Technical Methods for Specifying Requirements

Software specifications

- **Definition:** *"Specifications represent a model of how inputs are related to system reactions and outputs"*
- Specification is a **representation** process.
- Requirements are represented in a form that is more understandable
- Specifications will increase the level of details given in the requirements
- Requirements must be specified when the system deals with life-and-death issue

Specification techniques

- Natural language
- Pseudo-code
- Finite State Machines
- Decision Trees
- Activity Diagrams (flowcharts)
- Entity-relationship diagrams
- Dataflow diagrams
- OO Analysis
- Z and VDM

Specification techniques classification

- Informal
  - Natural language
  - Pseudo-code
- Semi-formal
  - Entity-relationship diagrams
  - Dataflow diagrams
  - OO Analysis
- Formal
  - Z, VDM

Natural Language (English, Arabic, etc)

- In many software projects, the specification document is written in a human language (such as English, Arabic, etc.)
- ***What are the Problems of natural language***
  - Not precise (words have many meanings)
  - Ambiguous (I saw a boy with a telescope)
  - Hard to be formally checked for completeness/consistency.

Pseudo-code

- A "quasi" programming language, consisting of
  - Imperative sentences with a single verb and a single object;
  - A limited set of "action-oriented" verbs from which the sentences must be constructed;
  - Decisions represented with a formal IF-ELSE-ENDIF structure;
  - Iterative activities represented with DO-WHILE or FOR-NEXT structures.

Pseudo-code

- Example of pseudo-code
  - Calculating the deferred-service revenue earned for any month is:

```
Set SUM(x)=0
For each customer X
    If X has paid support AND
        ( current month) >= (2 months after ship date) AND
        (current month) <= (14 months after ship date)
    THEN  SUM(X) = SUM(X) + ((amount customer paid)/12)
```

- More examples:
  - http://www.unf.edu/~broggio/cop3530/3530pseu.htm

Pseudo-code

- Advantages:
  - Understandable by a non-programming person.
  - Reduce the ambiguity of requirements.
  - The writer does need some programming skills.
- Problems:
  - Not good for a high level abstraction of the project.
  - The requirements engineer is prone to fall into programming details.

Finite state machines

- Represent a system as a "hypothetical machine" that can be in only one state at a given time".
- The output and next state can be determined by the current state and the event that caused the transition.
- Two popular notations for finite state machines:
  - State-transition diagram
  - State-transition matrix
- Examples:
  - http://www.agilemodeling.com/artifacts/stateMachineDiagram.htm

Light Box Example

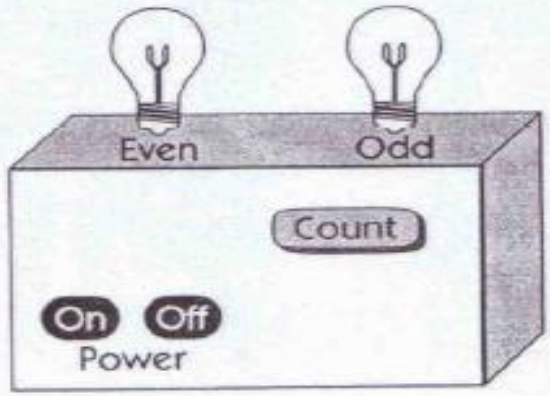
After On pushed but before Off pushed, system is termed "powered on."

After Off pushed but before On pushed, system is termed "powered off," and no lights shall be lit.

Since most recent On press, if Count has been pressed an odd number of times, Odd shall be lit.

Since most recent On press, if Count has been pressed an even number of times, Even shall be lit.

If either light burns out, the other light shall flash every 1 second.



Features

- Microprocessor controlled
- Keeps track of whether count button has been pressed even or odd number of times
- Burned-out-bulb detector flashes remaining bulb

Figure 26–1 Light box

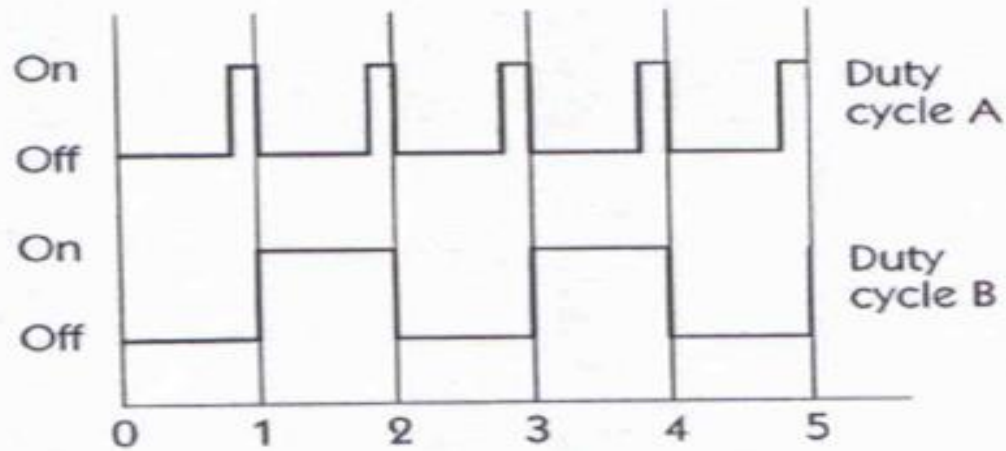


Figure 26–2 Possible lamp duty cycles

State transition diagram for the light box (chapter 26)

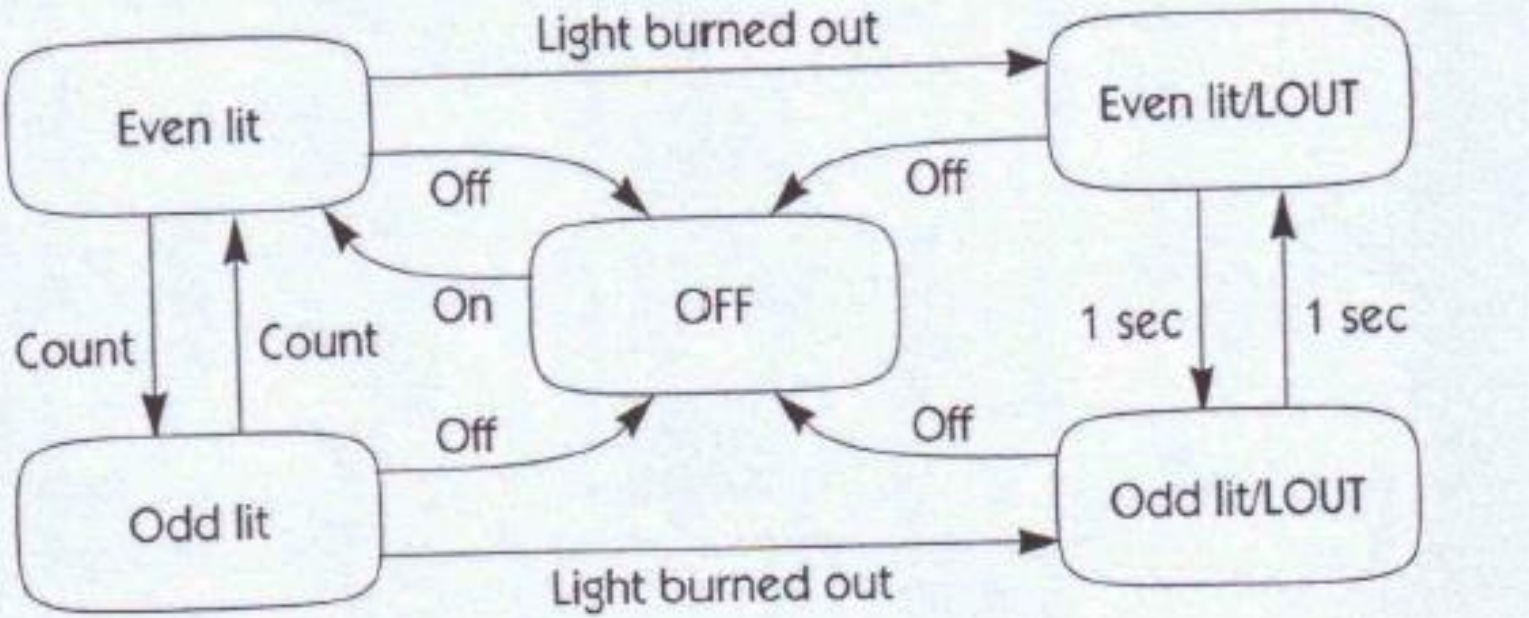


Figure 28–2 State transition diagram

State-transition representation

Table 28–1 State-transition matrix for on/off counting device

State	Event					Output
	On press	Off press	Count press	Bulb burns out	Every second	
Off	Even lit	—	—	—	—	Both off
Even lit	—	Off	Odd lit	LO/Even lit	—	Even lit
Odd lit	—	Off	Even lit	LO/Odd lit	—	Odd lit
Light out/Even lit	—	Off	—	Off	LO/Odd lit	Even lit
Light out/Odd lit	—	Off	—	Off	LO/Even lit	Odd lit

Finite state machines (cont.)

- Suitable for describing the interaction between the user and the system.
- Not good for presenting a system's behavior with several inputs.

Decision trees/tables

- Tree-like diagrams or tables used to describe complex logic relationships within a requirement.
- Suitable for a requirement that deals with a combination of inputs; and different combinations of the inputs lead to different outputs.

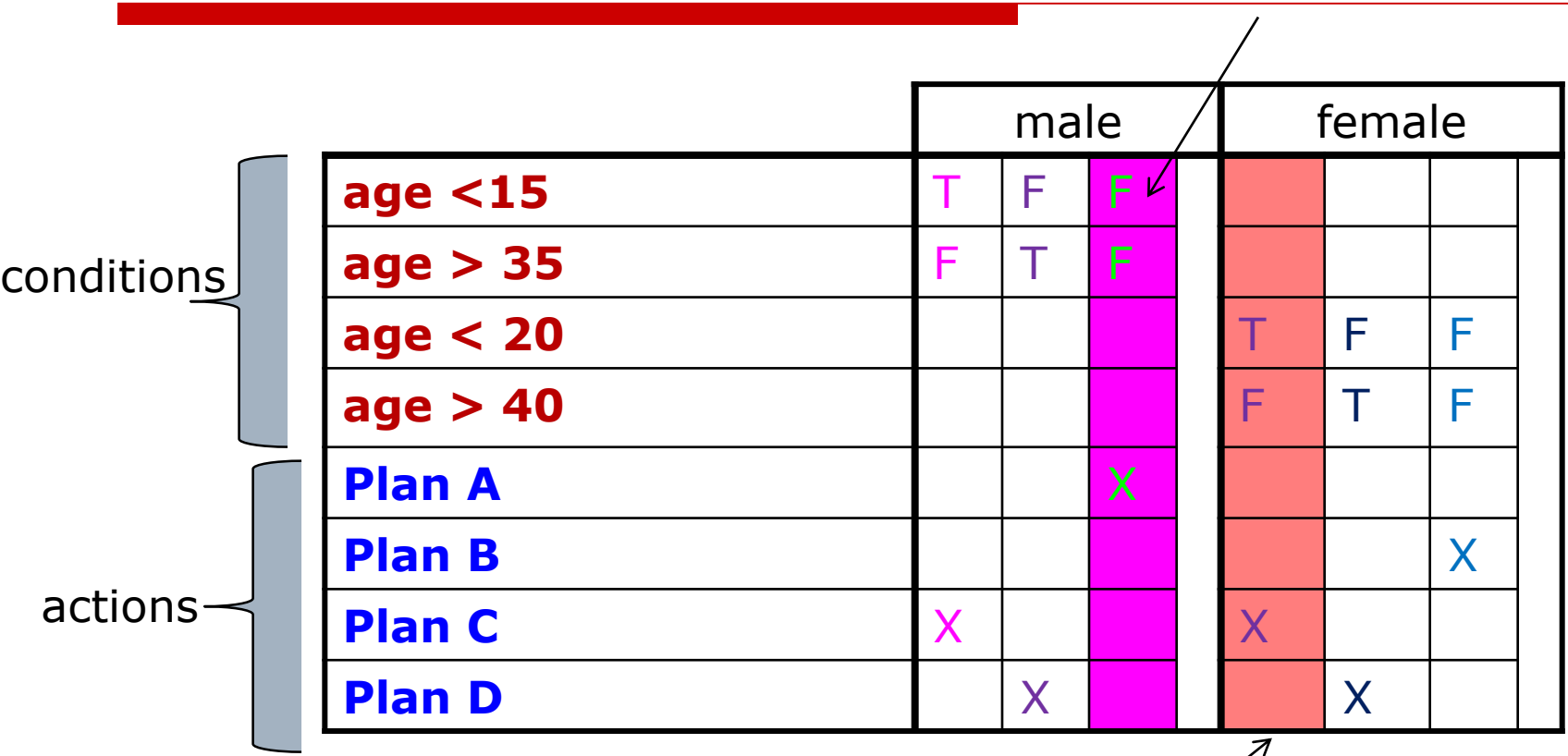
Decision trees/tables

- Develop a decision table for the following description of the allowance policy at the Emirates Computer Company:
- *Males between 15 and 35 years old are applied allowance plan A. Females between 20 and 40 years old are applied allowance plan B. All males under 15 and females under 20 are applied allowance plan C. All males over 35 and all females over 40 years old are applied allowance plan D.*

Decision trees/tables

if male and age not less than 15 and age is not greater than 35 then apply plan A.

15 35



	male			female		
age <15	T	F	F			
age > 35	F	T	F			
age < 20				T	F	F
age > 40				F	T	F
Plan A			X			
Plan B						X
Plan C	X			X		
Plan D		X			X	

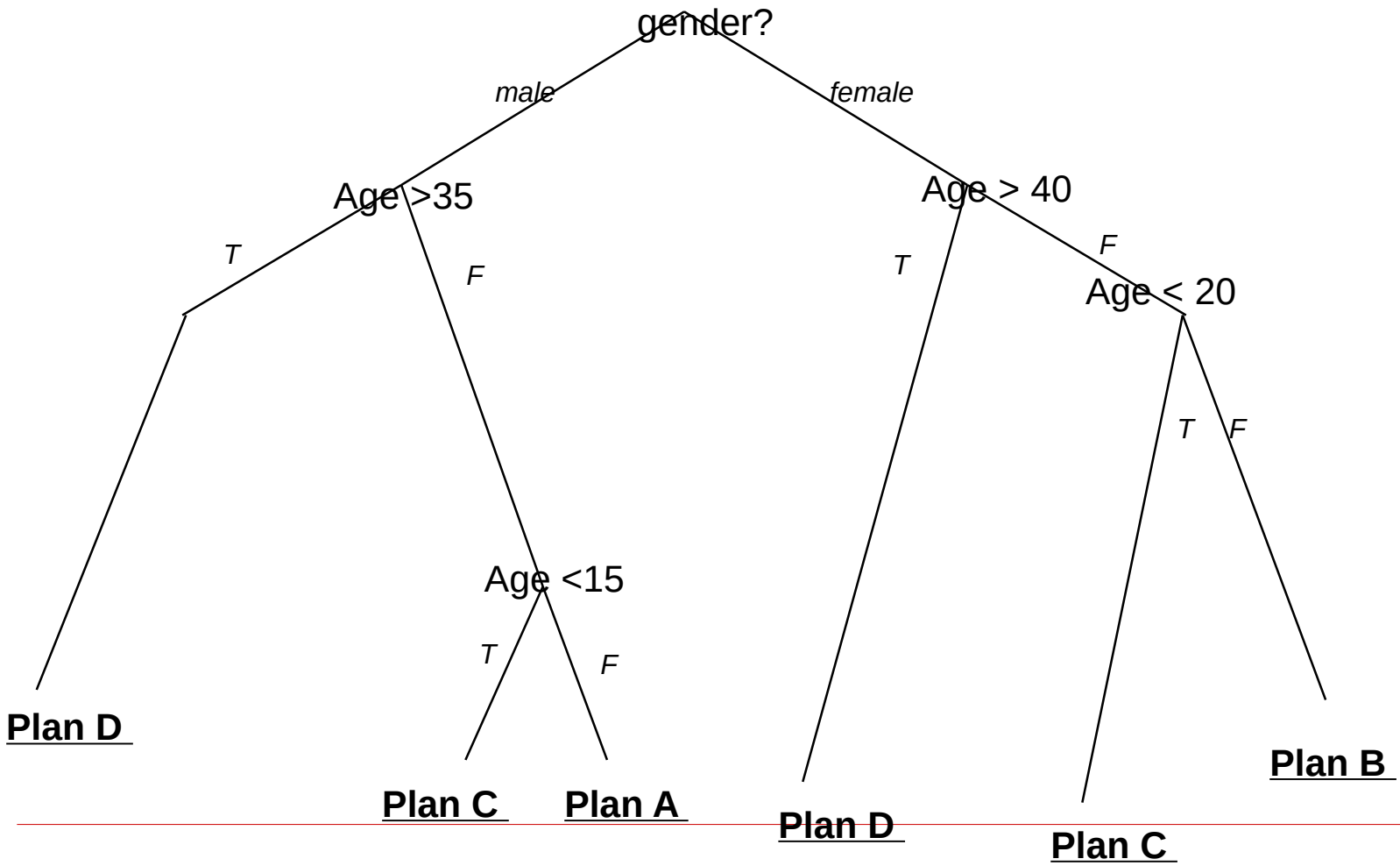
conditions

actions

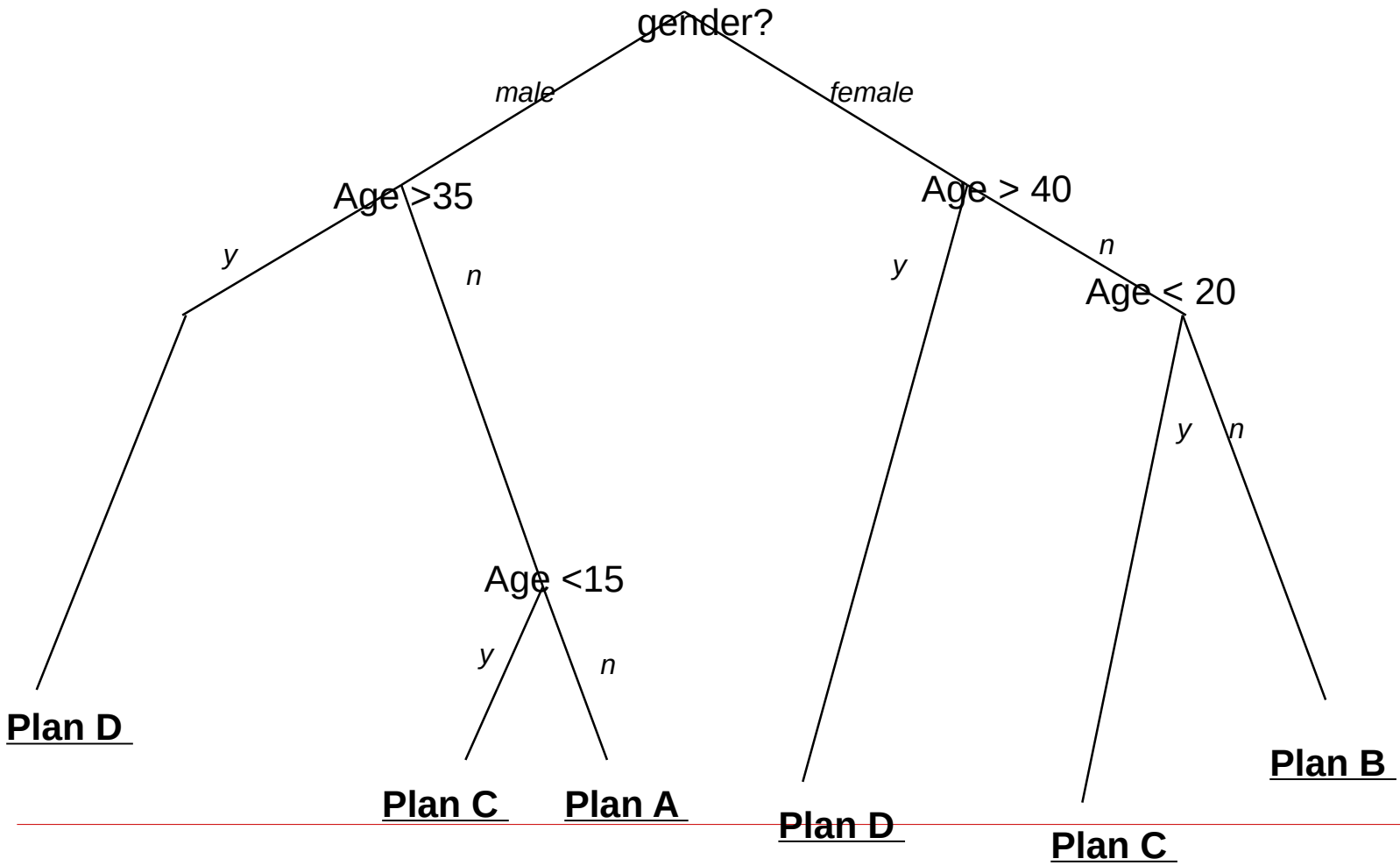
T: true
F: false
X: N/A

If female and age < 20 then apply plan C

Graphical Decision Trees



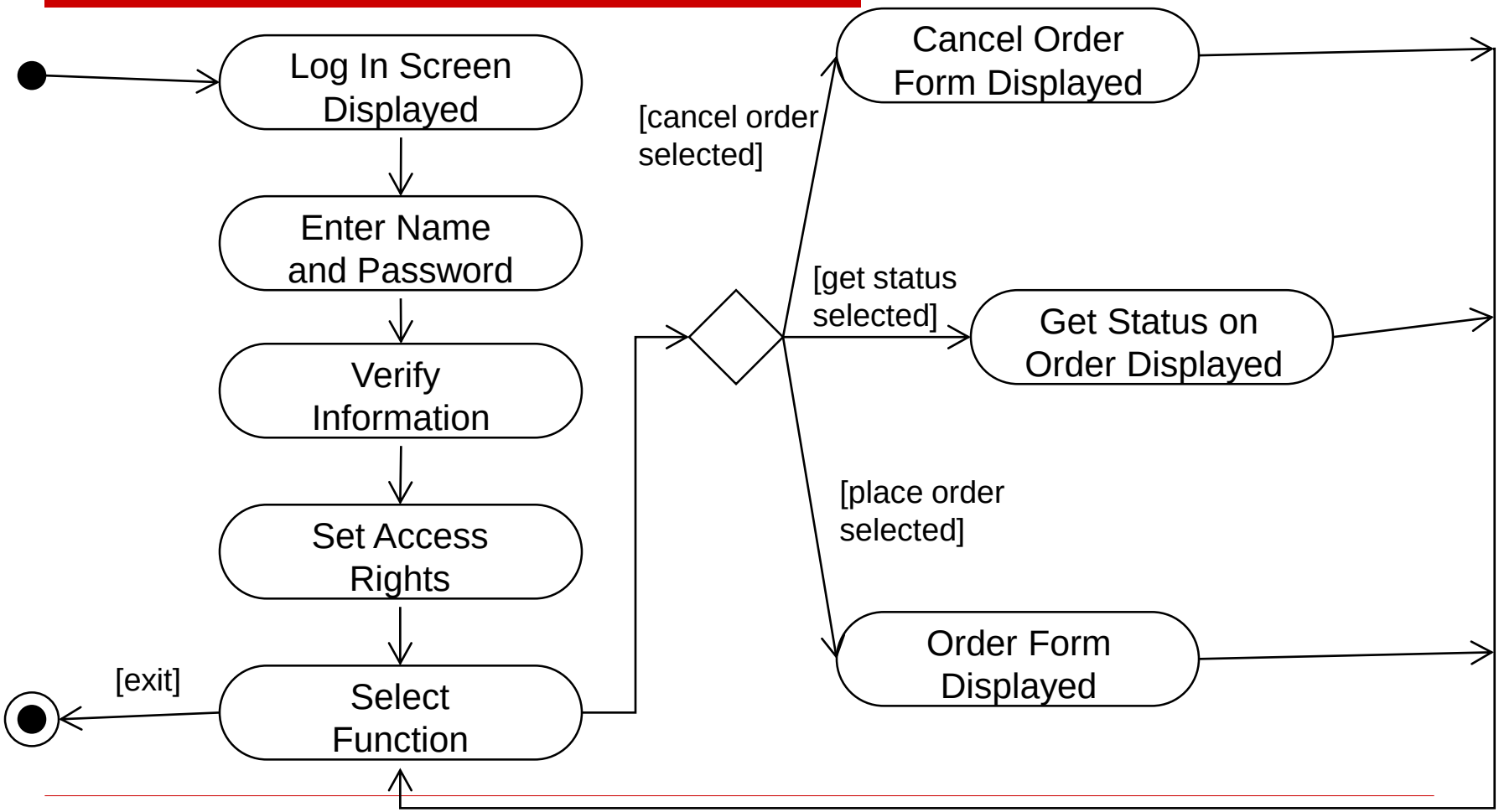
Graphical Decision Trees



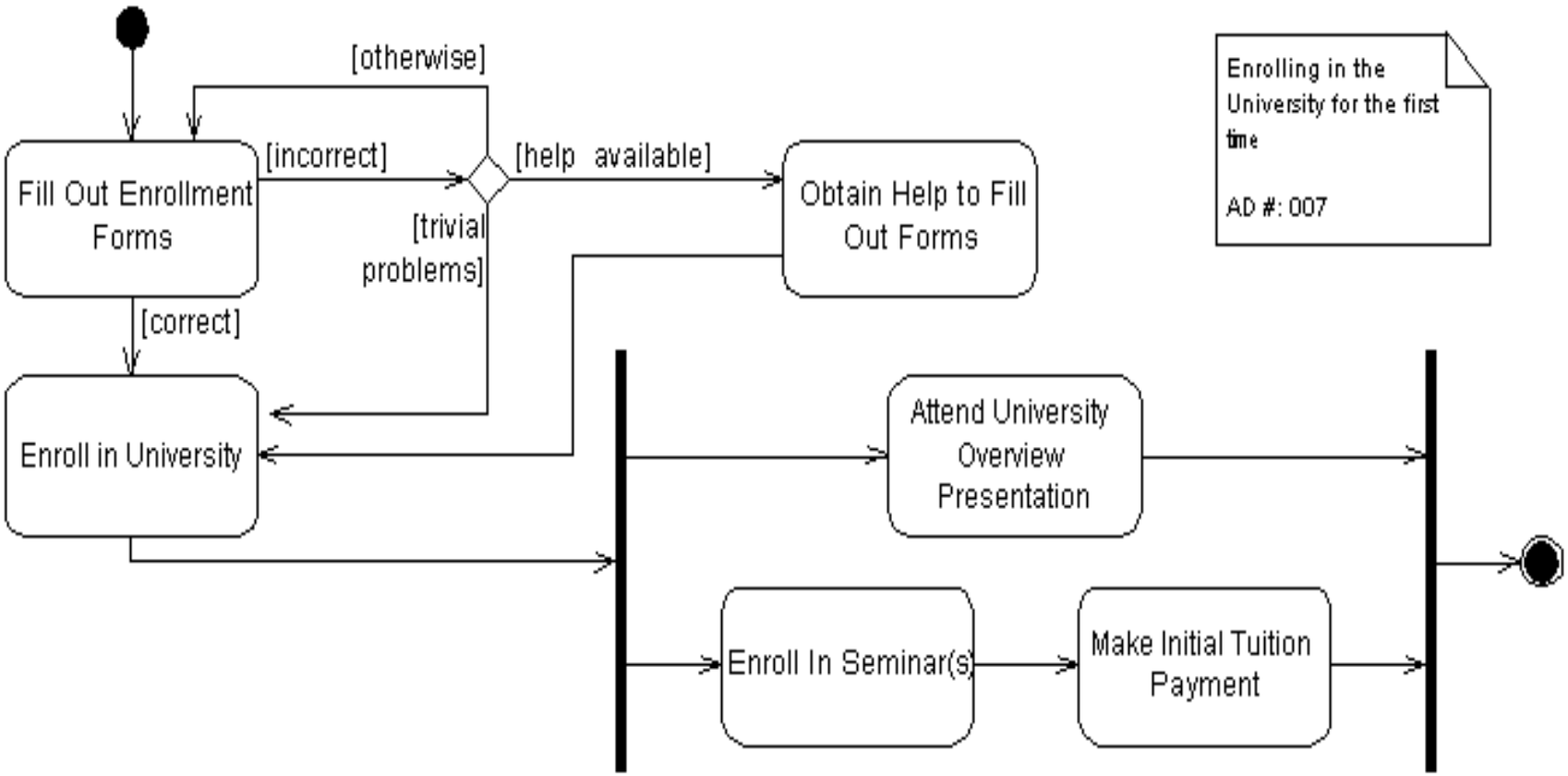
Activity Diagrams

- A state transition diagram where all states are action states
- Very easy to write and understand
- Enables a representation of branching, repetition and process forking
- Used to naturally represent flows of events in scenarios

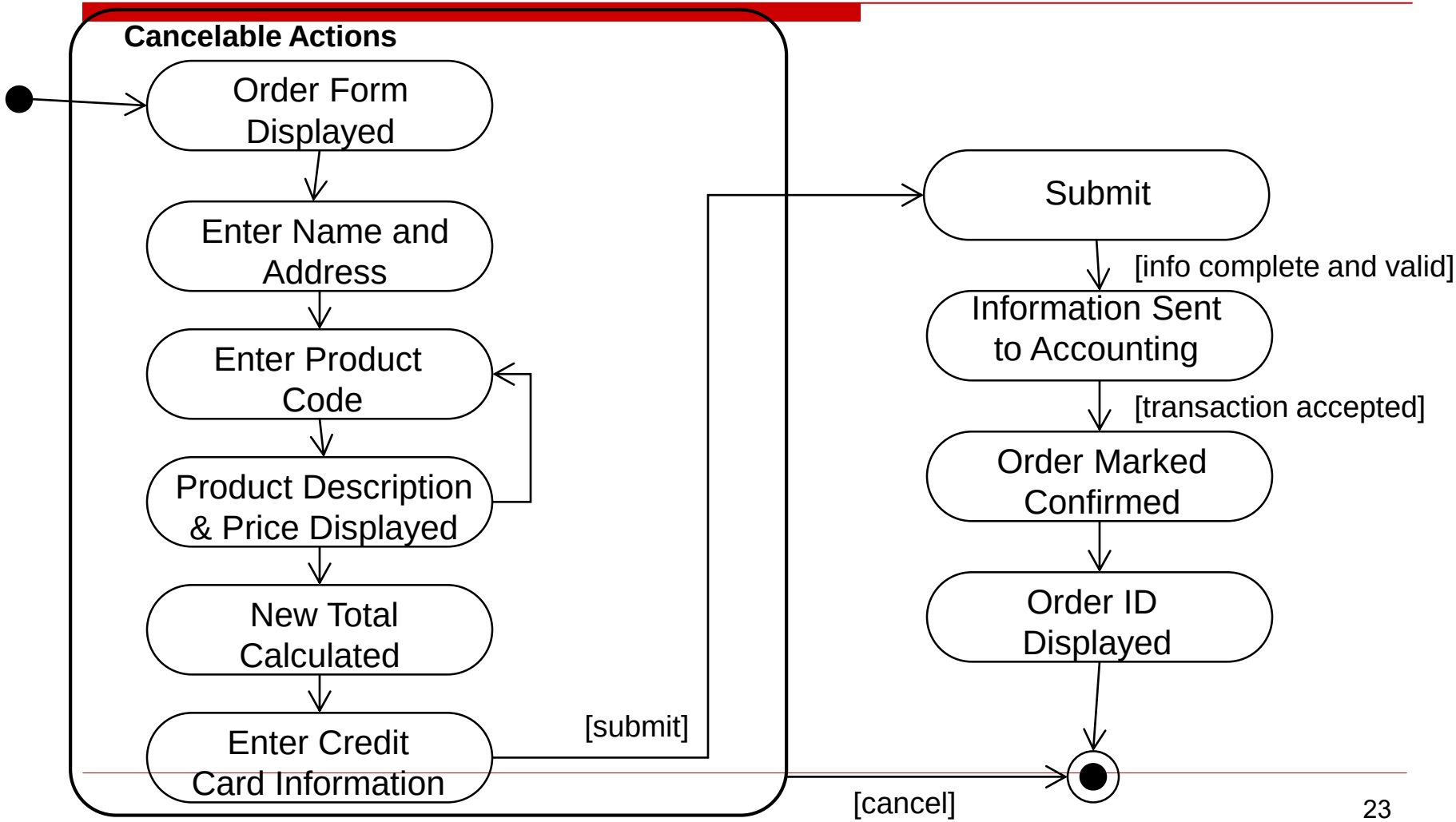
Activity Diagrams



Activity diagram (cont.)



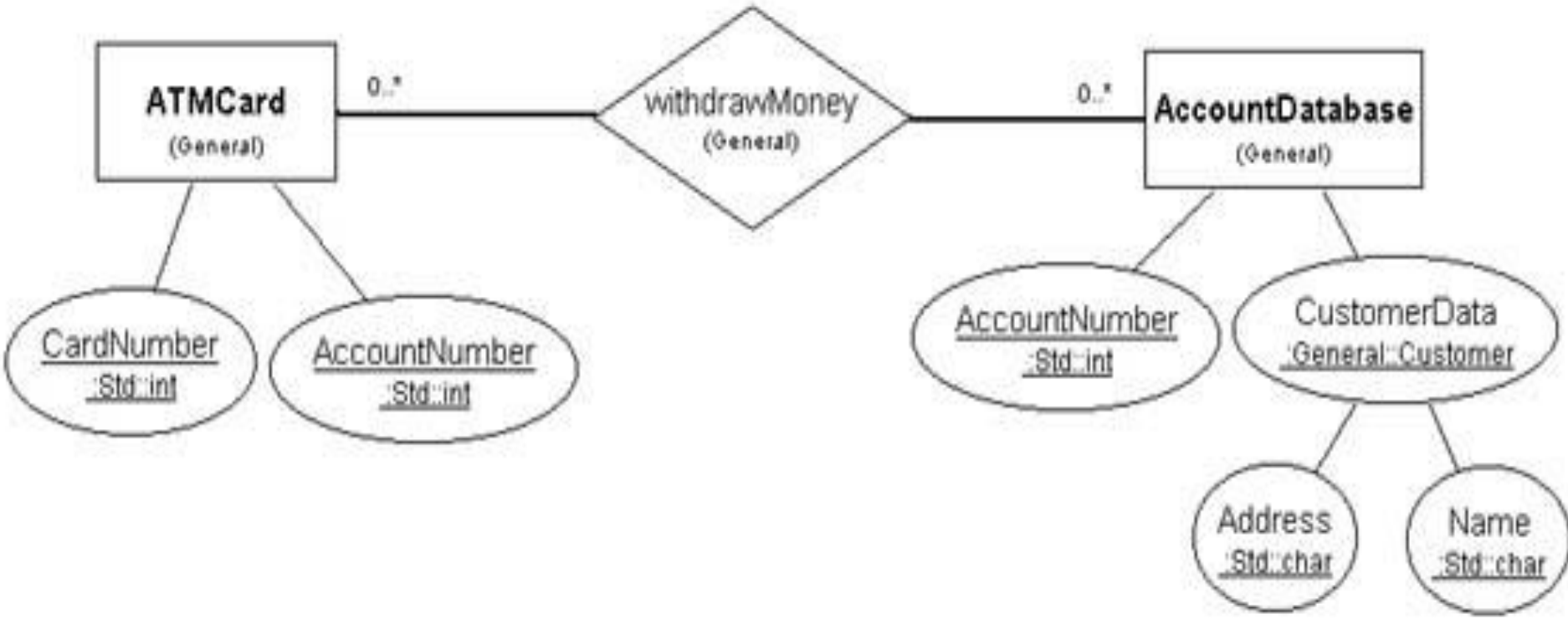
Activity Diagrams



Entity-relationship models

- Use ER diagram to represent the structure and relationships among *data* within the system.
- It provides a high-level "architectural" view of the system data.
- It focuses on the external behaviors of the system.

Entity-relationship models



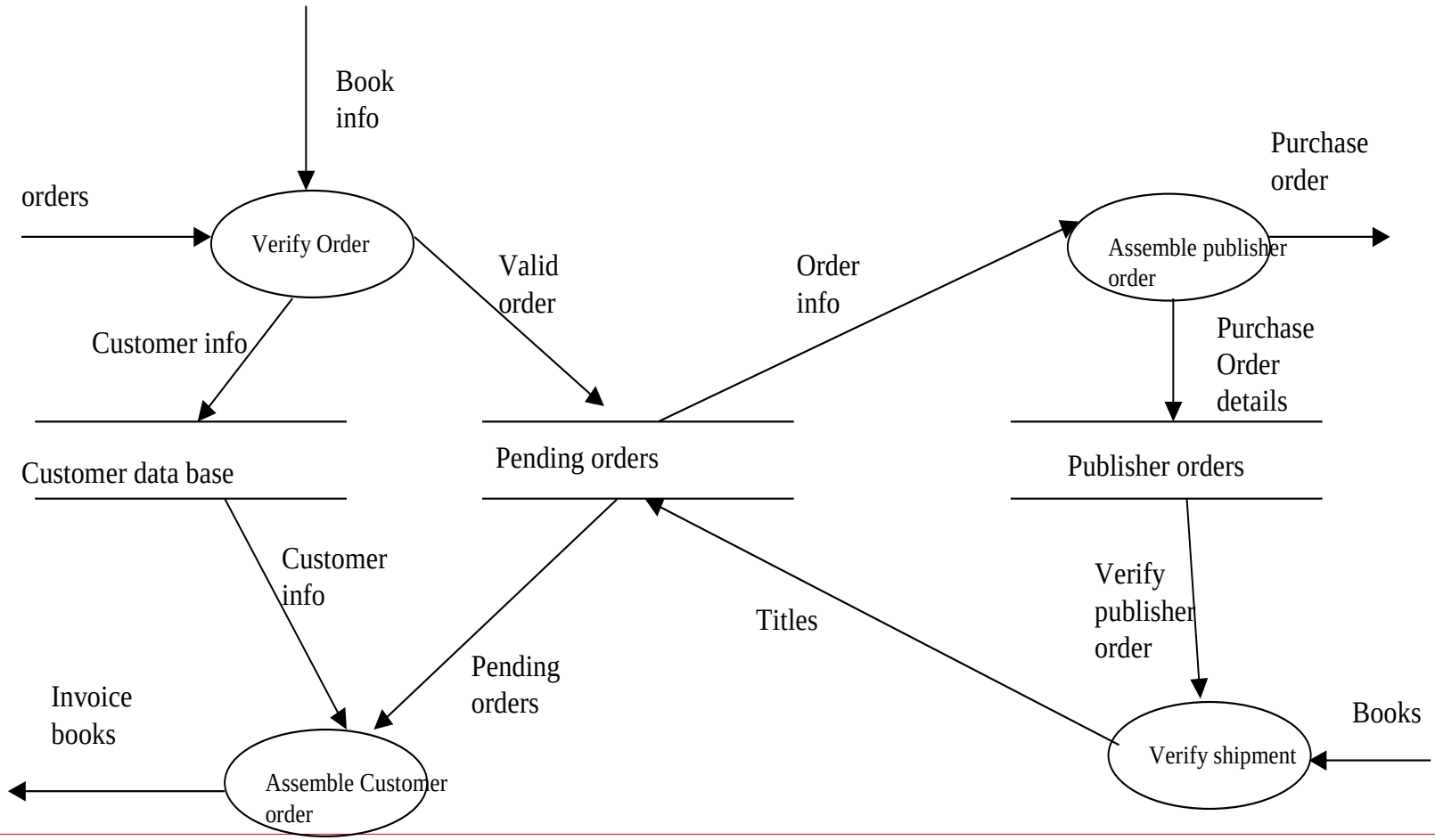
Entity-relationship models

- Problems
  - Might be difficult for non-technical readers to understand.
  - Different notations exist
  - Easy to fall in the trap of defining database structure, which is a design activity, not a specification activity.

Data flow diagrams

- Visual presentation of the structure and the organization of the low-level requirements and the input/output relationship among them.
- Representation of the flow of data in the system
  - Data stores
  - Data filters
  - Relationships

Data flow diagrams Example Ordering book



Data flow diagrams

- Could be further elaborated with lower-level diagrams to show the details.
- Could be the basis for communication between non-technical users and technical developers.
- Might be difficult for a non-technical reader to understand.

Object-oriented analysis

- Describe the structure and relationships among *entities* within the system.
- UCD Approach
  - Use case diagrams
  - Activity diagrams
  - Sequence diagrams
  - Collaboration diagrams
  - Class diagrams
- Almost all other kinds of diagrams and techniques can be be “objectified”.

Object-oriented analysis

- Relatively new, but spreading rapidly along with the popularity of OO and the adoption of the UML.
- More appropriate in the implementation models used to realize the functionalities of the system.

Object-oriented analysis