Chapter 3 — System Models

March 16, 2009

Introduction

- Graphical models aid in requirements and development

Introduction

- Graphical models aid in requirements and development
- Different perspectives are possible:
  - *external*: context or environment modeling
  - *behavioral*
  - *structural*: system or data architecture

Introduction

- Graphical models aid in requirements and development
- Different perspectives are possible:
  - *external*: context or environment modeling
  - *behavioral*
  - *structural*: system or data architecture
- Leave out *details*
  - *representation*, *abstraction*

Introduction

- Graphical models aid in requirements and development
- Different perspectives are possible:
  - *external*: context or environment modeling
  - *behavioral*
  - *structural*: system or data architecture
- Leave out *details*
  - *representation*, *abstraction*
- types of models:
  - *data-flow model*
  - *architectural model*
  - *classification model*: object model
  - *stimulus-response model*: state-transition diagrams

Introduction

- Graphical models aid in requirements and development
- Different perspectives are possible:
  - *external*: context or environment modeling
  - *behavioral*
  - *structural*: system or data architecture
- Leave out *details*
  - *representation*, *abstraction*
- types of models:
  - *data-flow model*
  - *architectural model*
  - *classification model*: object model
  - *stimulus-response model*: state-transition diagrams
- use **UML**

Outline

Context Models

- decide early on system boundaries

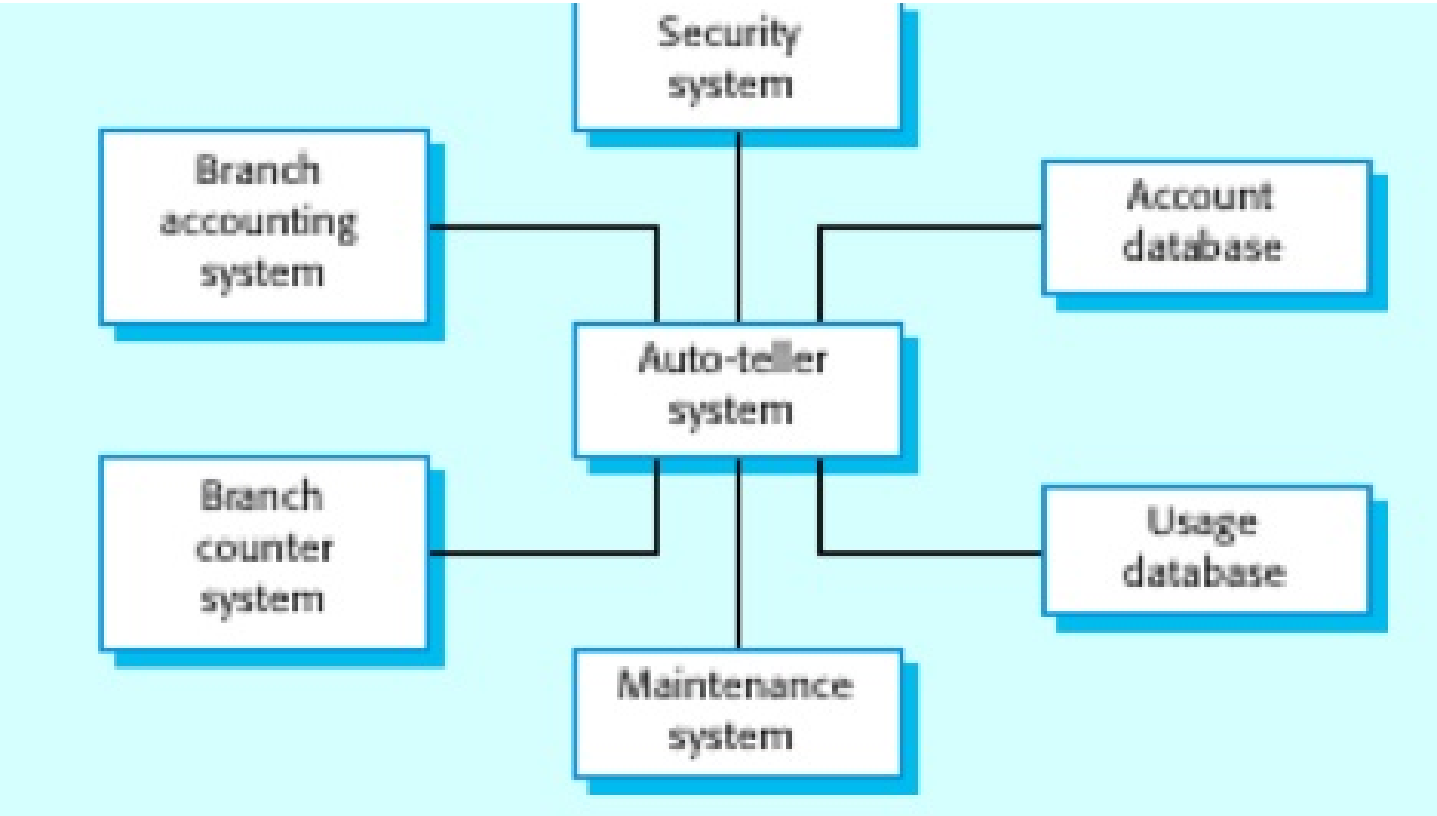
Context Models

- decide early on system boundaries
- easy in some cases; flexible in other
  - *example*: replace existing manual system

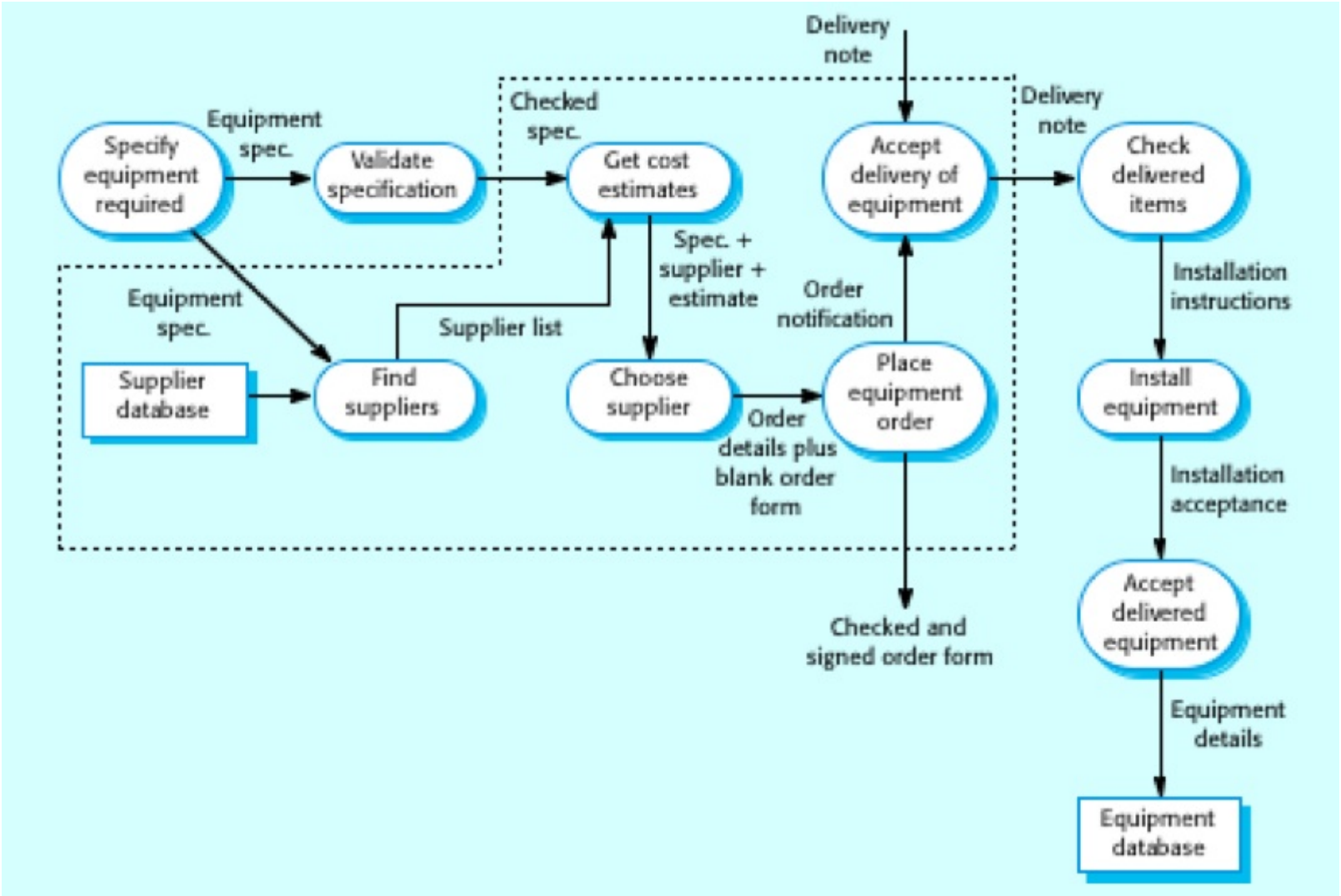
Context Models

- decide early on system boundaries
- easy in some cases; flexible in other
  - *example*: replace existing manual system
- next develop context model
  - show context and dependencies on environment
  - detailed relationships are note shown
    - use *process* or *data-flow* models

Context Model for an ATM System



Process Model for Equipment Producrement



Outline

Introductiom

- Describe overall behavior of system
- Two types:
  - data-flow models
  - state machine models

Introductiom

- Describe overall behavior of system
- Two types:
  - data-flow models
  - state machine models
- Businesses rely mostly on data or information
  - use data-flow models for modeling behavior

Introductiom

- Describe overall behavior of system
- Two types:
  - data-flow models
  - state machine models
- Businesses rely mostly on data or information
  - use data-flow models for modeling behavior
- Real-time systems are event driven with minimal data processing
  - use state machines for preseting behavior

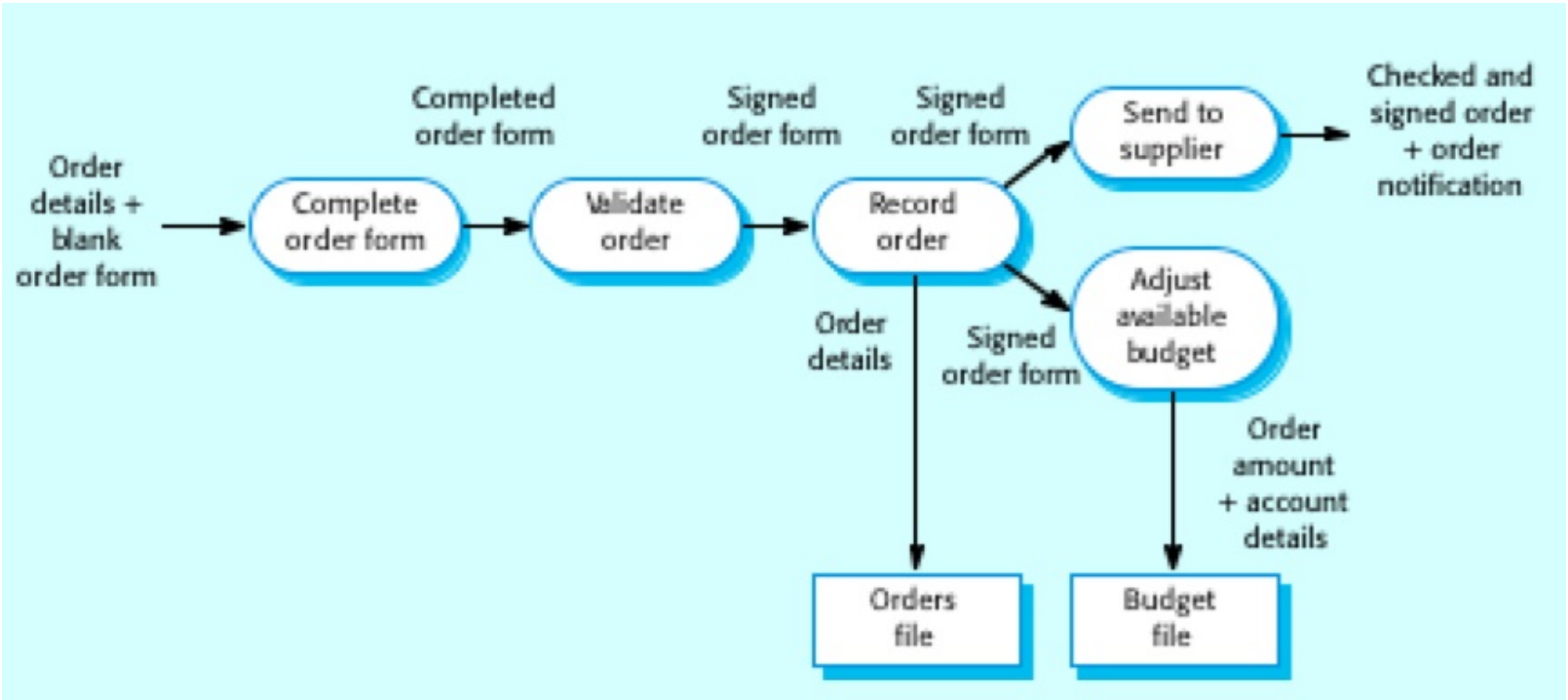
Data-Flow Models

- Shows how the system processes data
  - e.g.: filter duplicate records from a customer database

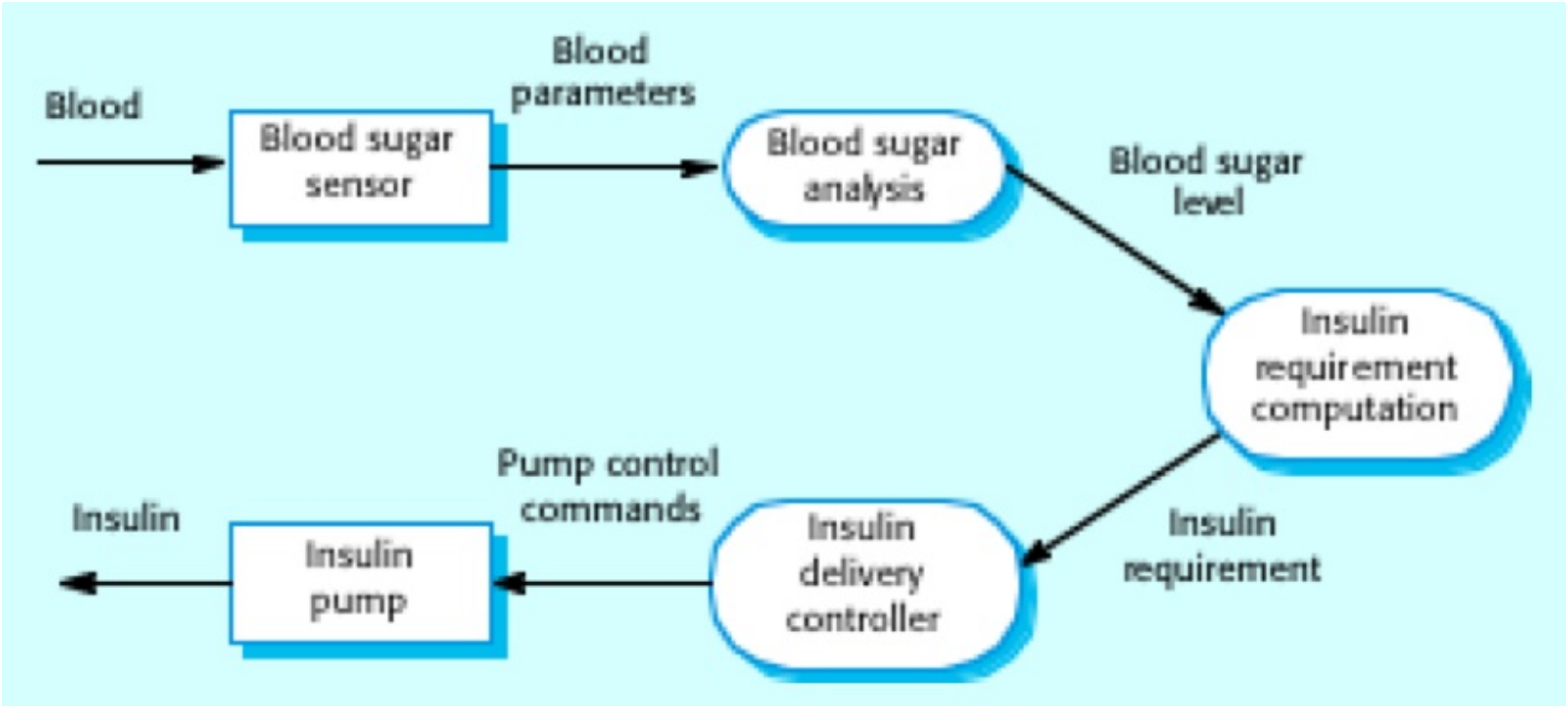
Data-Flow Models

- Shows how the system processes data
  - e.g.: filter duplicate records from a customer database
- Convention:
  - rounded rectangles: *data processing*
  - square rectangles: *data stores*
  - labelled arrows: *data flow through system*

DFD for Order Processing



DFD for Insulin Pump



Use and Advantages of DFD

- show a functional perspective
  - each transformation represents a single function or process
  - show end-to-end processing

Use and Advantages of DFD

- show a functional perspective
  - each transformation represents a single function or process
  - show end-to-end processing
- Simple and intuitive

Use and Advantages of DFD

- show a functional perspective
  - each transformation represents a single function or process
  - show end-to-end processing
- Simple and intuitive
- Understandable by system users

Use and Advantages of DFD

- show a functional perspective
  - each transformation represents a single function or process
  - show end-to-end processing
- Simple and intuitive
- Understandable by system users
- “Top-down approach”, in principle

State Machine Models

- System's responses to internal or external events

State Machine Models

- System's responses to internal or external events
- Shows
  - system states (one at a time)
  - events that cause transformation between states

State Machine Model of a Microwave Oven

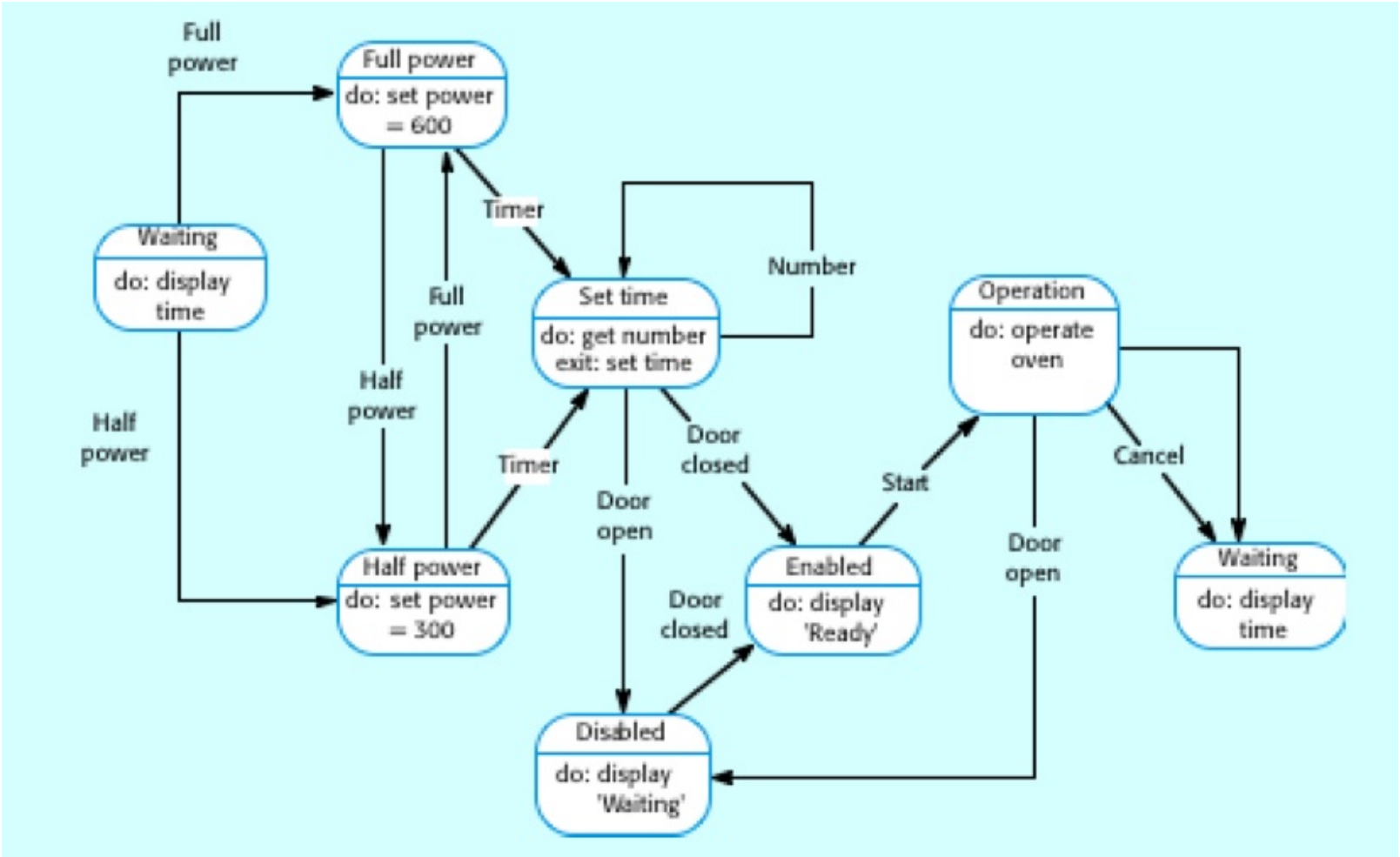


Figure: UML-based state machine model

Microwave Oven States

State	Description
Waiting	The oven is waiting for input. The display shows the current time.
Half power	The oven power is set to 300 watts. The display shows 'Half power'.
Full power	The oven power is set to 600 watts. The display shows 'Full power'.
Set time	The cooking time is s et to the user's input value. The display shows the cooking time selected and is updated as the time is set.
Disabled	Oven operation is disabled for safety. Interior oven light is on. Display shows 'Not ready'.
Enabled	Oven operation is enabled. Interior oven light is off. Display shows 'Ready to cook'.
Operation	Oven in operation. Interior oven light is on. Display shows the timer countdown. On completion of cooking, the buzzer is sounded for 5 s econds. Oven light is on. Display shows 'Cooking complete' while buzzer is sounding.

Microwave Oven Stimuli

Stimulus	Description
Half power	The user has pressed the half power button
Full power	The user has pressed the full power button
Timer	The user has pressed one of the timer buttons
Number	The user has pressed a numeric key
Door open	The oven door switch is not closed
Door closed	The oven door switch is closed
Start	The user has pressed the start button
Cancel	The user has pressed the cancel button

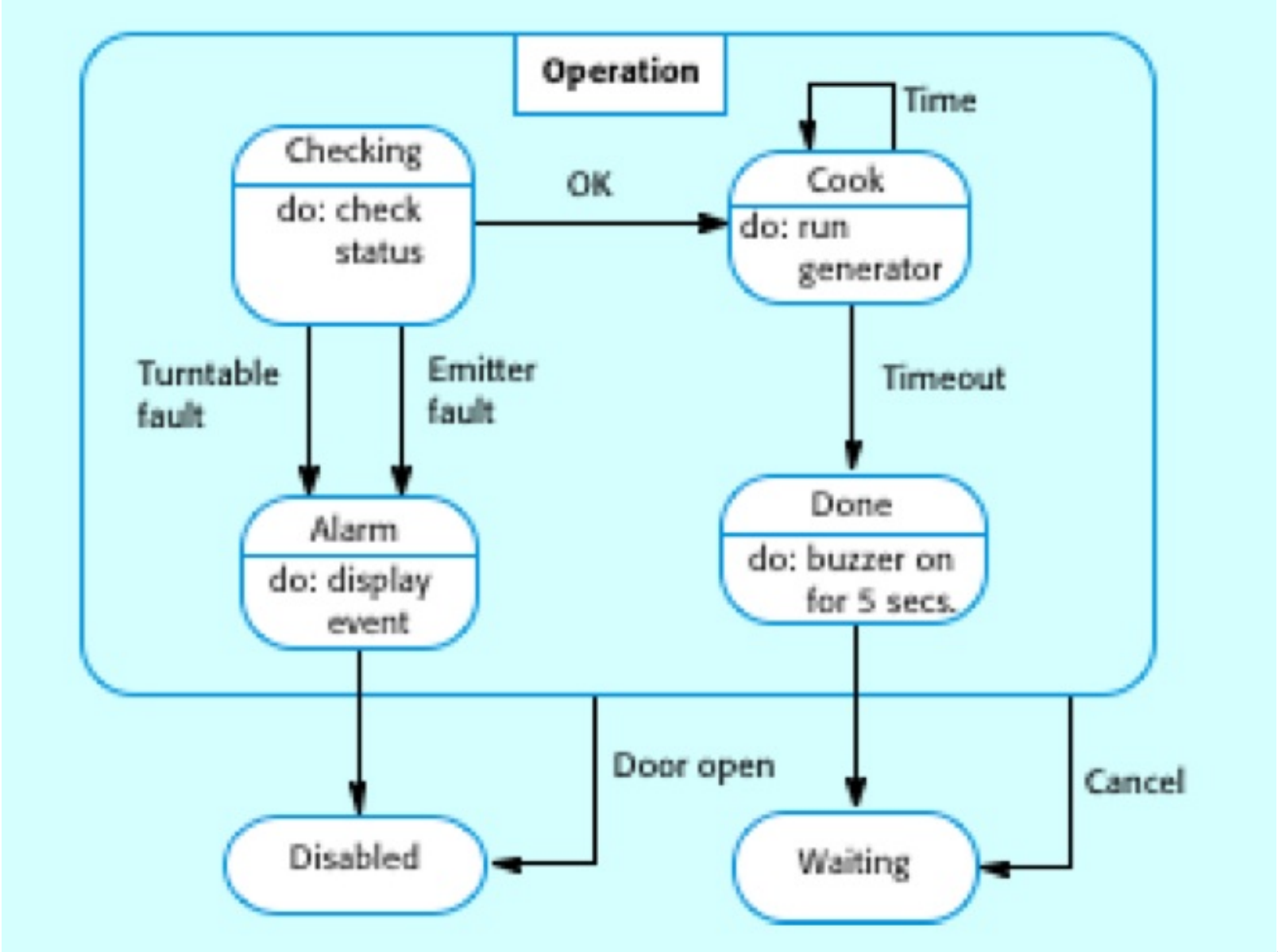
Some Issues with State Machine Models

- Can grow exponentially with size of system

Some Issues with State Machine Models

- Can grow exponentially with size of system
- One solution: use *super* and *sub* states

Microwave Operation Super-state



Outline

Data

- Most software systems focus on data

Data

- Most software systems focus on data
  - independent of system
  - created for the system

Data

- Most software systems focus on data
  - independent of system
  - created for the system
- *Semantic data models*: logical form of the data

Data

- Most software systems focus on data
  - independent of system
  - created for the system
- *Semantic data models*: logical form of the data

Semantic Data Models

- Most common:

Semantic Data Models

- Most common:
  - *entity-relation-attribute* model

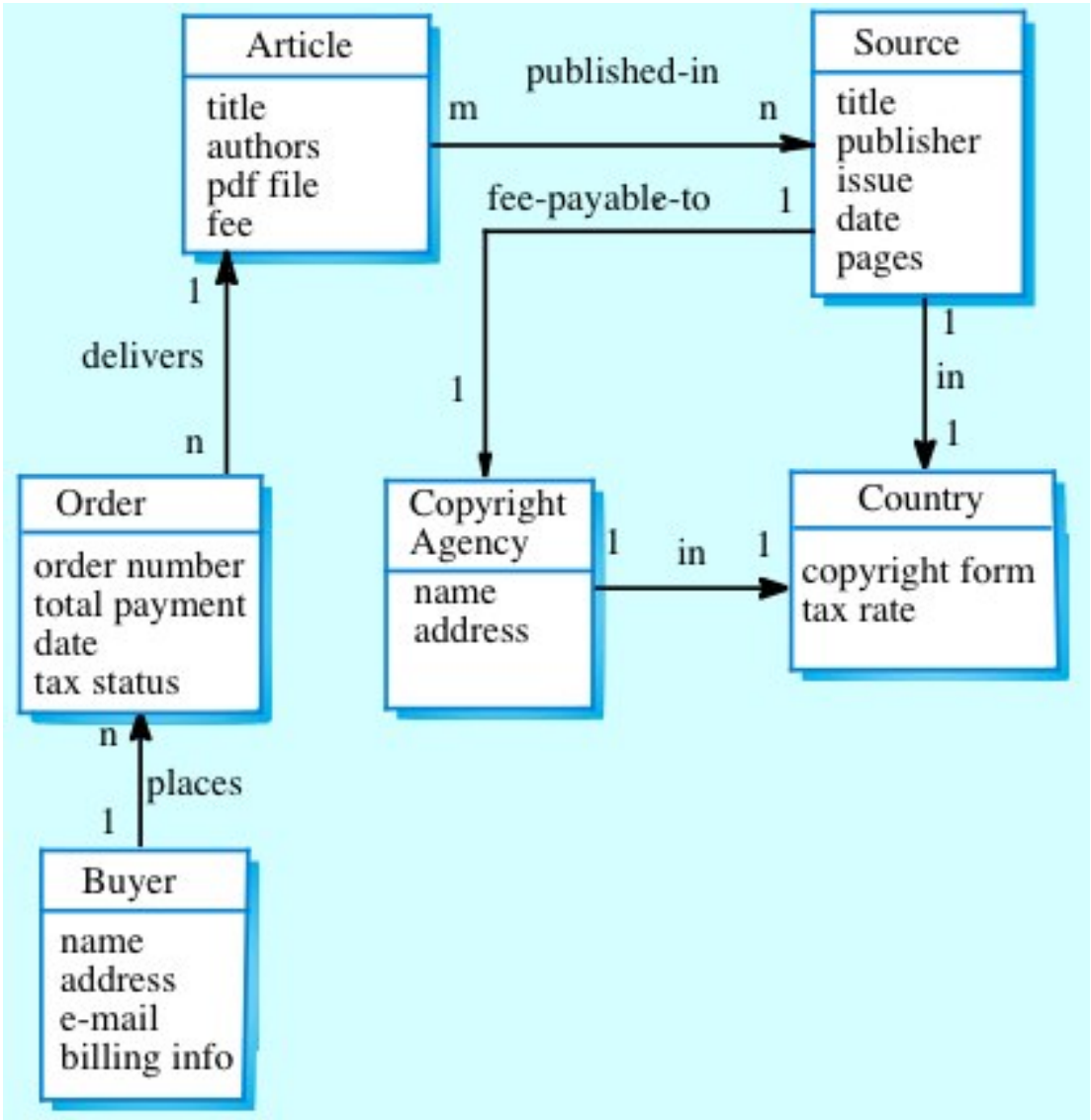
Semantic Data Models

- Most common:
  - *entity-relation-attribute* model
- Usually in third normal form

Semantic Data Models

- Most common:
  - *entity-relation-attribute* model
- Usually in third normal form
- Easy to tranform to OO design

Sample ERA mdoel



Semantic Data Models

- Since graphical, lacks details

Semantic Data Models

- Since graphical, lacks details
- Use *data dictionary*

Semantic Data Models

- Since graphical, lacks details
- Use *data dictionary*
  - keep detailed description of all names

Semantic Data Models

- Since graphical, lacks details
- Use *data dictionary*
  - keep detailed description of all names
  - *name management*

Semantic Data Models

- Since graphical, lacks details
- Use *data dictionary*
  - keep detailed description of all names
  - *name management*
  - *organization information storage*

Data Dictionary

Name	Description	Type	Date
Article	Details of the published article that may be ordered by people using LIBSYS.	Entity	30.12.2002
authors	The names of the authors of the article who may be due a share of the fee.	Attribute	30.12.2002
Buyer	The person or organisation that orders a co py of the article.	Entity	30.12.2002
fee-payable-to	A 1:1 relationship between Article and the Copyright Agency who should be paid the copyright fee.	Relation	29.12.2002
Address (Buyer)	The address of the buyer. This is used to any paper billing information that is required.	Attribute	31.12.2002

Outline

Object-Oriented Modeling

- Identify classes of objects used in the domain

Object-Oriented Modeling

- Identify classes of objects used in the domain
- Most commonly modeling technique today

Object-Oriented Modeling

- Identify classes of objects used in the domain
- Most commonly modeling technique today
  - particularly for **interactive systems**
  - *as compared to what?*

Object-Oriented Modeling

- Identify classes of objects used in the domain
- Most commonly modeling technique today
  - particularly for **interactive systems**
  - *as compared to what?*
- Involves:

Object-Oriented Modeling

- Identify classes of objects used in the domain
- Most commonly modeling technique today
  - particularly for **interactive systems**
  - *as compared to what?*
- Involves:
  - system requirements in object model

Object-Oriented Modeling

- Identify classes of objects used in the domain
- Most commonly modeling technique today
  - particularly for **interactive systems**
  - *as compared to what?*
- Involves:
  - system requirements in object model
  - design using objects

Object-Oriented Modeling

- Identify classes of objects used in the domain
- Most commonly modeling technique today
  - particularly for **interactive systems**
  - *as compared to what?*
- Involves:
  - system requirements in object model
  - design using objects
  - develop in an OO language (Java, C++, Smalltalk, ...)

Object-Oriented Modeling

- Identify classes of objects used in the domain
- Most commonly modeling technique today
  - particularly for **interactive systems**
  - *as compared to what?*
- Involves:
  - system requirements in object model
  - design using objects
  - develop in an OO language (Java, C++, Smalltalk, ...)
- We will learn more in OO *design*

Object-Oriented Modeling

- Identify classes of objects used in the domain
- Most commonly modeling technique today
  - particularly for **interactive systems**
  - *as compared to what?*
- Involves:
  - system requirements in object model
  - design using objects
  - develop in an OO language (Java, C++, Smalltalk, ...)
- We will learn more in OO *design*
- Represent: *data* and its *processing*

Object Model

- **Class**

Object Model

- **Class**
  - abstraction over set of objects

Object Model

- **Class**
  - abstraction over set of objects
  - identifies common attributes and services

Object Model

- **Class**
  - abstraction over set of objects
  - identifies common attributes and services
- **Object**

Object Model

- **Class**
  - abstraction over set of objects
  - identifies common attributes and services
- **Object**
  - executable entity instantiated from a class

Object Model

- **Class**
  - abstraction over set of objects
  - identifies common attributes and services
- **Object**
  - executable entity instantiated from a class
- models developed during requirements phase focus on:
  - classes
  - relationships
  - *not* details of objects

Object Model

- **Class**
  - abstraction over set of objects
  - identifies common attributes and services
- **Object**
  - executable entity instantiated from a class
- models developed during requirements phase focus on:
  - classes
  - relationships
  - *not* details of objects
- identification of objects is difficult

Unified Modeling Language

- Developed specifically for OO, circa 1999

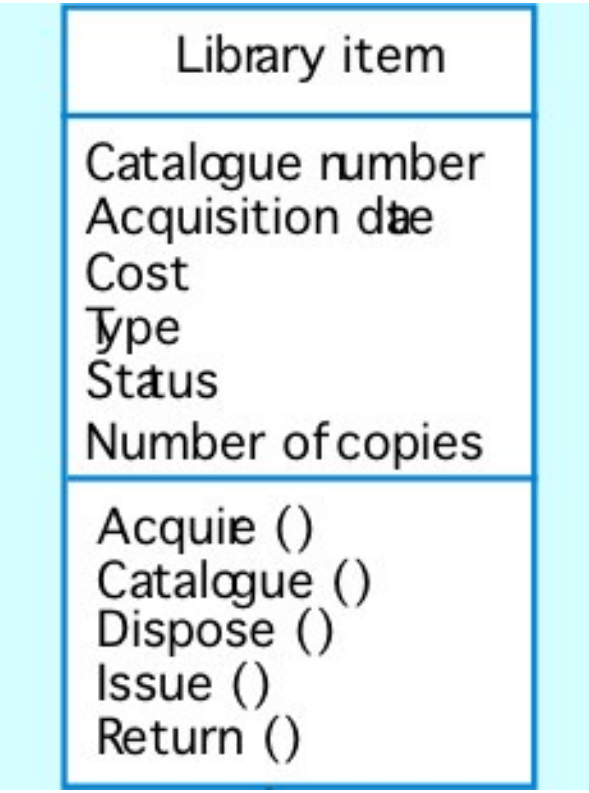
Unified Modeling Language

- Developed specifically for OO, circa 1999
- Standard graphical language with several notation scheme

Unified Modeling Language

- Developed specifically for OO, circa 1999
- Standard graphical language with several notation scheme
- Class represented as rectangle with sections:
  - name
  - attributes
  - operations

UML Example


Library item
Catalogue number
Acquisition date
Cost
Type
Status
Number of copies
Acquire ()
Catalogue ()
Dispose ()
Issue ()
Return ()

Inheritance Models

- Class model is organized into a *taxonomy*

Inheritance Models

- Class model is organized into a *taxonomy*
  - classification scheme showing how classes are related

Inheritance Models

- Class model is organized into a *taxonomy*
  - classification scheme showing how classes are related
- Taxonomy is often shown by *inheritance diagram*

Inheritance Models

- Class model is organized into a *taxonomy*
  - classification scheme showing how classes are related
- Taxonomy is often shown by *inheritance diagram*
  - general object at top
  - specialized objects at bottom
    - sharing attributes and services

Inheritance Models

- Class model is organized into a *taxonomy*
  - classification scheme showing how classes are related
- Taxonomy is often shown by *inheritance diagram*
  - general object at top
  - specialized objects at bottom
    - sharing attributes and services
- inheritance show upwards/downwards

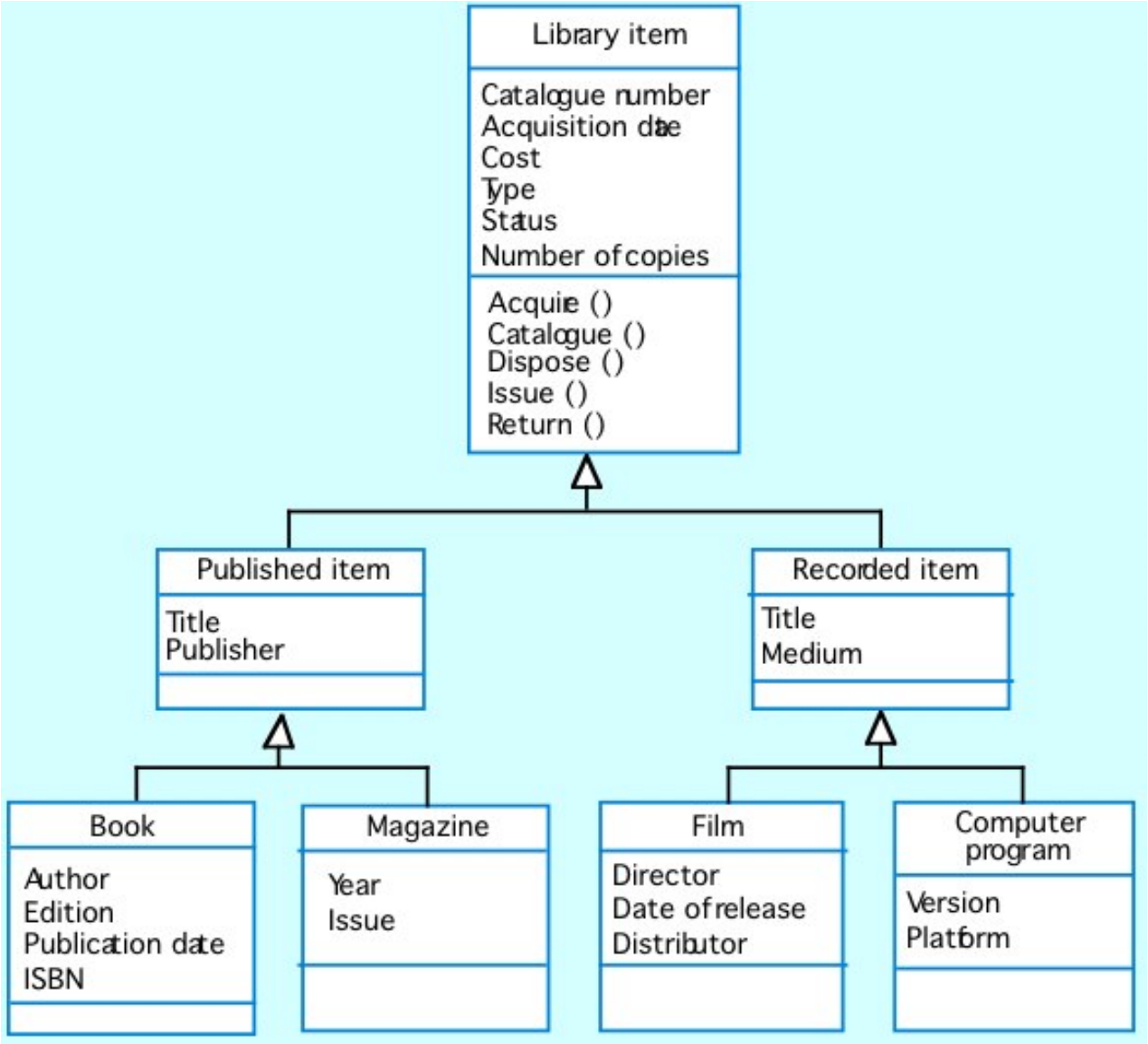
Inheritance Models

- Class model is organized into a *taxonomy*
  - classification scheme showing how classes are related
- Taxonomy is often shown by *inheritance diagram*
  - general object at top
  - specialized objects at bottom
    - sharing attributes and services
- inheritance show upwards/downwards
  - point from subclass to super class

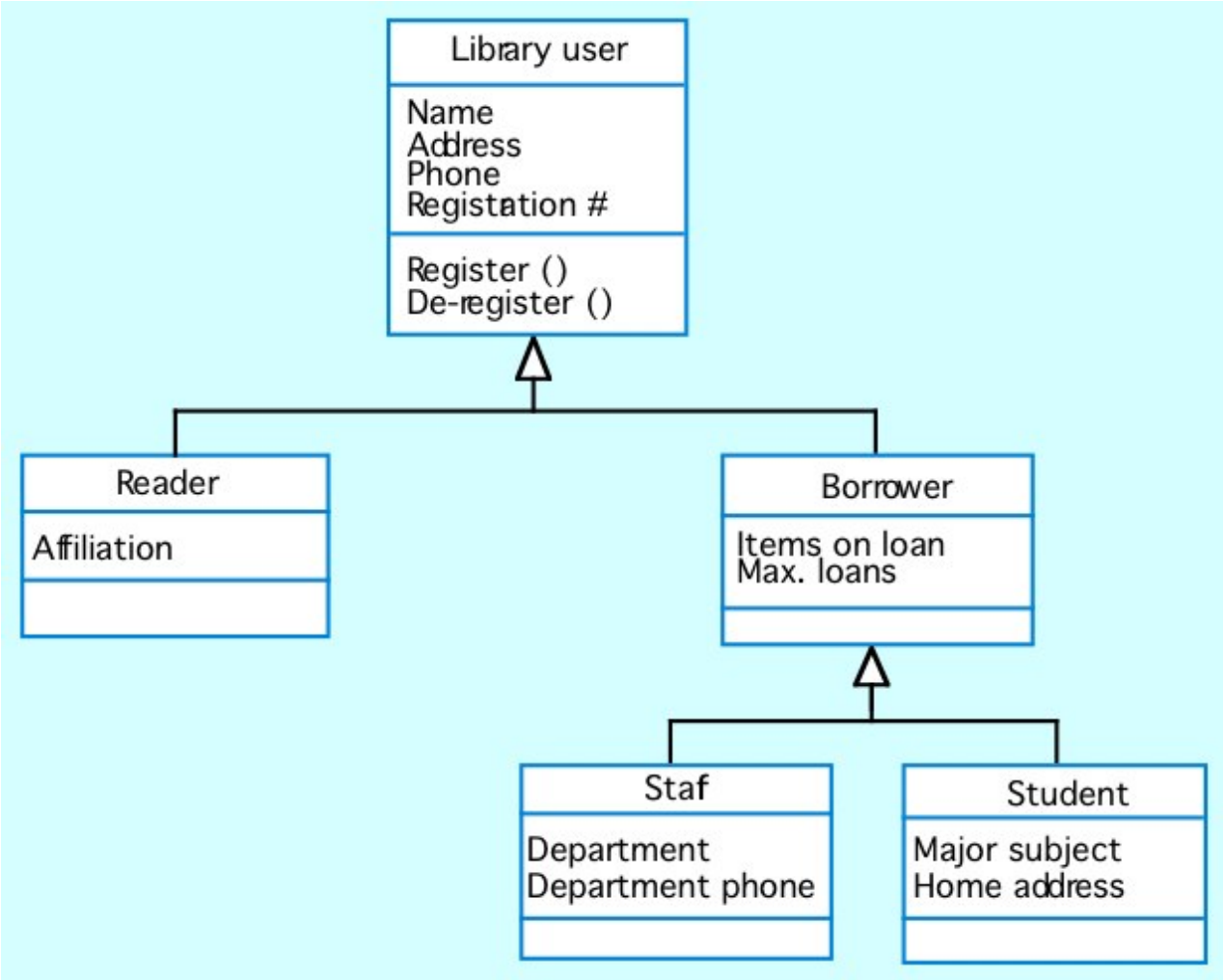
Inheritance Models

- Class model is organized into a *taxonomy*
  - classification scheme showing how classes are related
- Taxonomy is often shown by *inheritance diagram*
  - general object at top
  - specialized objects at bottom
    - sharing attributes and services
- inheritance show upwards/downwards
  - point from subclass to super class
- inheritance → *generalization relationship* in UML

Library Item Inheritance Model



Library User Inheritance Model



Object Modeling

- Requires domain knowledge; subtle issues are common

Object Modeling

- Requires domain knowledge; subtle issues are common
- For e.g.:
  - *why is the 'title' not in Library Item?*

Object Modeling

- Requires domain knowledge; subtle issues are common
- For e.g.:
  - *why is the 'title' not in Library Item?*
  - *how much generalization?*

Object Modeling

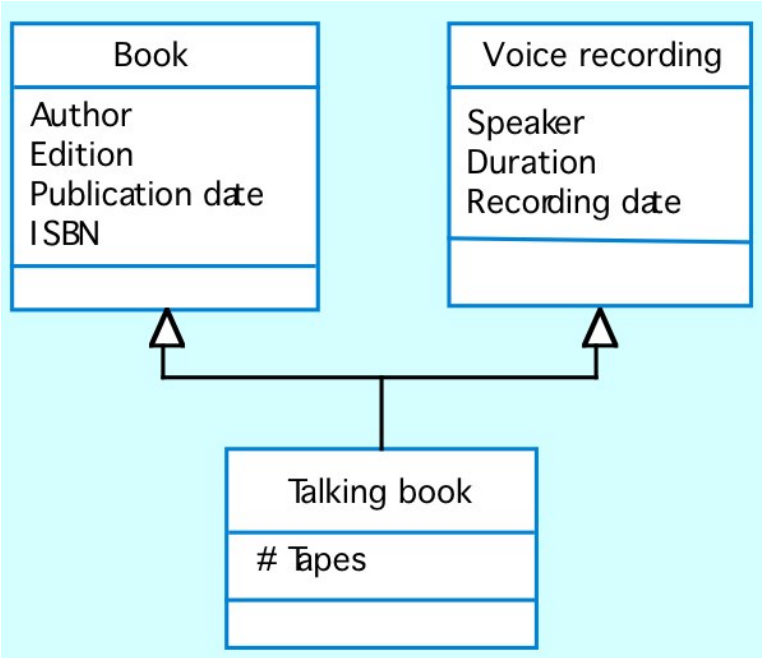
- Requires domain knowledge; subtle issues are common
- For e.g.:
  - *why is the 'title' not in Library Item?*
  - *how much generalization?*

Multiple Inheritance

- Can inherit from multiple parent classes

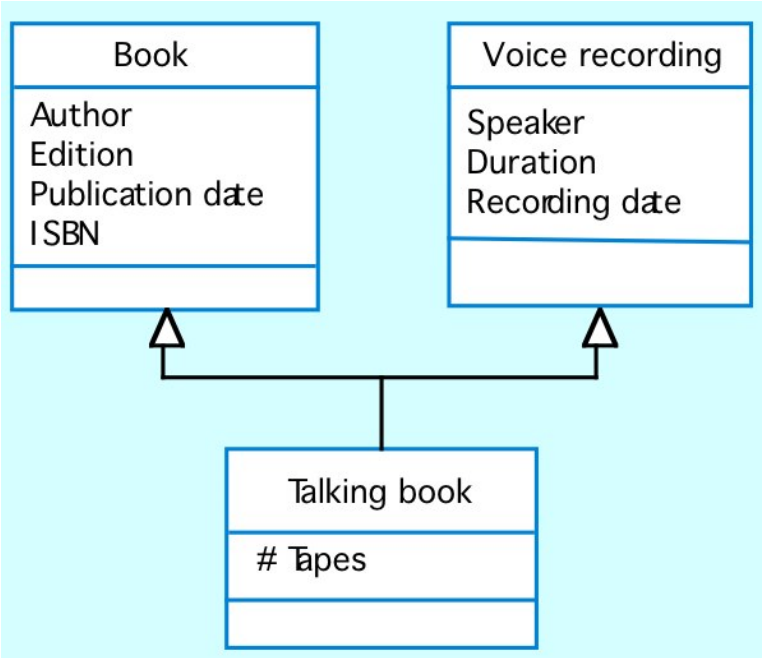
Multiple Inheritance

- Can inherit from multiple parent classes



Multiple Inheritance

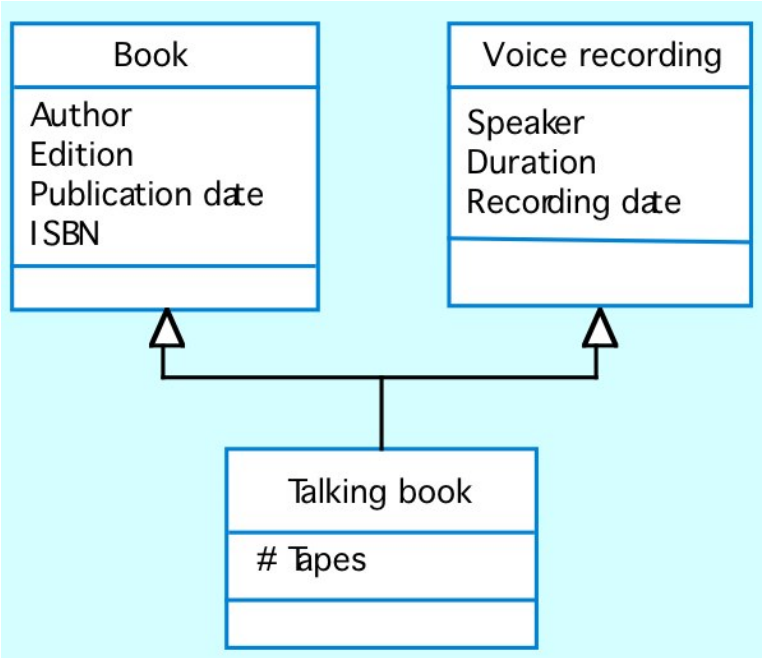
- Can inherit from multiple parent classes



- Issues:

Multiple Inheritance

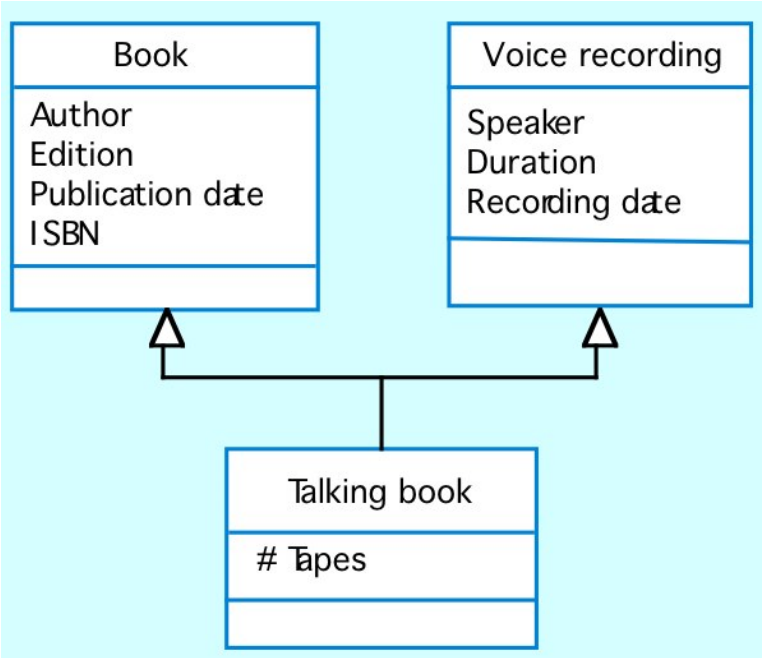
- Can inherit from multiple parent classes



- Issues:
  - should not inherit unnecessary attributes

Multiple Inheritance

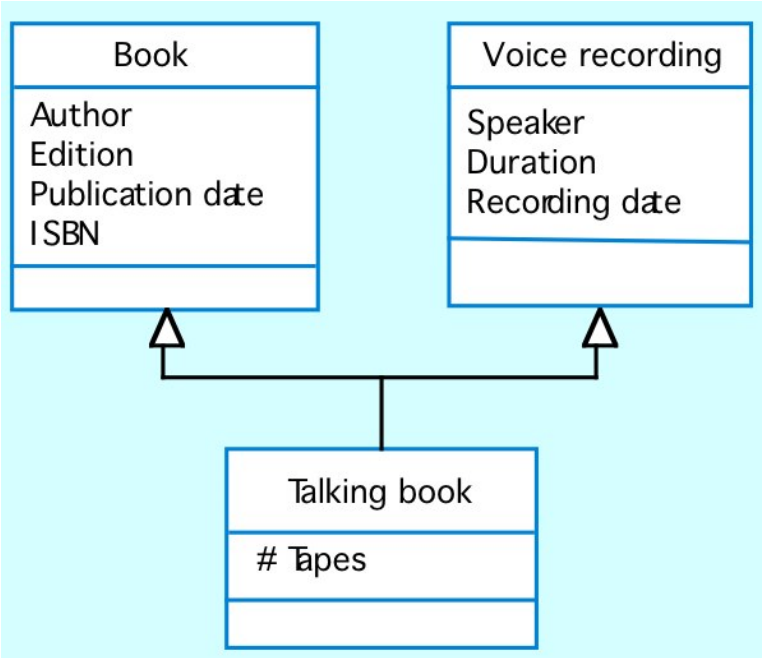
- Can inherit from multiple parent classes



- Issues:
  - should not inherit unnecessary attributes
  - reorganizing is difficult

Multiple Inheritance

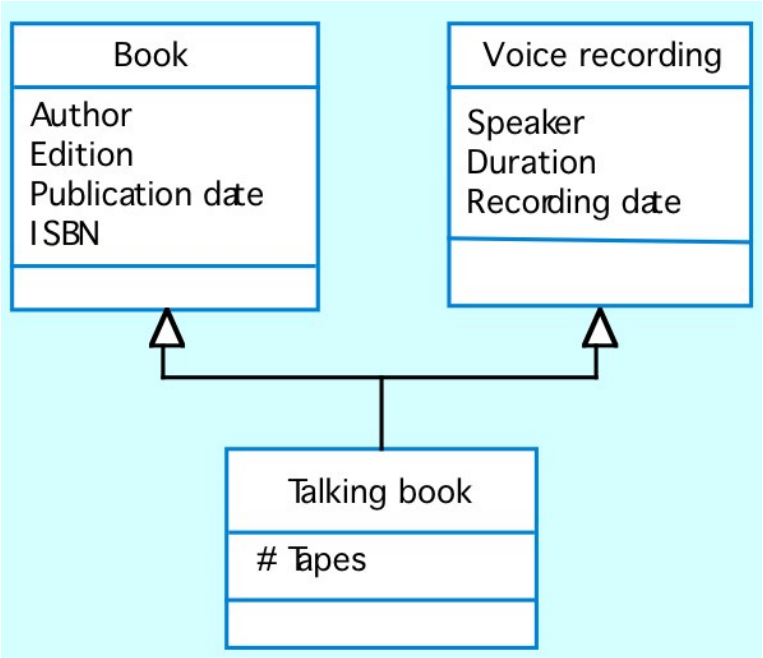
- Can inherit from multiple parent classes



- Issues:
  - should not inherit unnecessary attributes
  - reorganizing is difficult
  - name clashes

Multiple Inheritance

- Can inherit from multiple parent classes



- Issues:
  - should not inherit unnecessary attributes
  - reorganizing is difficult
  - name clashes
  - *more problematic at programming stage*

Object Aggregation

- An object may contain other objects

Object Aggregation

- An object may contain other objects