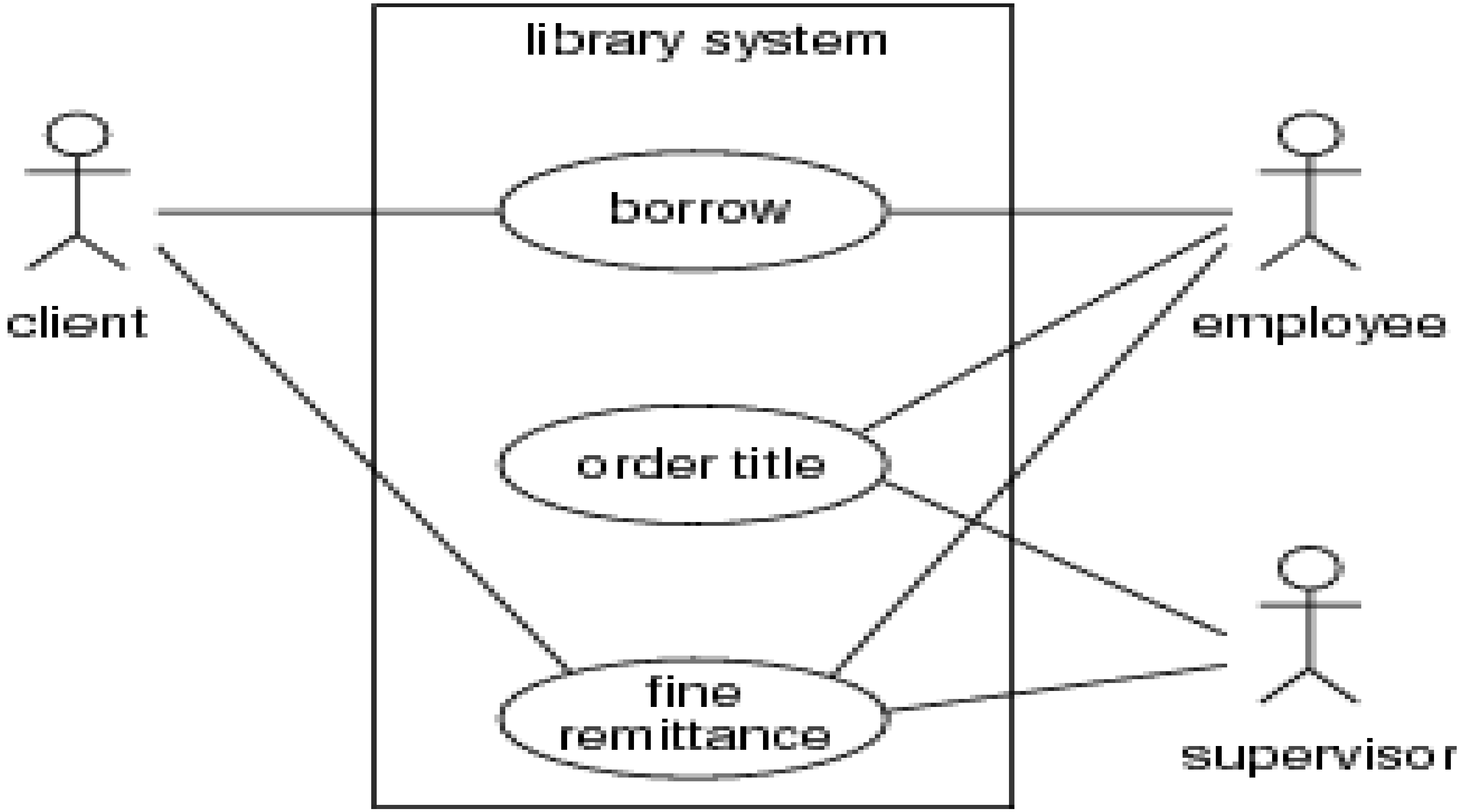
EXAMPLE SEQUENCE DIAGRAM



EXAMPLE COMMUNICATION DIAGRAM



USE CASE DIAGRAM



SUMMARY

- Classic notations:
  - Entity-relationship diagrams
  - Finite state machines
  - Data flow diagrams
  - CRC cards
- Unified Modeling Language (UML)
  - evolved from earlier OO notations
  - 13 diagram types
  - widely used

END

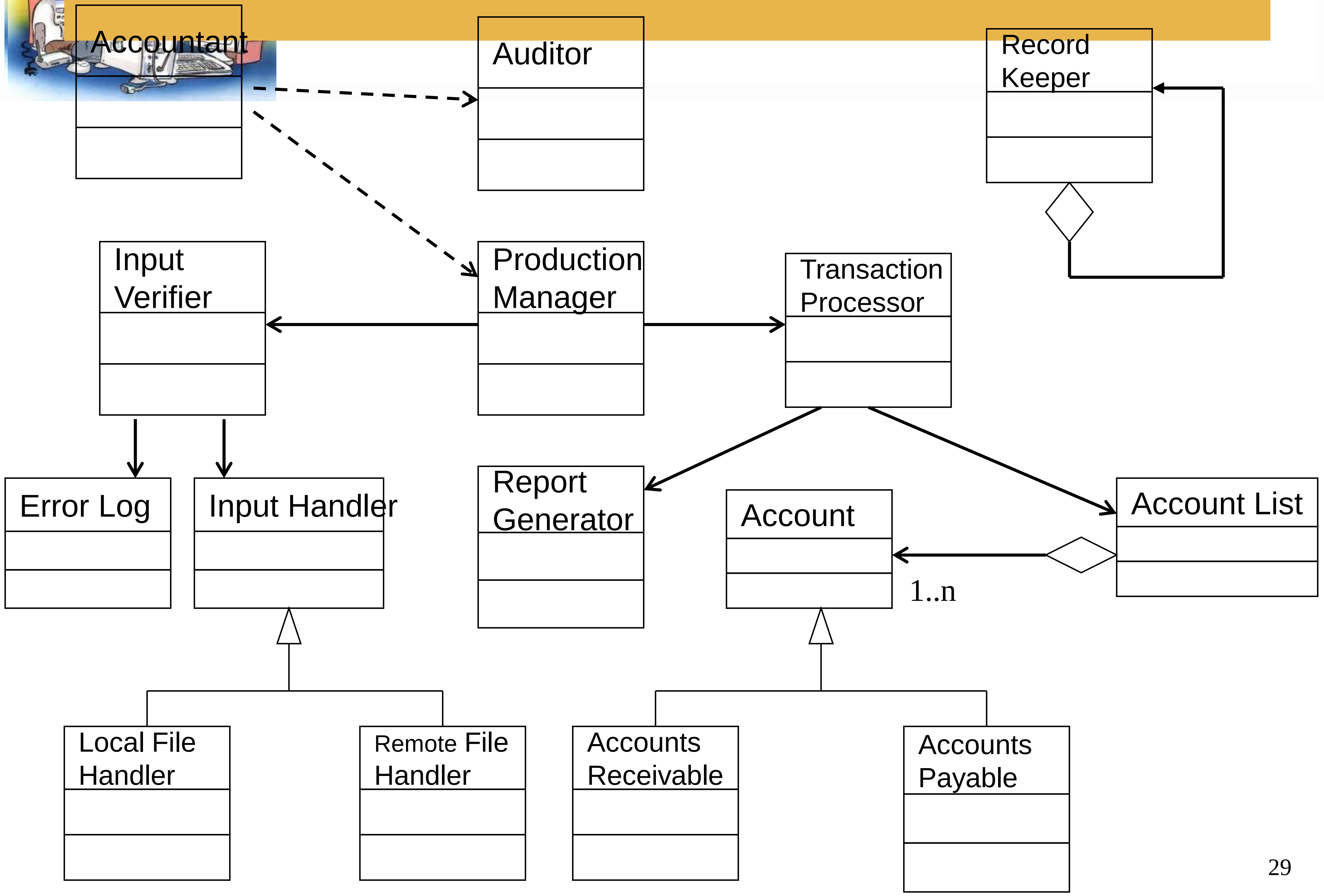
DEFINING OPERATIONS OF A CLASS

- Operations define the behavior of an object
- Four categories of operations
  - Operations that manipulate data in some way to change the state of an object (e.g., add, delete, modify)
  - Operations that perform a computation
  - Operations that inquire about the state of an object
  - Operations that monitor an object for the occurrence of a controlling event
- An operation has knowledge about the state of a class and the nature of its associations
- The action performed by an operation is based on the current values of the attributes of a class
- Using a grammatical parse again, circle the verbs; then select the verbs that relate to the problem domain classes that were previously identified

ASSOCIATION, GENERALIZATION AND DEPENDENCY (REF: FOWLER)

- Association
  - Represented by a solid line between two classes directed from the source class to the target class
  - Used for representing (i.e., pointing to) object types for attributes
  - May also be a part-of relationship (i.e., aggregation), which is represented by a diamond-arrow
- Generalization
  - Portrays inheritance between a super class and a subclass
  - Is represented by a line with a triangle at the target end
- Dependency
  - A dependency exists between two elements if changes to the definition of one element (i.e., the source or supplier) may cause changes to the other element (i.e., the client)
  - Examples
    - One class calls a method of another class
    - One class utilizes another class as a parameter of a method

EXAMPLE CLASS DIAGRAM



ENTITY-RELATIONSHIP MODELING

ER diagramming tools

- There are number of free software ER diagramming tools that can interpret and generate ER models and SQL and do database analysis like MySQL Workbench, and Open ModelSphere (open-source).
- There are also freeware ER tool that can generate database and application layer code (web services) like the RISE Editor.

Limitations

- ER models assume information content can readily be represented in a relational database. They describe only a relational structure for this information.
- They are inadequate for systems in which the information cannot readily be represented in relational form, such as with semi structured data where entities with same class could be having different attributes
- For many systems, the possible changes to the information contained are nontrivial and important enough to warrant explicit specification.

FINITE STATE MACHINES - 1(3)

- FSM models the behaviour of the system in response to external and internal events.
- The system is modeled in terms of number of entities, (a finite number of) states of these entities, and transitions between those states
  - The system is in one state at any given point of time.
    (for visualization a state could be thought of to be the status of the entity in "memory" or which is waiting to be executed).
  - The "initial state" is simply a designated state from where the system starts and there could be a "final state" of the system too. The state it is in at any given time is called the current state.
  - It can change from one state to another when initiated by a triggering event or condition; this is called a transition.

FINITE STATE MACHINES - 2(3)

- These could be documented as state transition tables and depicted as state transition diagrams:
- This represents system's responses to stimuli so are often used for modelling real-time systems.
- A single state chart to represent a FSM for a large system would become unwieldy and is not recommended.
  - A hierarchical decomposition of FSMs where a group of states could be abstracted and looked at as a single entity at one level, and refined at the next level, typically is a way this would be approached.

DATA FLOW DIAGRAMS – 1(3)

- Data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects.
- A DFD shows what kinds of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored.
- DFD does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel
- A DFD is started with a high level context for the whole system, and start "exploding“ this, to produce different levels of DFD, that shows some of the detail of the system being modelled.
- Data flow diagrams can be used to provide the end user with a physical idea on how, the data they input, ultimately has an effect upon the structure of the whole system eg. Order to dispatch, to report.