CSC 342

Software Engineering

Chapter 7:

System Models

Instructor:

Dr. Ghazy Assassa

System Models

- Abstract descriptions of systems whose requirements are being analysed

Objectives

- To explain why the context of a system should be modelled as part of the RE process
- To describe
  - Interface modelling
  - behavioural modelling
  - data modelling
  - and object modelling
- To introduce some of the notations used in the Unified Modeling Language (UML)

Topics covered

- Context models
- Behavioural models
- Data models
- Object models
- CASE workbenches

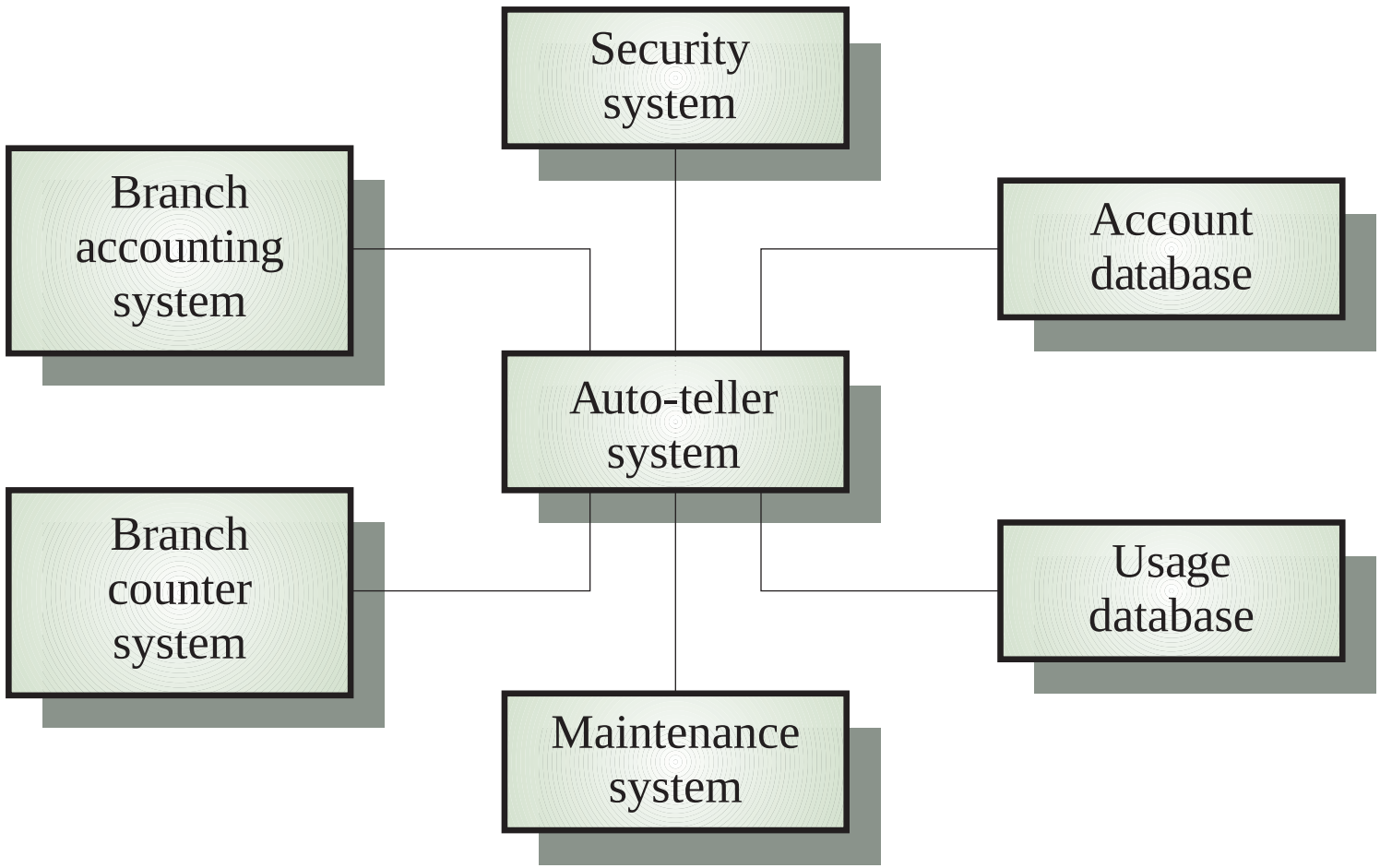
System modelling

- Different models present the system from different perspectives
  - External perspective showing the system's context or environment: **Interface model**
  - Behavioural perspective showing the behaviour of the system: **State machine model & Process model**
  - Structural perspective showing the system or data architecture: **Data Model**

Context models

- Context models are used to illustrate the boundaries of a system
- Architectural models show the a system and its relationship with other systems

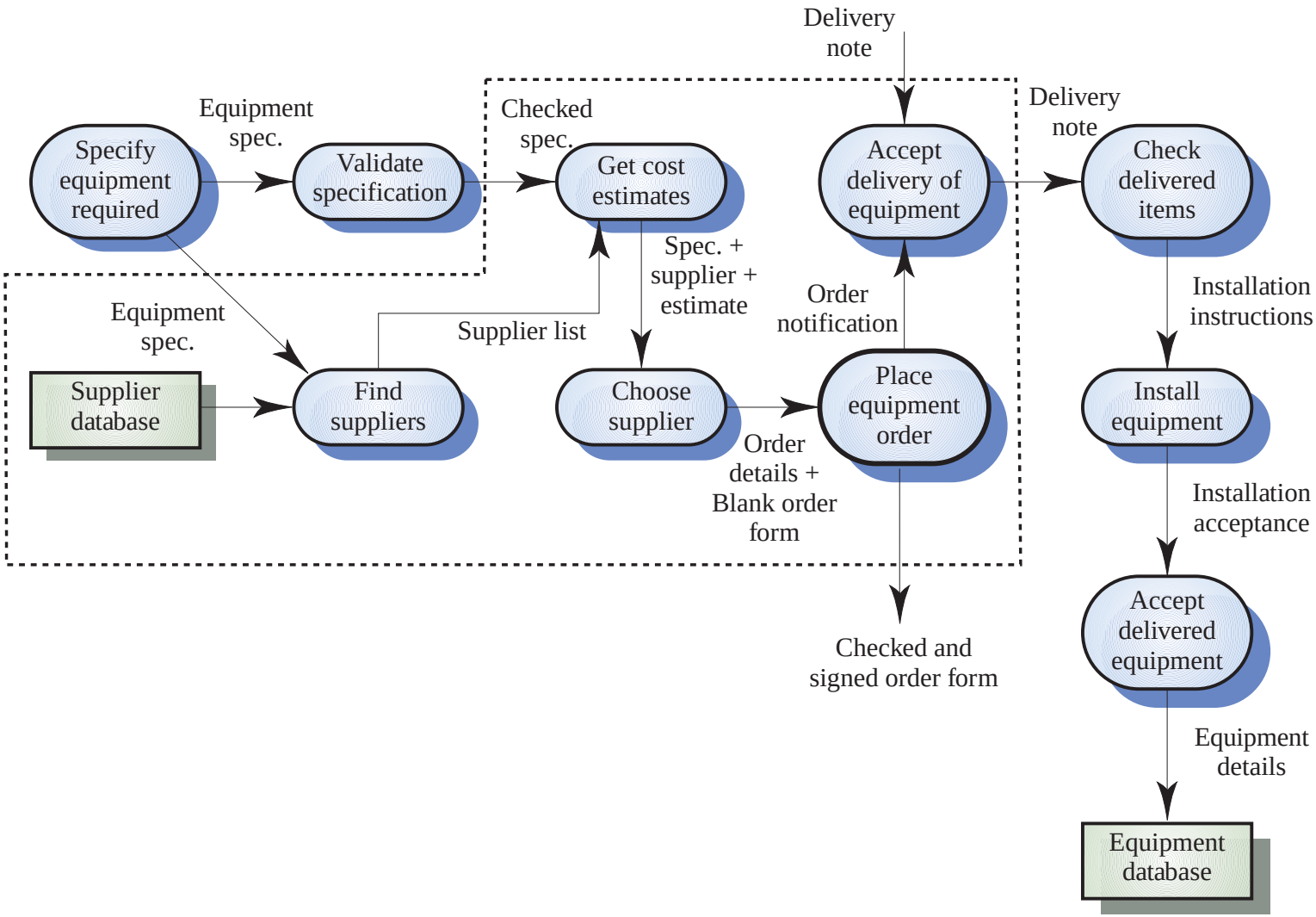
The context of an ATM system



Process models

- Process models show the overall process and the processes that are supported by the system

Equipment procurement process



Behavioural models

- Behavioural models are used to describe the overall behaviour of a system

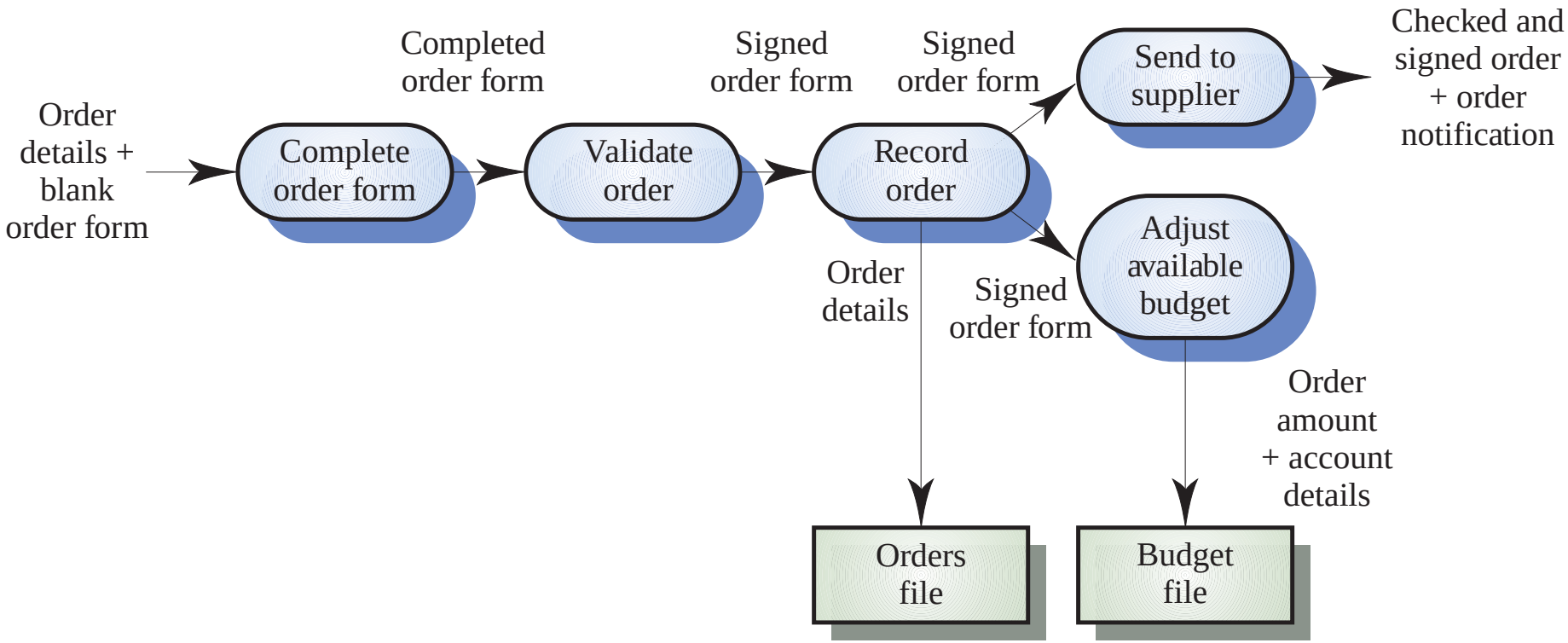
- Two types of behavioural model:
  - Data-Process models that show how data is processed as it moves through the system
  - State machine models that show the systems response to events

Data-processing models - DFD

- Data flow diagrams are used to model the system's data processing

- These show the processing steps as data flows through a system

Order processing DFD



Data flow diagrams

- DFDs model the system from a functional perspective
- Data flow diagrams may also be used in showing the data exchange between a system and other systems in its environment

State machine models

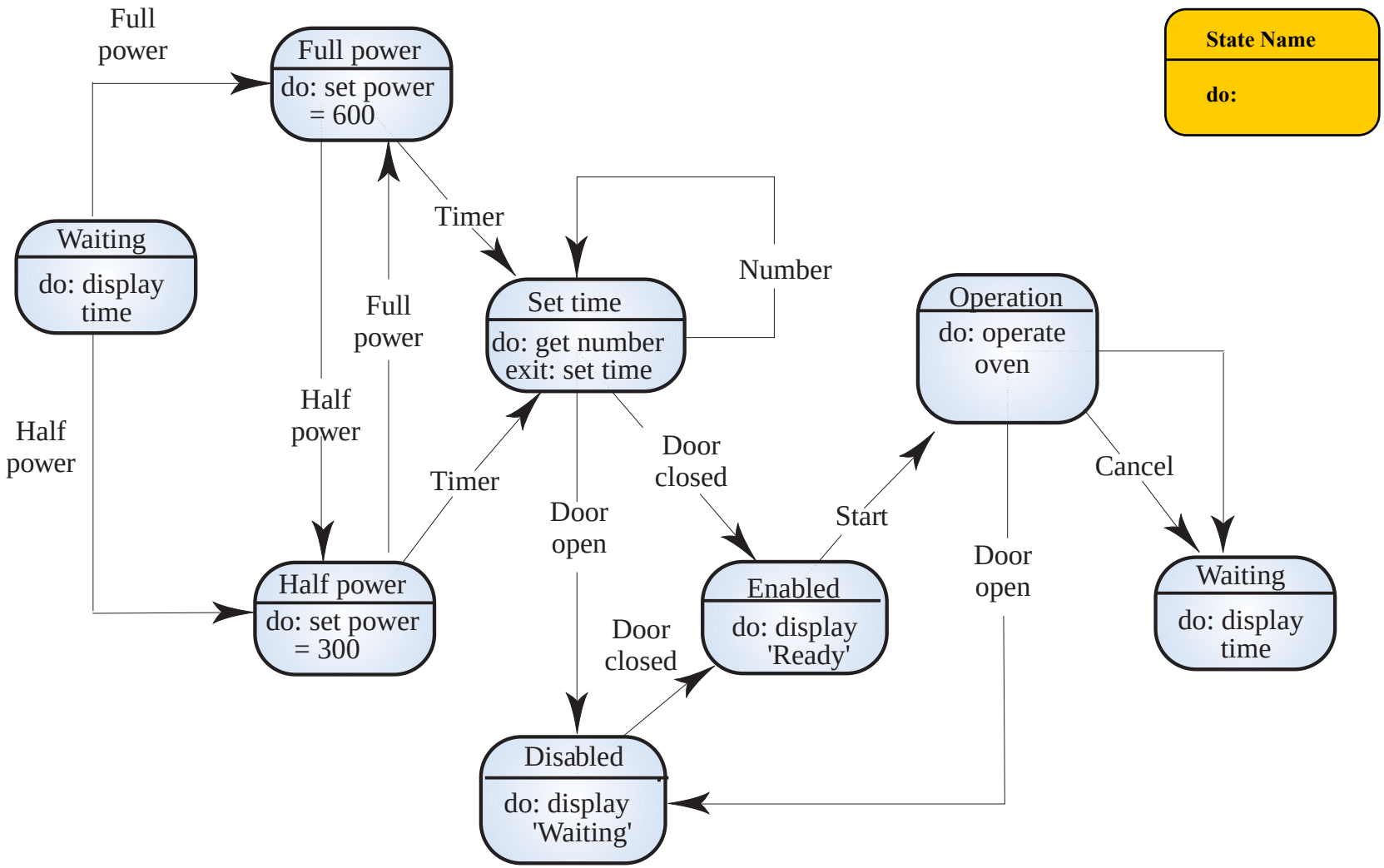
- These model the behaviour of the system in response to external and internal events

- They show the system's responses to stimuli so are often used for modelling real-time systems

State machine models

- State machine models show
  - system states as nodes and
  - system events as arcs between these nodes.
- When an event occurs, the system moves from one state to another
- State charts are an integral part of the UML

Microwave oven model



Microwave oven state description

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 set 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 seconds. Oven light is on. Display shows 'Cooking complete' while buzzer is sounding.

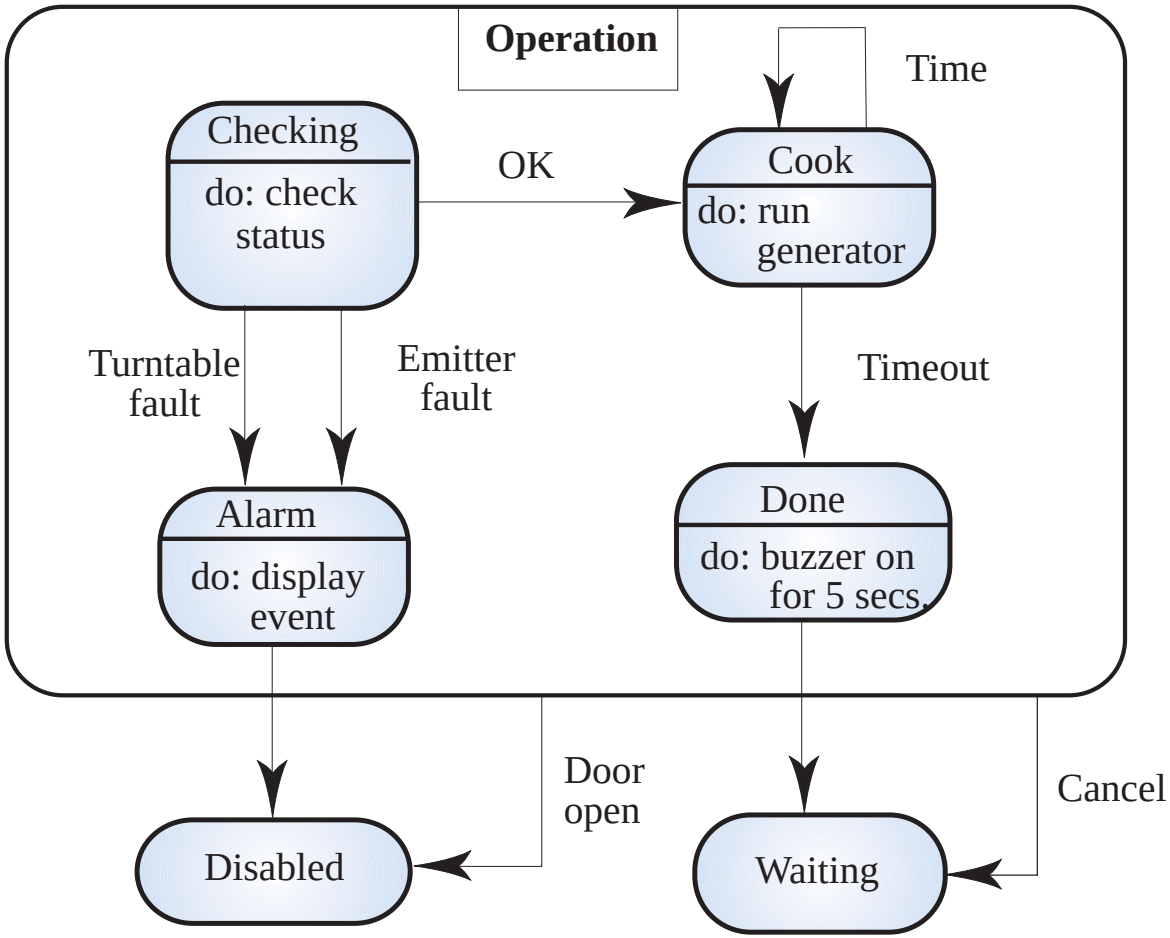
Microwave oven stimuli

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

State charts

- Allow the decomposition of a model into sub-models (see following slide)
- A brief description of the actions is included following the 'do' in each state
- Can be complemented by tables describing the states and the stimuli

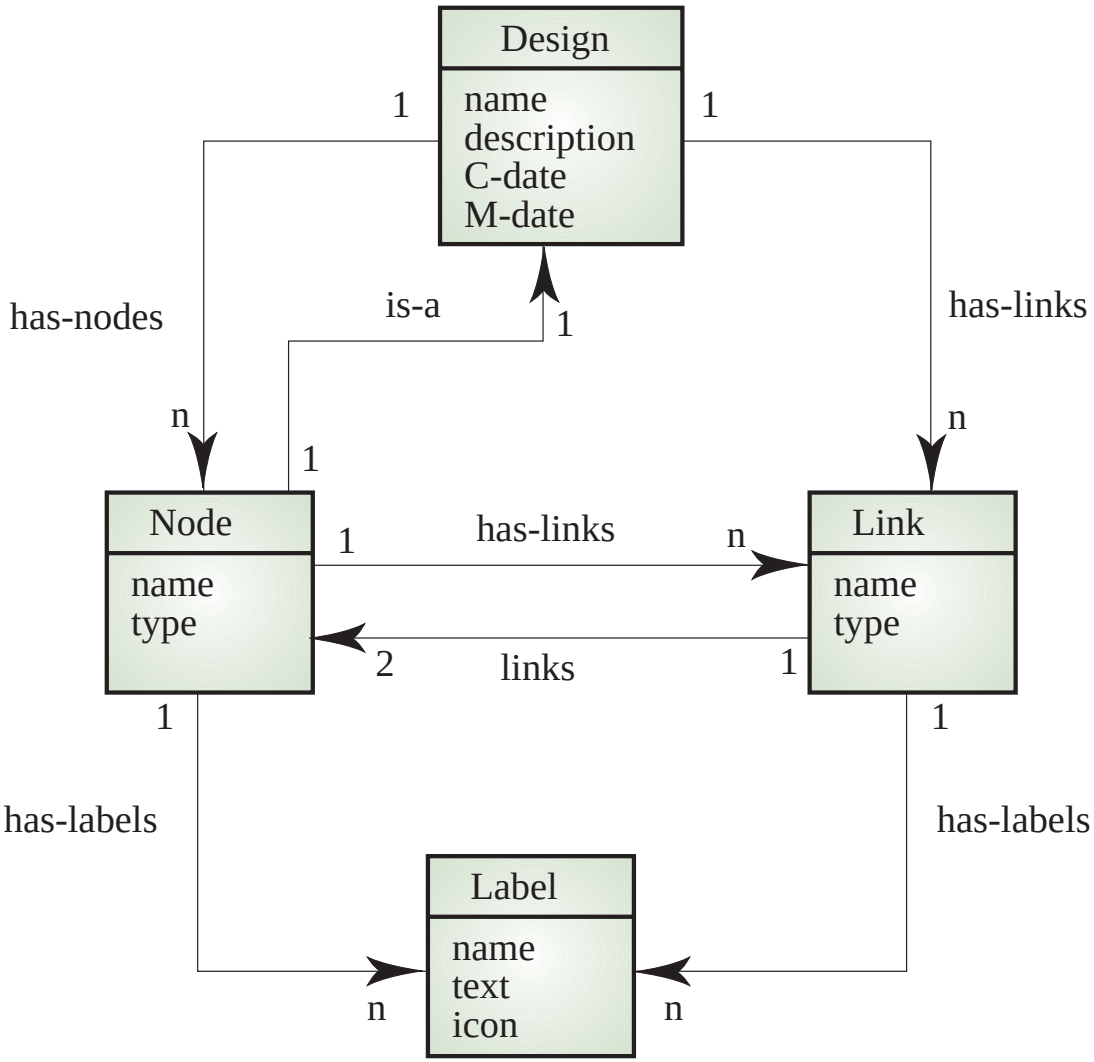
Microwave oven operation



Semantic data models – ER

- Used to describe the logical structure of data processed by the system
- Entity-relation-attribute model sets out the entities in the system, the relationships between these entities and the entity attributes
- Widely used in database design. Can readily be implemented using relational databases
- No specific notation provided in the UML but objects and associations can be used

Software design semantic model



Data dictionaries

- Data dictionaries are lists of all of the names used in the system models. Descriptions of the entities, relationships and attributes are also included
- Advantages
  - Support name management and avoid duplication
  - Store of organisational knowledge linking analysis, design and implementation
- Many CASE workbenches support data dictionaries

Data dictionary entries

Name	Description	Type	Date
has-labels	1:N relation between entities of type Node or Link and entities of type Label.	Relation	5.10.1998
Label	Holds structured or unstructured information about nodes or links. Labels are represented by an icon (which can be a transparent box) and associated text.	Entity	8.12.1998
Link	A 1:1 relation between design entities represented as nodes. Links are typed and may be named.	Relation	8.12.1998
name (label)	Each label has a name which identifies the type of label. The na me must be unique within the set of label types used in a design.	Attribute	8.12.1998
name (node)	Each node has a name which must be unique within a design. The name may be up to 64 characters long.	Attribute	15.11.1998

Object models

- Object models describe the system in terms of object classes

- An object class is an abstraction over a set of objects with common attributes and the services (operations) provided by each object

Object models

- Various object models may be produced
  - Inheritance models
  - Aggregation models
  - Interaction models

- Natural ways of reflecting the real-world entities manipulated by the system

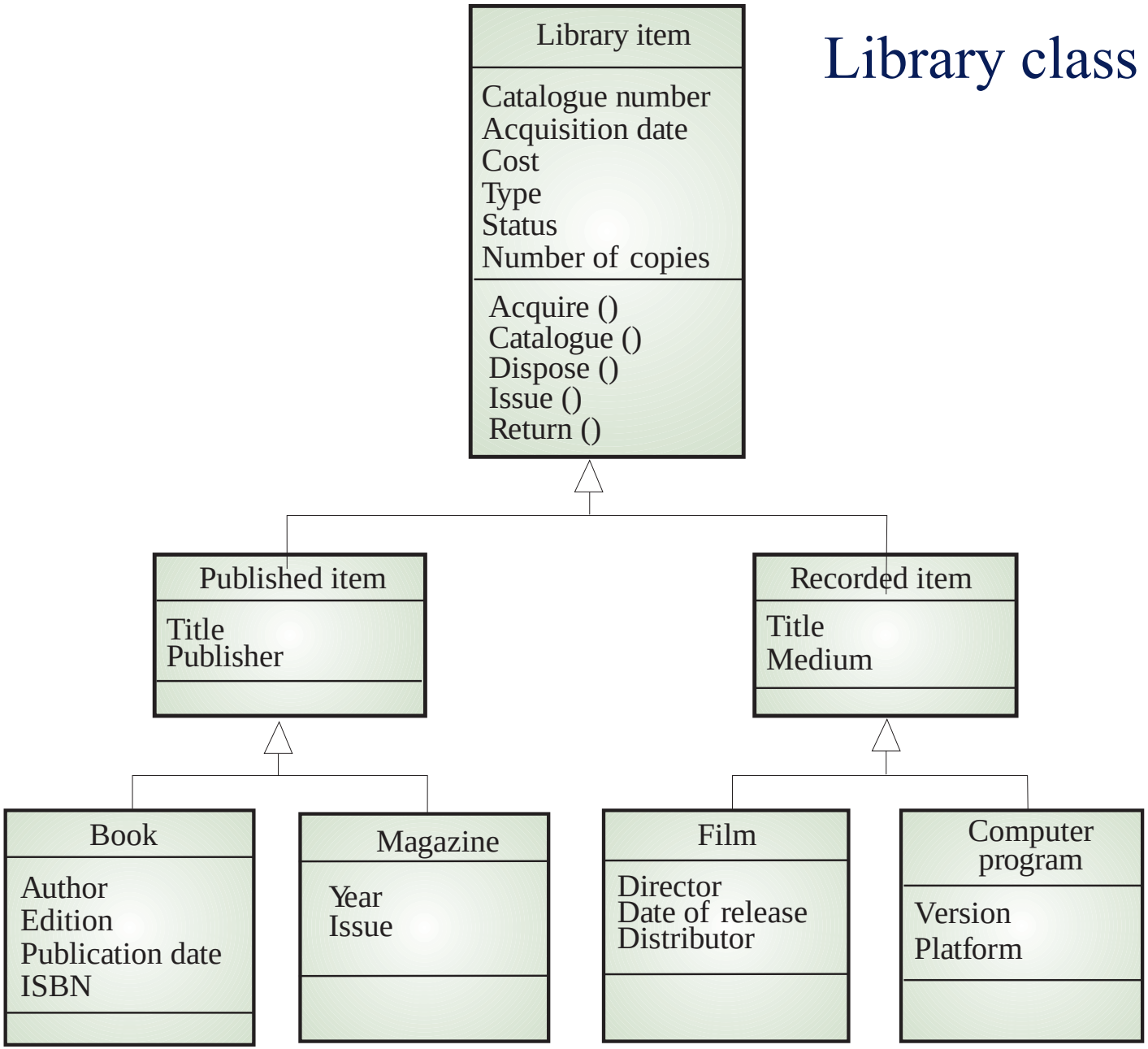
Inheritance models

- Organise the domain object classes into a hierarchy
- Classes at the top of the hierarchy reflect the common features of all classes
- Object classes
  - inherit their attributes and services from one or more super-classes.
  - these may then be specialised as necessary

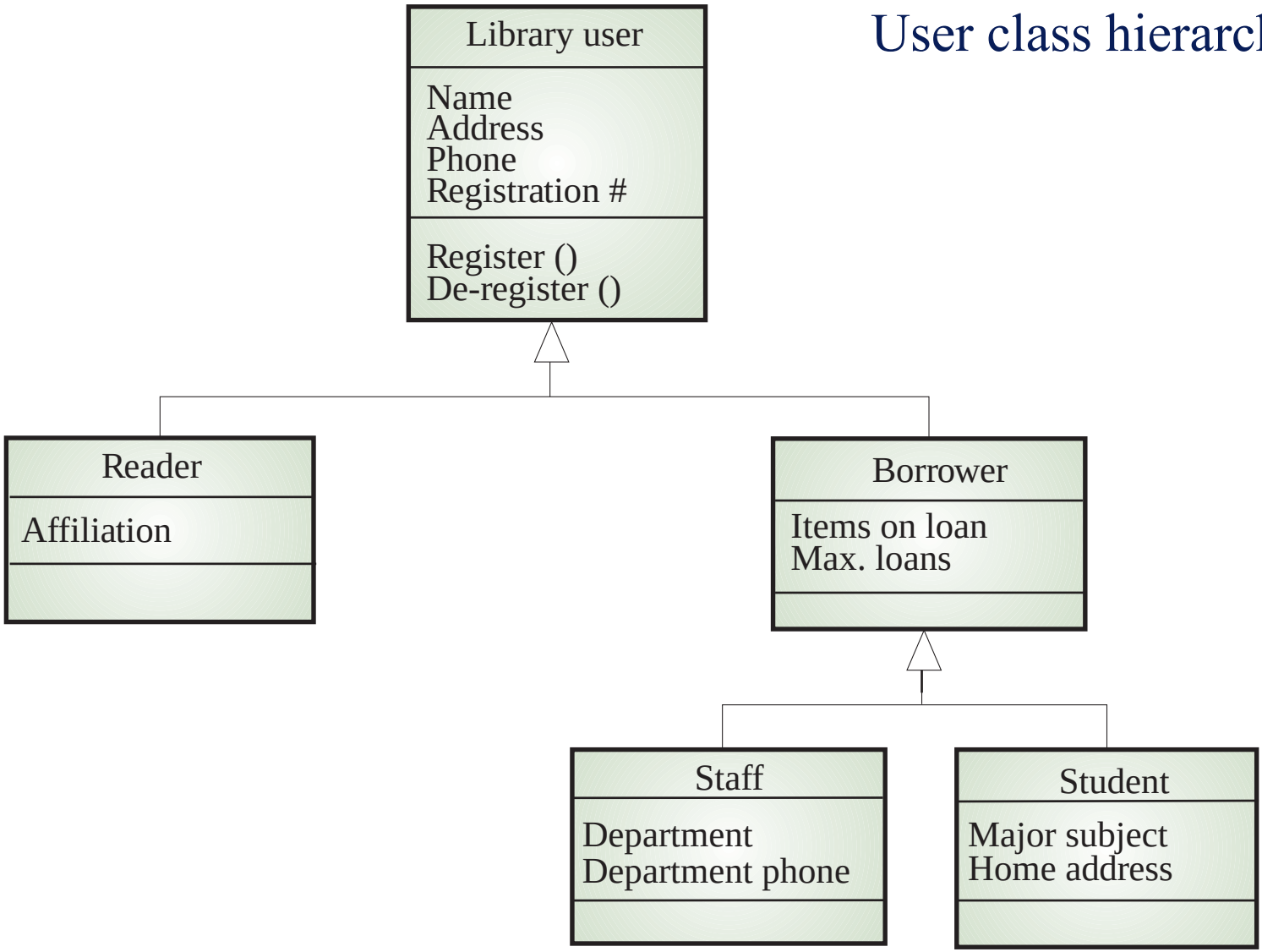
The Unified Modeling Language

- Devised by the developers of widely used object-oriented analysis and design methods
- Has become an effective standard for object-oriented modelling
- Notation
  - Object classes are rectangles with the name at the top, attributes in the middle section and operations in the bottom section
  - Relationships between object classes (known as associations) are shown as lines linking objects
  - Inheritance is referred to as generalisation and is shown 'upwards' rather than 'downwards' in a hierarchy

Library class hierarchy



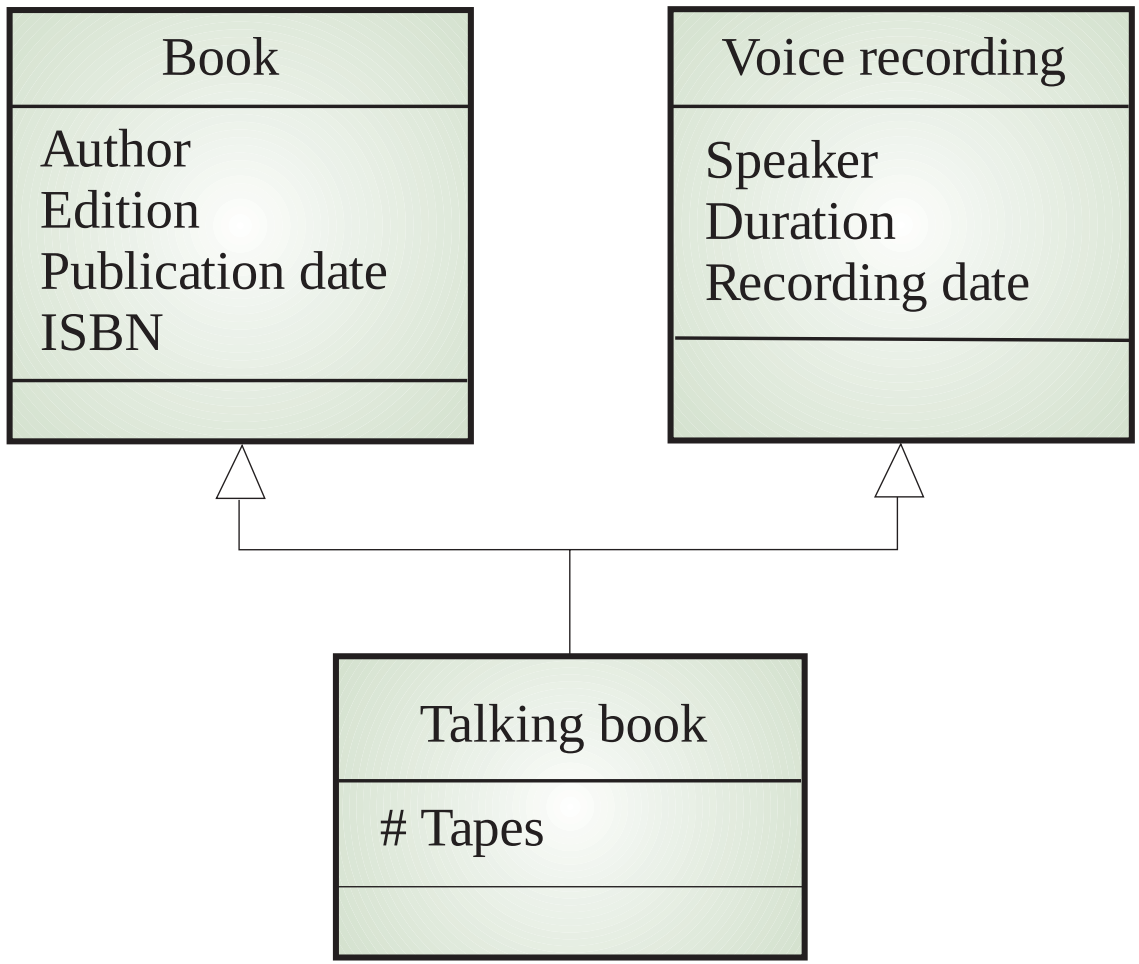
User class hierarchy



Multiple inheritance

- Rather than inheriting the attributes and services from a single parent class, a system which supports multiple inheritance allows object classes to inherit from several super-classes
- Can lead to semantic conflicts where attributes/services with the same name in different super-classes have different semantics
- Makes class hierarchy reorganisation more complex

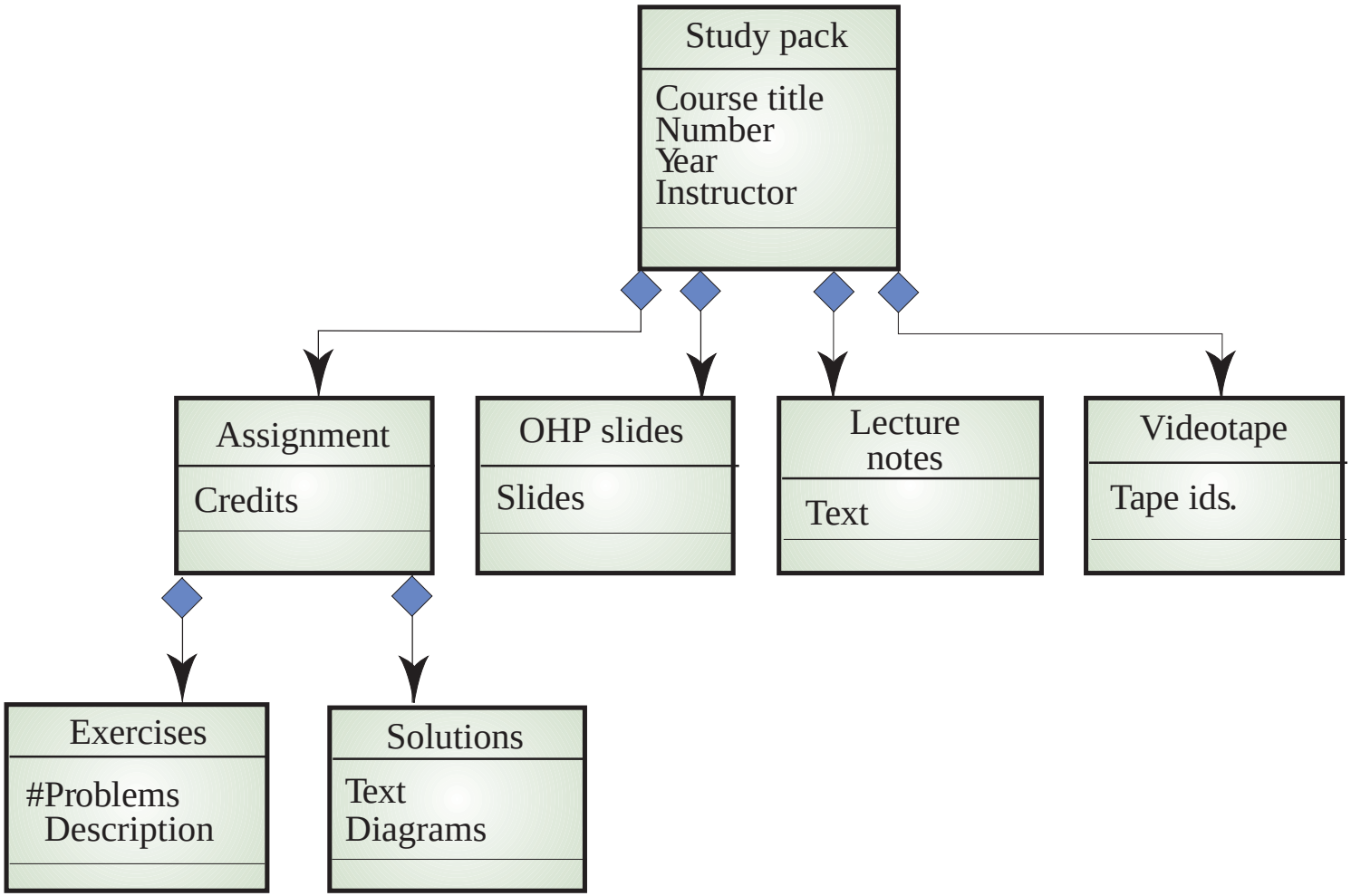
Multiple inheritance



Object aggregation

- Aggregation model shows how classes which are collections are composed of other classes
- Similar to the part-of relationship in semantic data models

Object aggregation

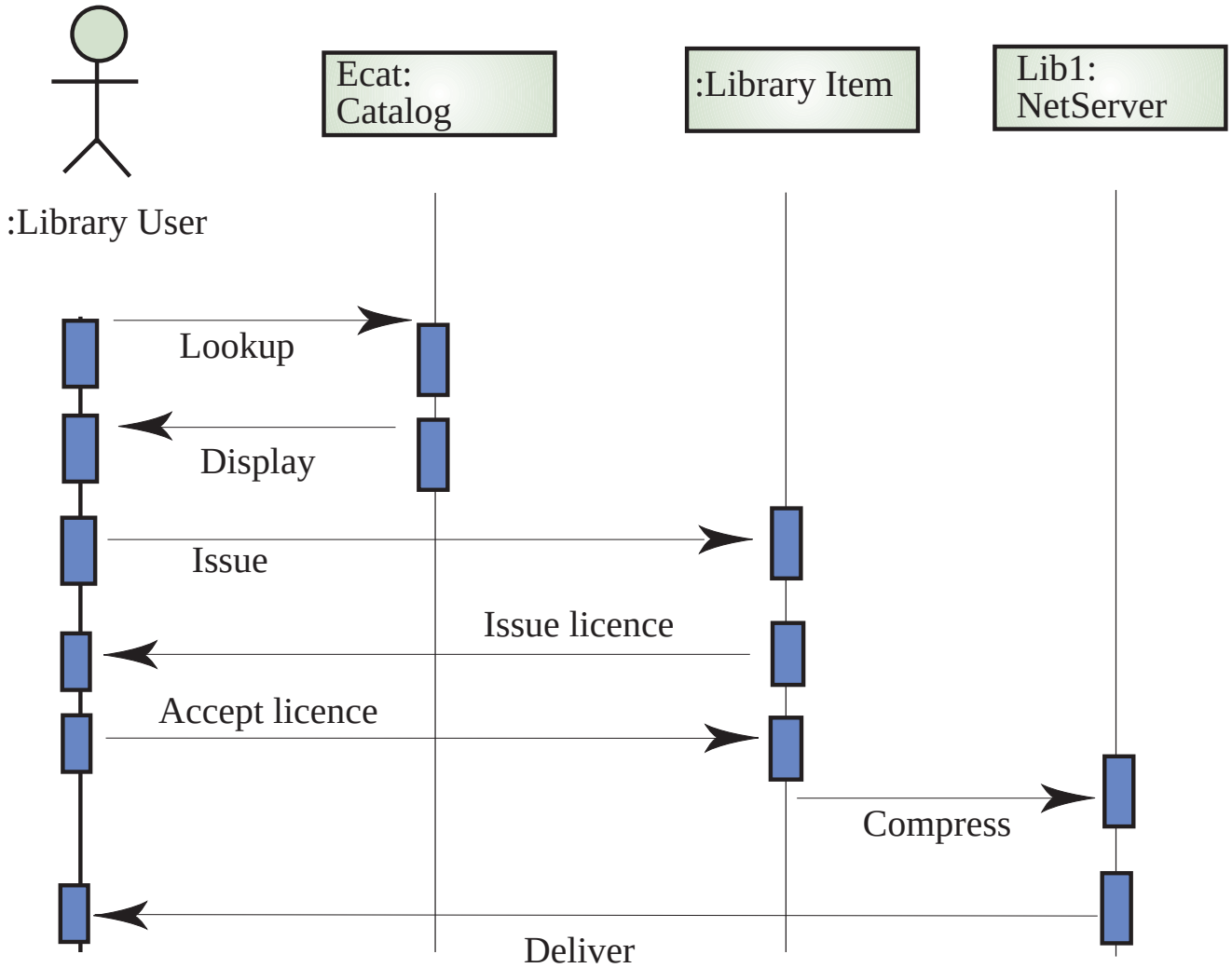


Object behaviour modelling

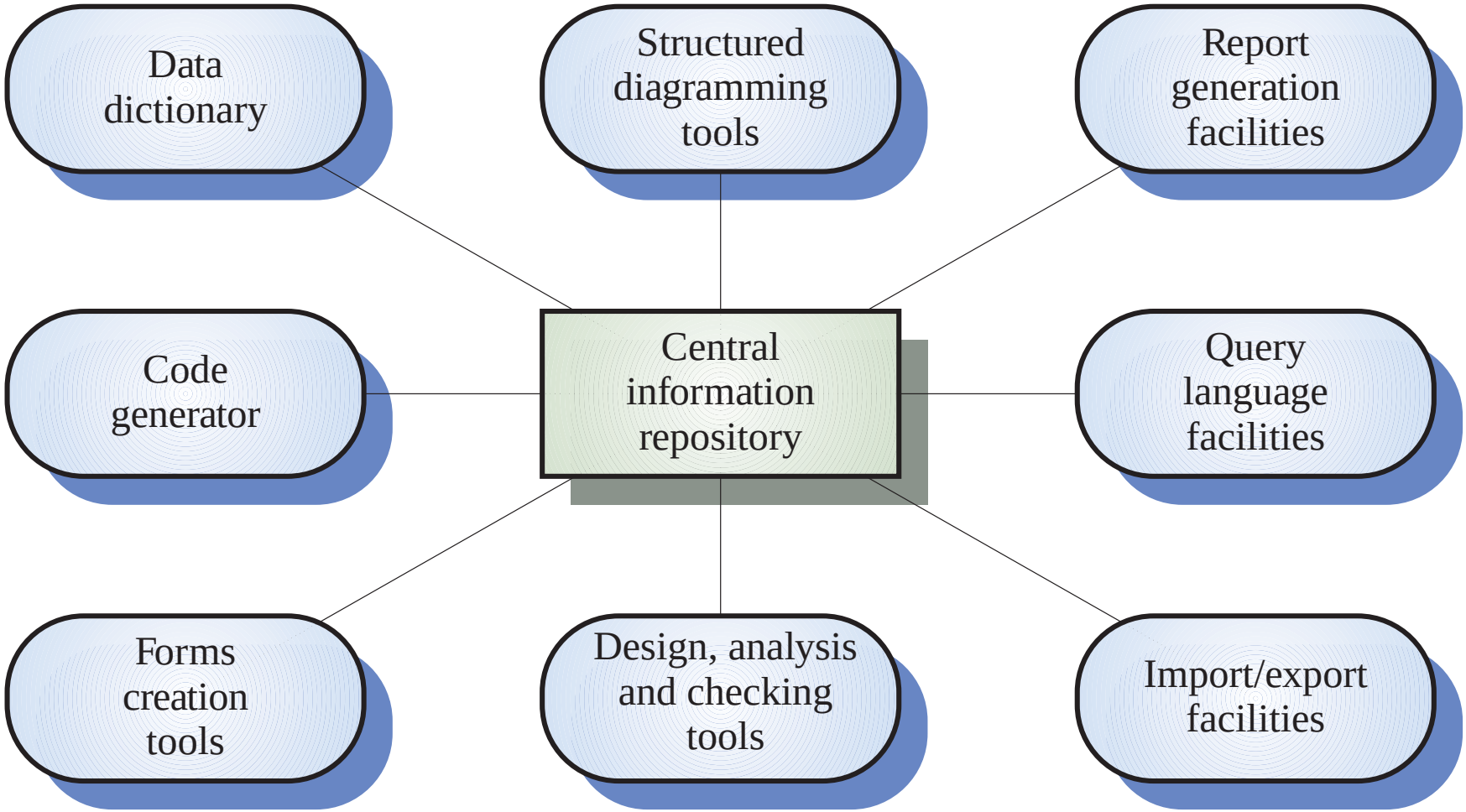
- A behavioural model shows the interactions between objects to produce some particular system behaviour that is specified as a use-case

- Sequence diagrams 'Time based' (or collaboration diagrams) in the UML are used to model interaction between objects

Sequence Diagram:
EX: Issue of electronic items

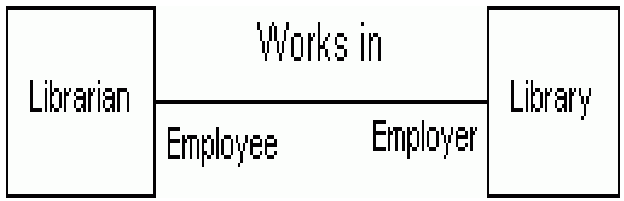


An analysis and design workbench



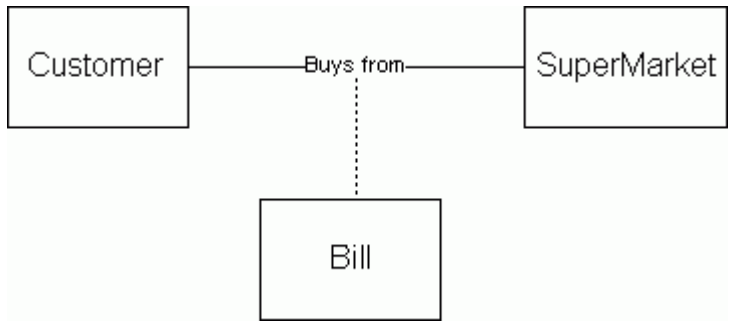
UML: **Associations** of regular classes

- When classes are connected together conceptually, that connection is called an association



UML: **Association class**

- Like a class, an association can have attributes and operations
- You visualize **association class** the same way you show a **regular class**
- Use dotted line to connect **association class** to the association line



UML: Multiplicity

Multiplicity:

- shows the number of objects from one class that relate with a number of objects in an associated class.

UML: Multiplicity

One class can be relate to another in a:

- one-to-one
- one-to-many
- one-to-one or more
- one-to-zero or one
- one-to-a bounded interval (one-to-two through twenty)
- one-to-exactly n
- one-to-a set of choices (one-to-five or eight)
- The UML uses an asterisk (*) to represent ***more*** and to represent ***many***.

OO: Visibility of attributes or operations

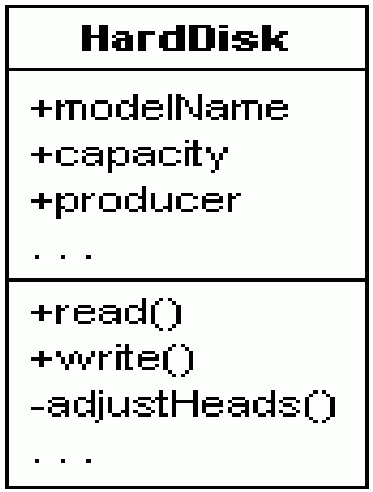
- Visibility: specifies the extent to which other classes can use a given class's attributes or methods/operations.

Three levels of visibility:

- \+ **: public level (**usability extends to other classes)
- \# **: protected level (**usability is open only to classes that inherit from original class)
- \- **: private level (**only the original class can use the attribute or operation)

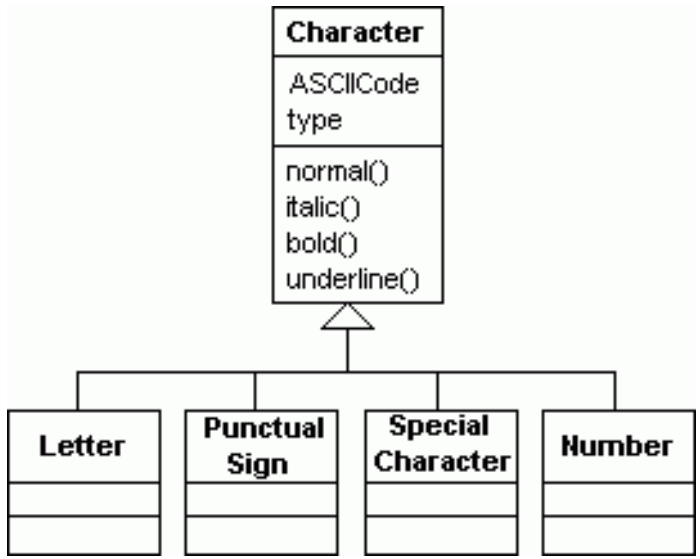
OO: Visibility

Ex: *Public and private operations in a Hard Disk*



Ex: *The character hierarchy*

- The **Character class** will have ***ASCIIcode*** and ***type*** as attributes (type tells the type of the character - normal, italic, bold or underline), and ***normal()***, ***bold()***, ***italic()*** and ***underline()*** as operations. The Character class children will be: **Letter**, **PunctualSign**, **SpecialCharacter** and **Number**.



Analysis workbench components

- Diagram editors
- Model analysis and checking tools
- Repository and associated query language
- Data dictionary
- Report definition and generation tools
- Forms definition tools
- Import/export translators
- Code generation tools

Key points

- A model is an abstract system view. Complementary types of model provide different system information
- Context models show the position of a system in its environment with other systems and processes
- Data flow models may be used to model the data processing in a system
- State machine models model the system's behaviour in response to internal or external events

Key points

- Semantic data models describe the logical structure of data which is imported to or exported by the systems
- Object models describe logical system entities, their classification and aggregation
- Object models describe the logical system entities and their classification and aggregation
- CASE workbenches support the development of system models