

**MATA KULIAH** : Rekayasa Piranti Lunak

**KODE MATA KULIAH/SKS** : TI1014– 4 SKS

**TAHUN** : 2013

**VERSI** : 1.0



EDUCATION FOR A BETTER LIFE



# KEMAMPUAN AKHIR YANG DIHARAPKAN

---

Mahasiswa mengenal metode dan  
Kebutuhan(Requirements) pada  
Rekayasa Piranti Lunak



# MATERI POKOK

---

- Model Konteks
- Model Perilaku
- Model data
- Model Obyek
- Metode Terstruktur



# Sumber Pustaka

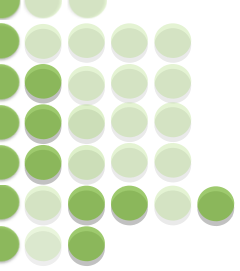
---

1. A. S. Rosa, Salahuddin M., Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek, Informatika, 20013.
2. Ian Sommerville, Software Engineering 9th Edition, Addison-Wesley, 2011.
3. Roger S. Pressman, Software Engineering: A Practitioner's Approach Seventh Edition, McGraw-Hill, 2010.
4. Yasin Verdi, Rekayasa Perangkat Lunak Berorientasi Objek, Mitra Wacana Media, 2012.



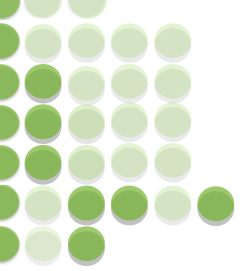
# System modelling

- System modelling helps the analyst to understand the functionality of the system and models are used to communicate with customers.
- Different models present the system from different perspectives
  - External perspective showing the system's context or environment;
  - Behavioural perspective showing the behaviour of the system;
  - Structural perspective showing the system or data architecture.



# Model types

- Data processing model showing how the data is processed at different stages.
- Composition model showing how entities are composed of other entities.
- Architectural model showing principal sub-systems.
- Classification model showing how entities have common characteristics.
- Stimulus/response model showing the system's reaction to events.



# The Unified Modeling Language (UML)

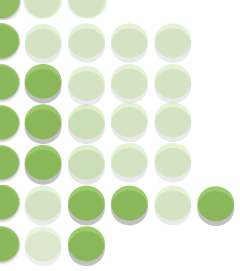
- The UML has become a standard diagrammatic notation for developing models of object-oriented systems.
- The UML is a unification of a range of different approaches to OO modelling that were developed in the 1990s.
- Its latest version (UML 2) also includes more general modelling constructs.
- UML 1 was the most widely used when this chapter was written. I have therefore used this notation here.  
Consequently, some diagrams such as data-flow diagrams are not expressed in the UML.



# UML 2 model types

- UML 2 has 13 different types of diagram that are used to model different aspects of a system
  - Behavioural diagrams are used to model behavioural features of a system. For example, use-case diagrams, activity diagrams, state machine diagrams
  - Interaction diagrams are used to model interactions between entities in the system. For example, sequence diagrams and communication diagrams.
  - Structure diagrams are used to model the organization of the system. For example, class diagrams, package diagrams, and deployment diagrams.

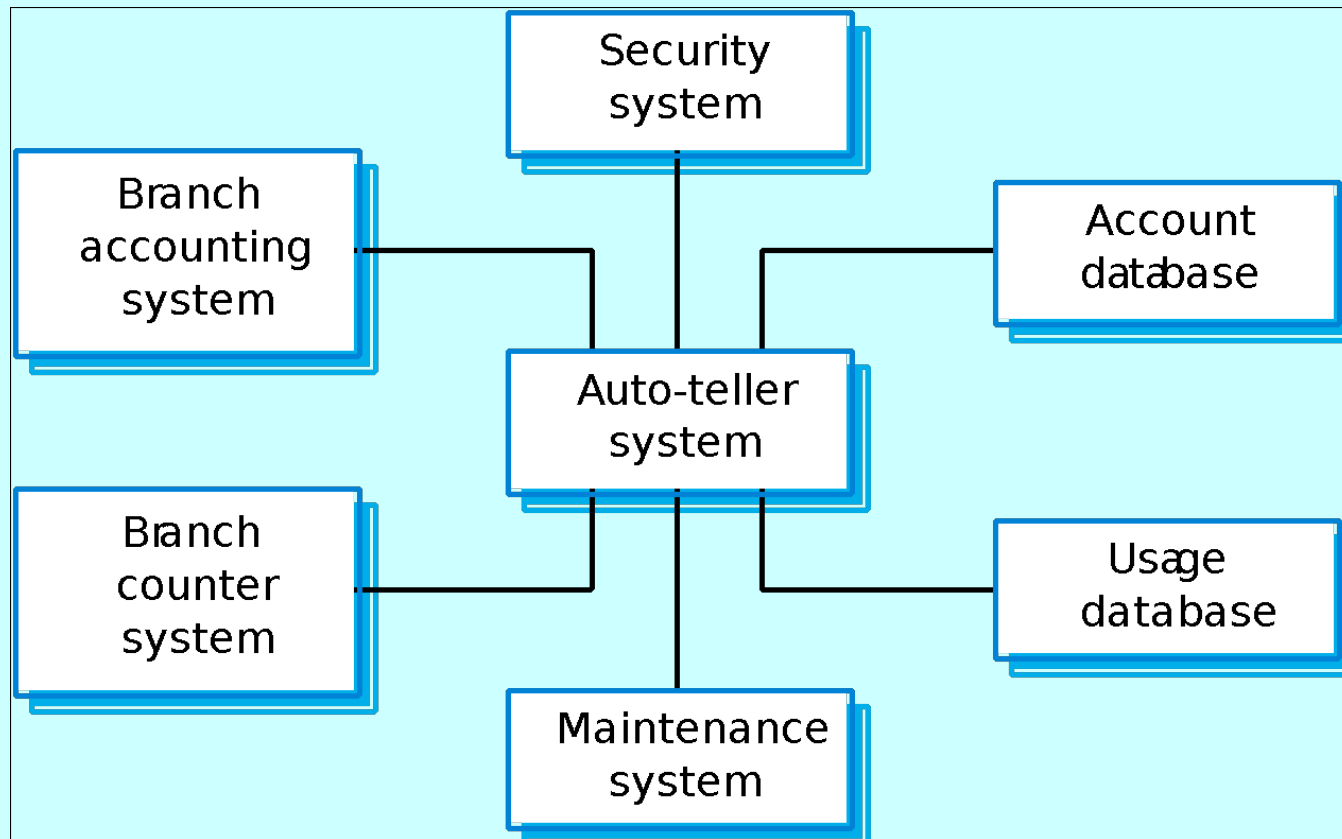


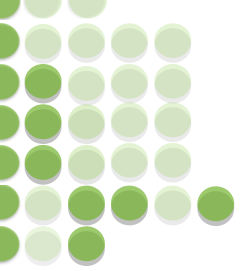


# Context models

- Context models are used to illustrate the operational context of a system - they show what lies outside the system boundaries.
- Social and organisational concerns may affect the decision on where to position system boundaries.
- Architectural models show the system and its relationship with other systems.

# The context of an ATM system

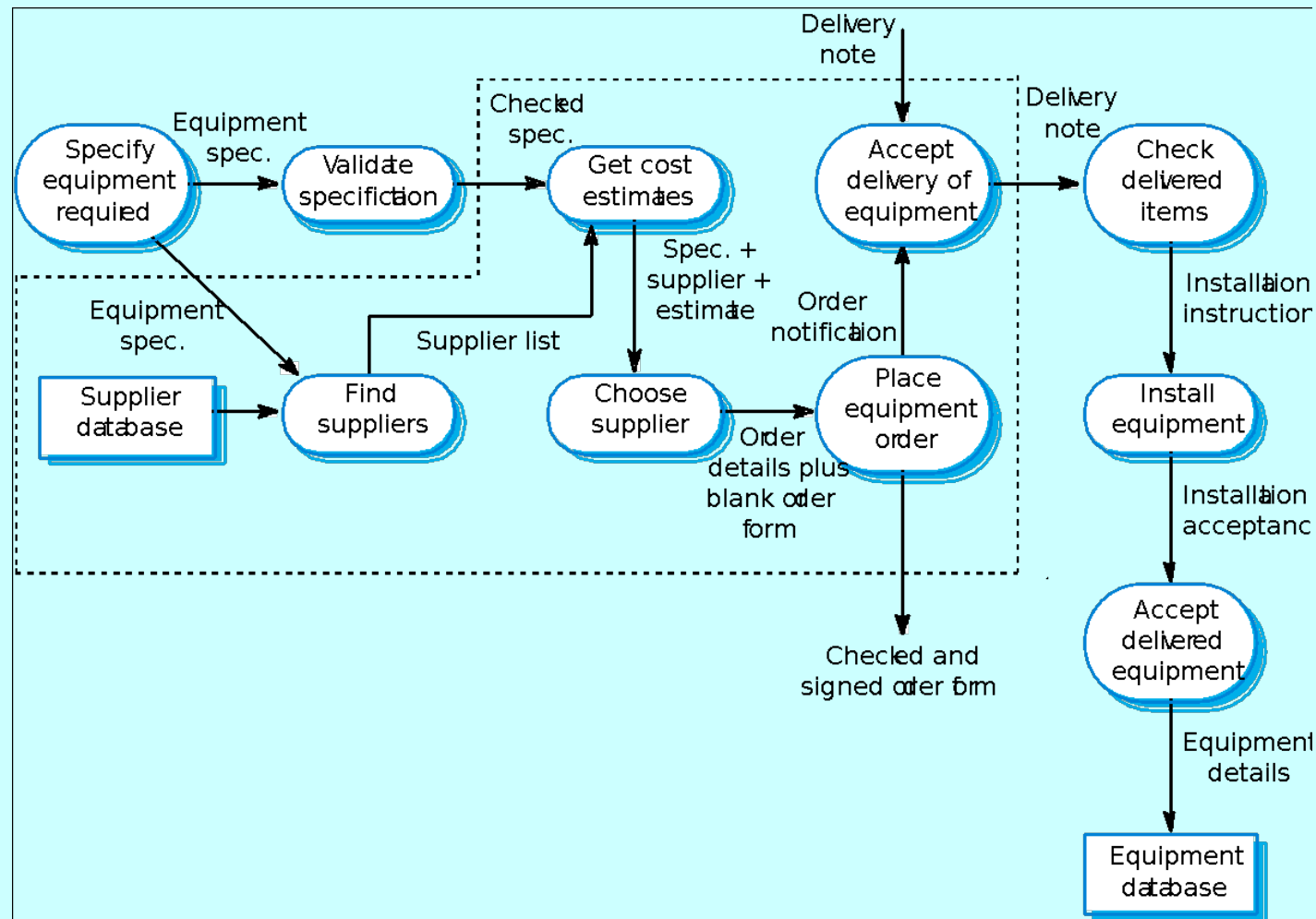


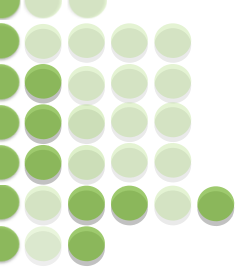


# Process models

- Process models show the overall process and the processes that are supported by the system.
- Data flow models may be used to show the processes and the flow of information from one process to another.
- These types of diagram are sometimes known as workflow diagrams.

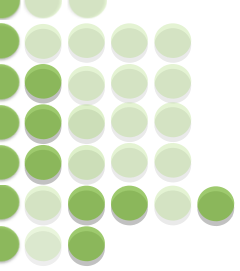
# Equipment procurement process





# Behavioural models

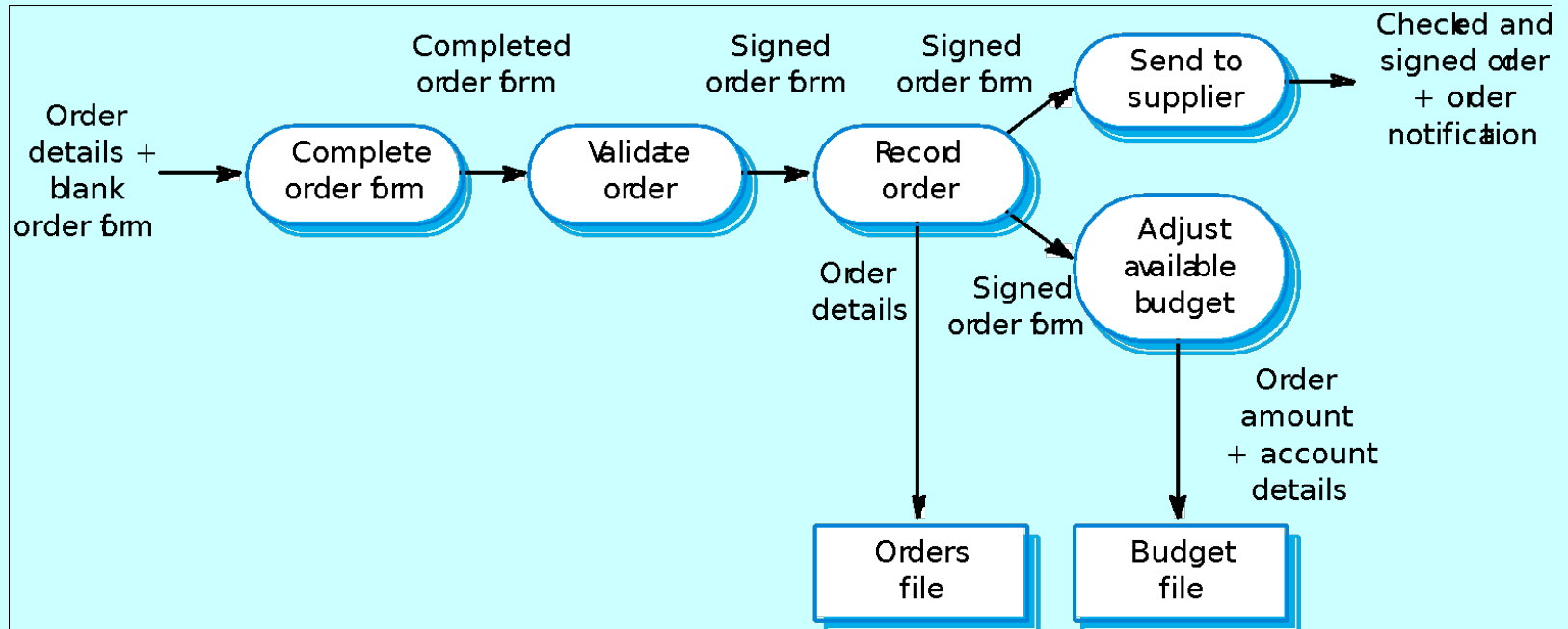
- Behavioural models are used to describe the overall behaviour of a system.
- Two types of behavioural model are:
  - Data processing models that show how data is processed as it moves through the system;
  - State machine models that show the systems response to events.
- These models show different perspectives so both of them are required to describe the system's behaviour.

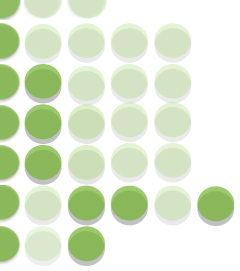


# Data-processing models

- Data flow diagrams (DFDs) may be used to model the system's data processing.
- These show the processing steps as data flows through a system.
- DFDs are an intrinsic part of many analysis methods.
- Simple and intuitive notation that customers can understand.
- Show end-to-end processing of data.

# Order processing DFD



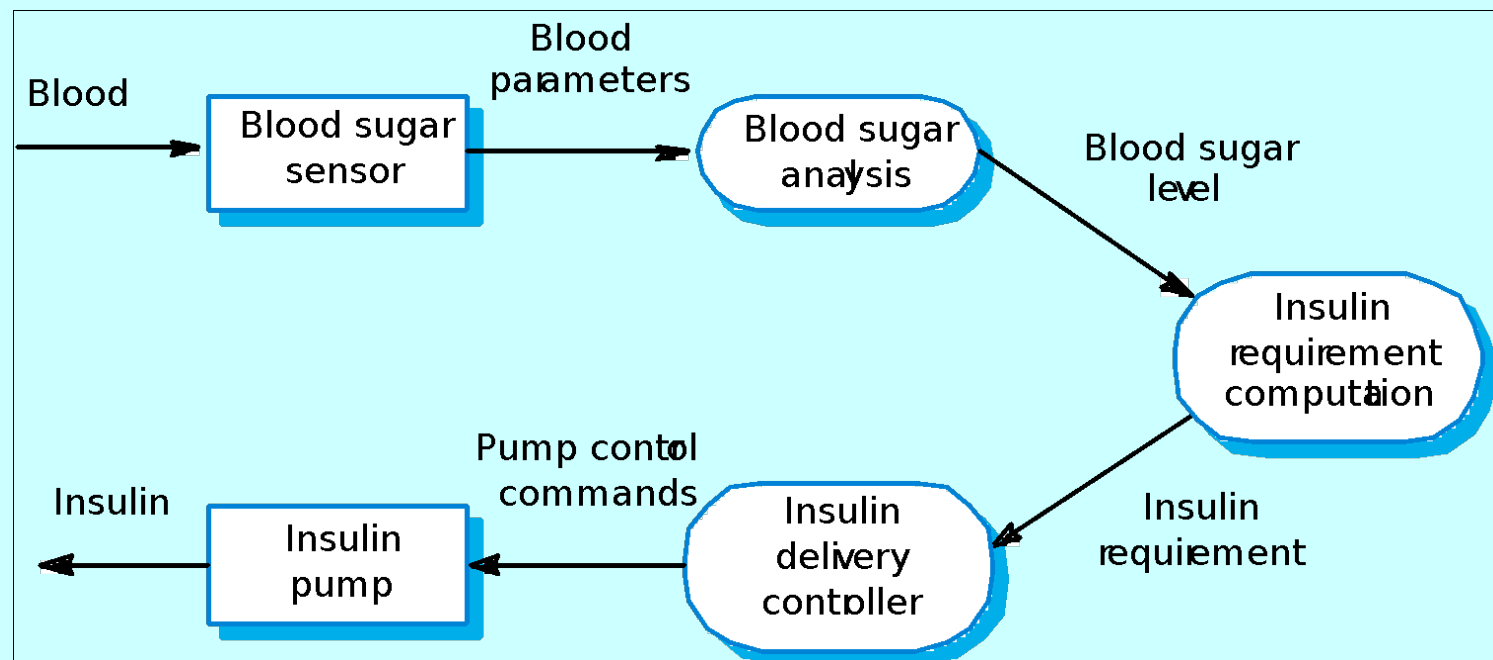


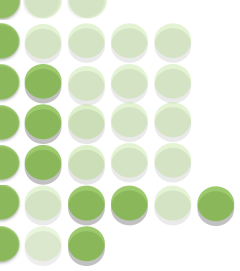
# Data flow diagrams

- DFDs model the system from a functional perspective.
- Tracking and documenting how the data associated with a process is helpful to develop an overall understanding of the system.
- Data flow diagrams may also be used in showing the data exchange between a system and other systems in its environment.



# Insulin pump DFD





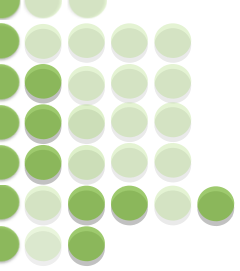
# UML diagrams

- To model workflow or data-flow, you can use UML 2 activity diagrams.
- These provide a richer notation than that used here so translation is simple
- UML state machine diagrams may be used to model how events are processed by a system



# State machine models

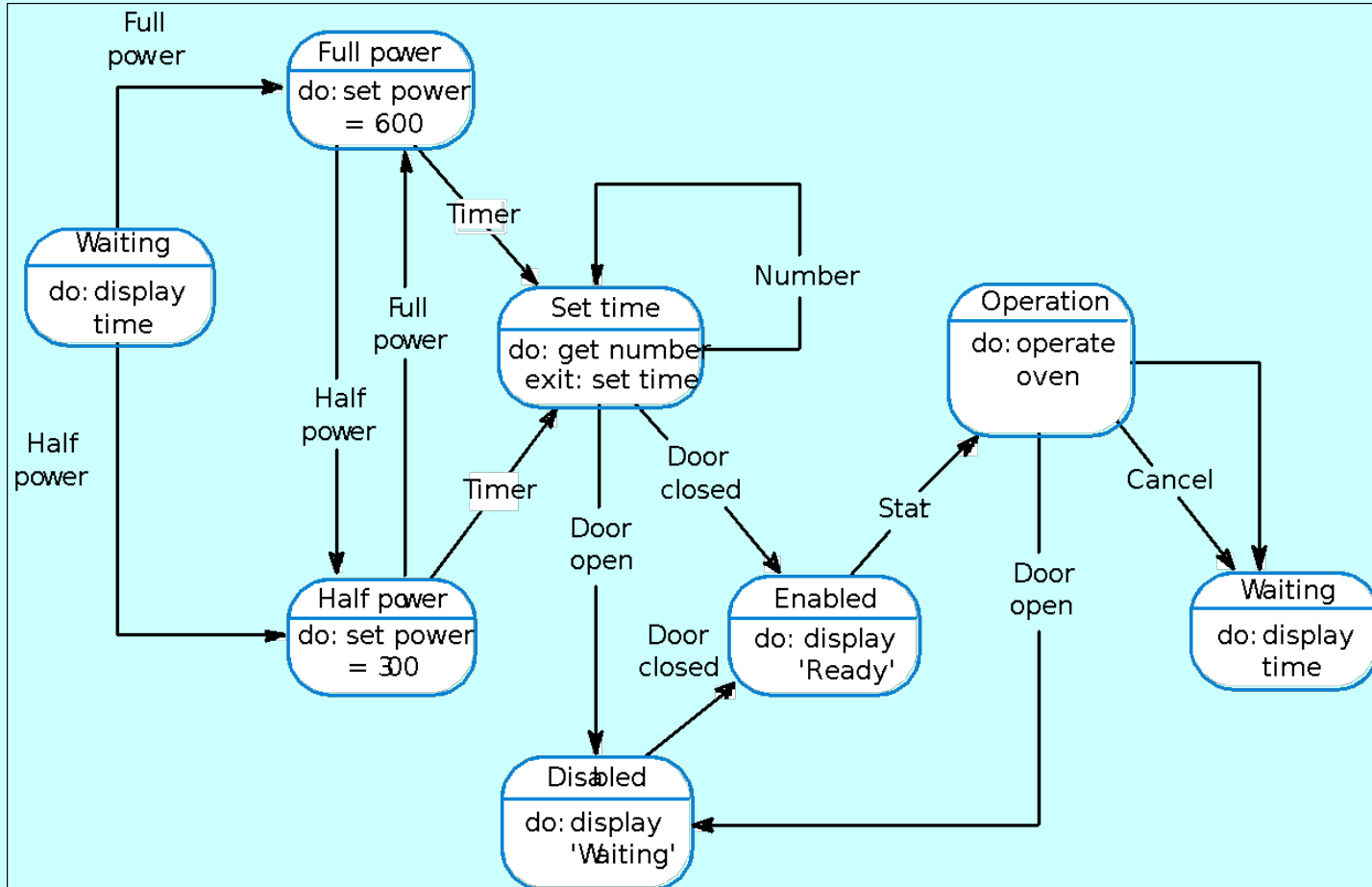
- These model the behaviour of the system in response to external and internal events.
- They show the system's responses to stimuli so are often used for modelling real-time systems.
- State machine models show system states as nodes and events as arcs between these nodes. When an event occurs, the system moves from one state to another.
- Statecharts are an integral part of the UML and are used to represent state machine models.

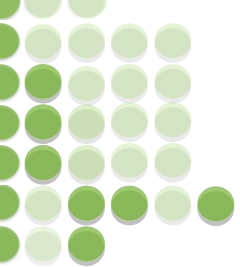


# Statecharts

- 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 model





# Microwave oven state description

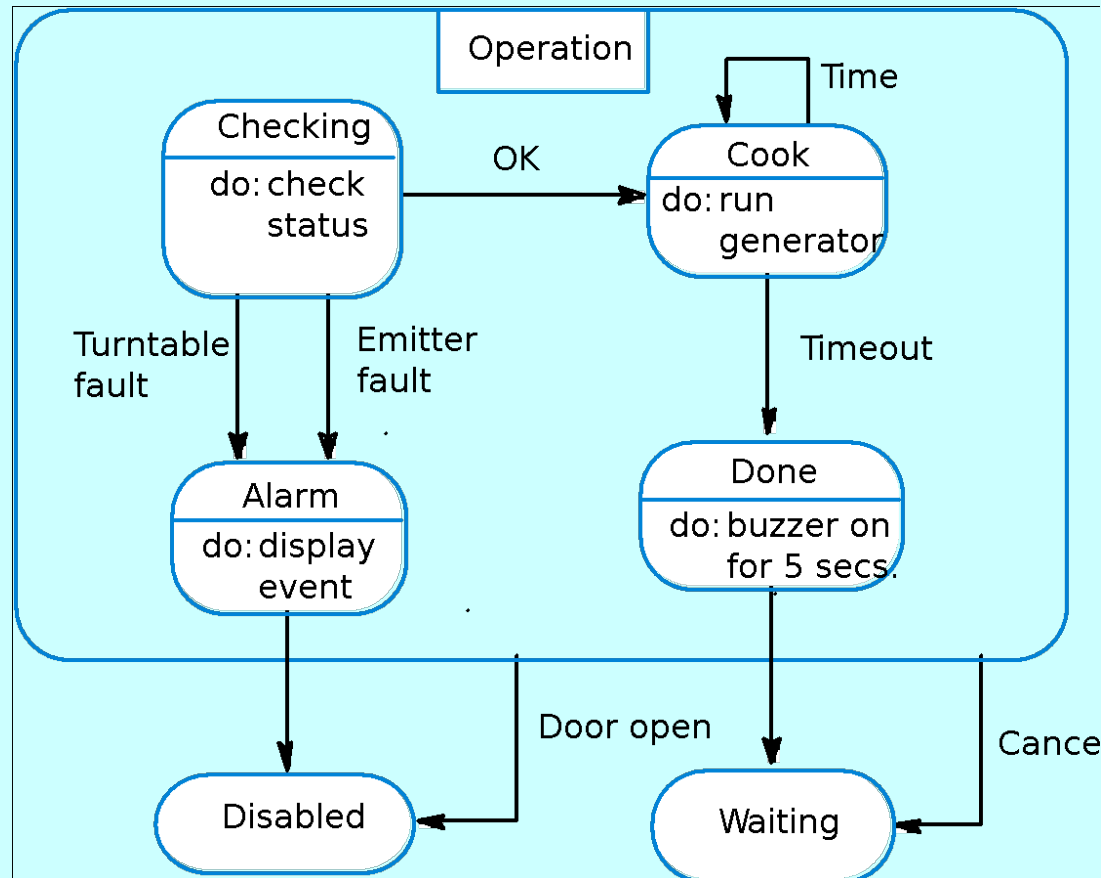
State	Description
Waiting	The oven is waiting for input. The display shows the current time.
Half power	The oven power is set to 300 watts. The display shows "Half power"
Full power	The oven power is set to 600 watts. The display shows "Full power"
Set time	The cooking time is set to the user's input 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.



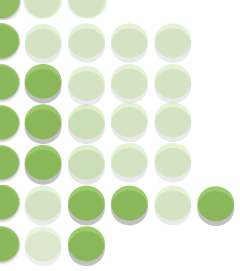
# Microwave oven stimuli

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

# Microwave oven operation







# Key points

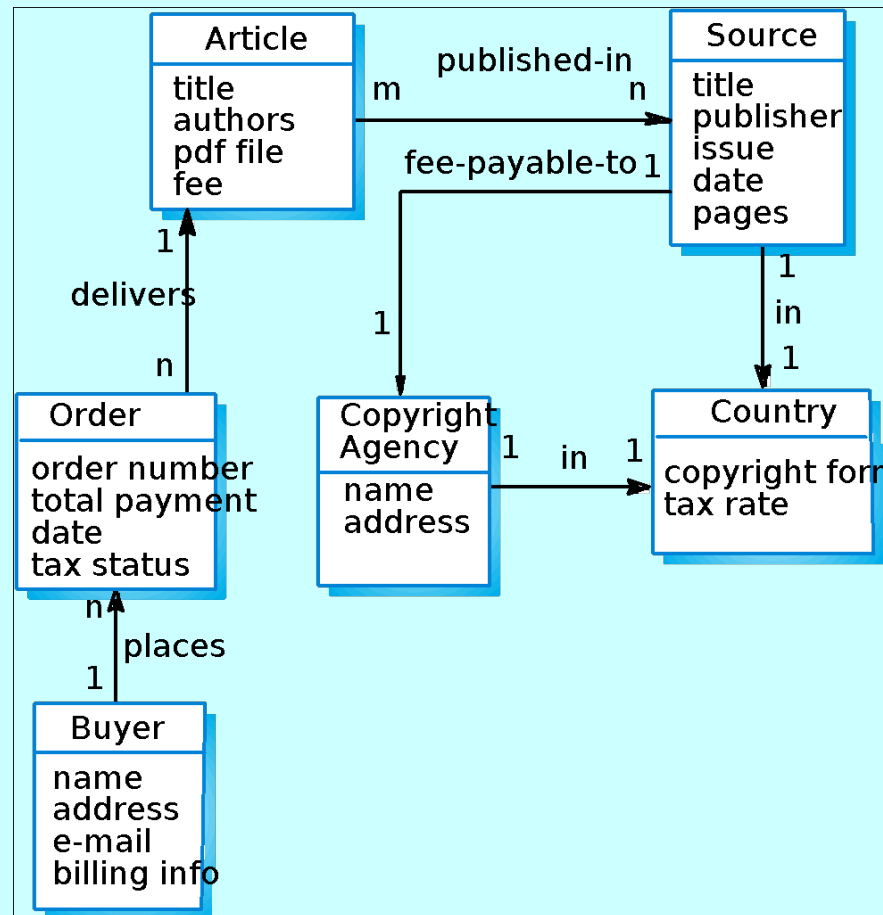
- A model is an abstract system view. Complementary types of model provide different system information.
- Context models show the position of a system in its environment with other systems and processes.
- Data flow models may be used to model the data processing in a system.
- State machine models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

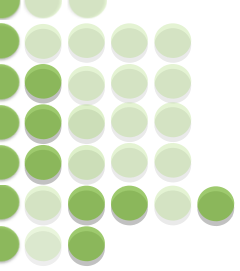


# Semantic data models

- Used to describe the logical structure of data processed by the system.
- An 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.
- In the UML, objects and associations may be used for semantic data modelling.

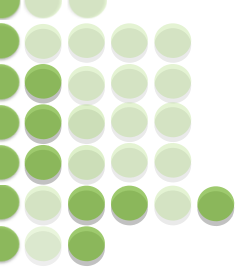
# Library semantic model





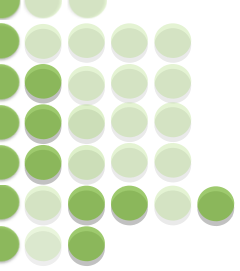
# Data dictionaries

- 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;
- CASE tools can sometimes automatically generate data dictionaries from system models.



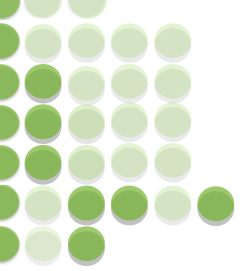
# Data dictionary entries

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 copy 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



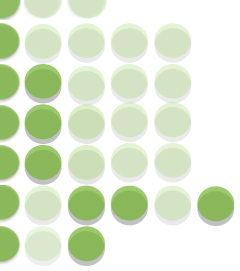
# Object models

- Object models describe the system in terms of object classes and their associations.
- An object class is an abstraction over a set of objects with common attributes and the services (operations) provided by each object.
- Various object models may be produced
  - Inheritance models;
  - Aggregation models;
  - Interaction models.



# Object models

- Natural ways of reflecting the real-world entities manipulated by the system
- More abstract entities are more difficult to model using this approach
- Object class identification is recognised as a difficult process requiring a deep understanding of the application domain
- Object classes reflecting domain entities are reusable across systems



# 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.
- Class hierarchy design can be a difficult process if duplication in different branches is to be avoided.

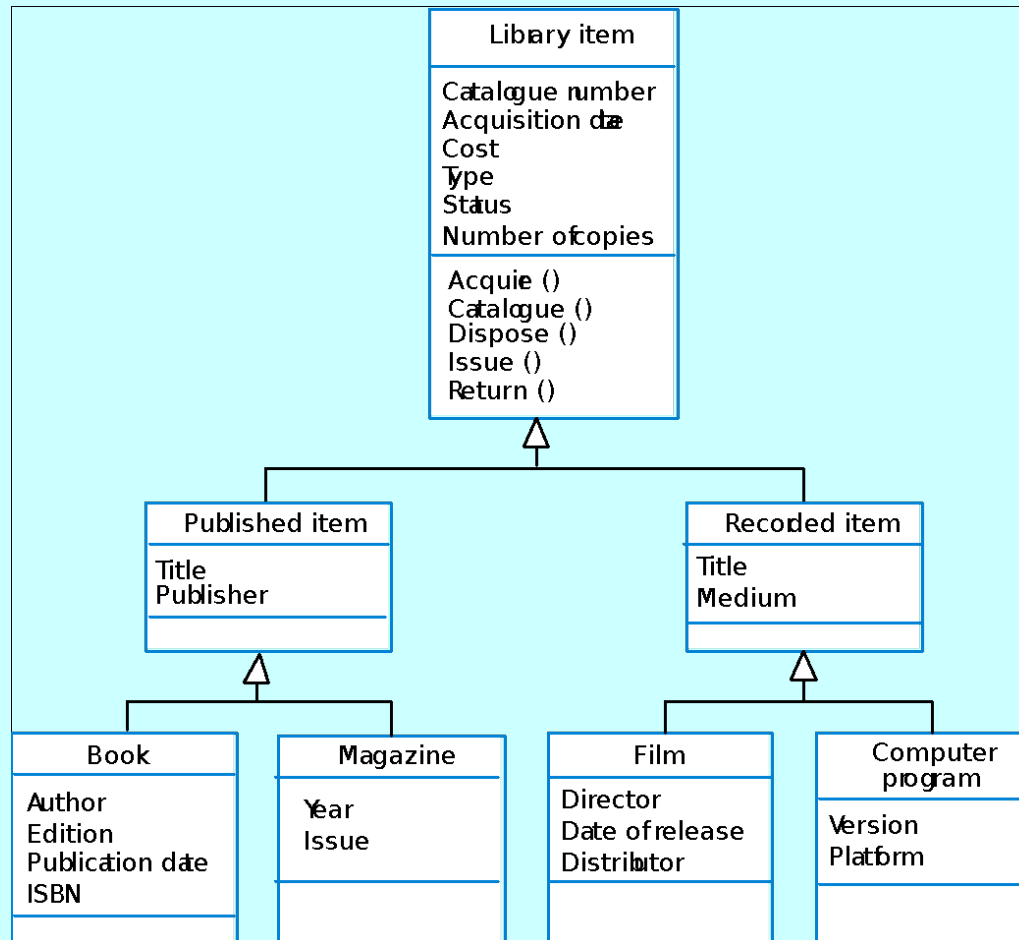




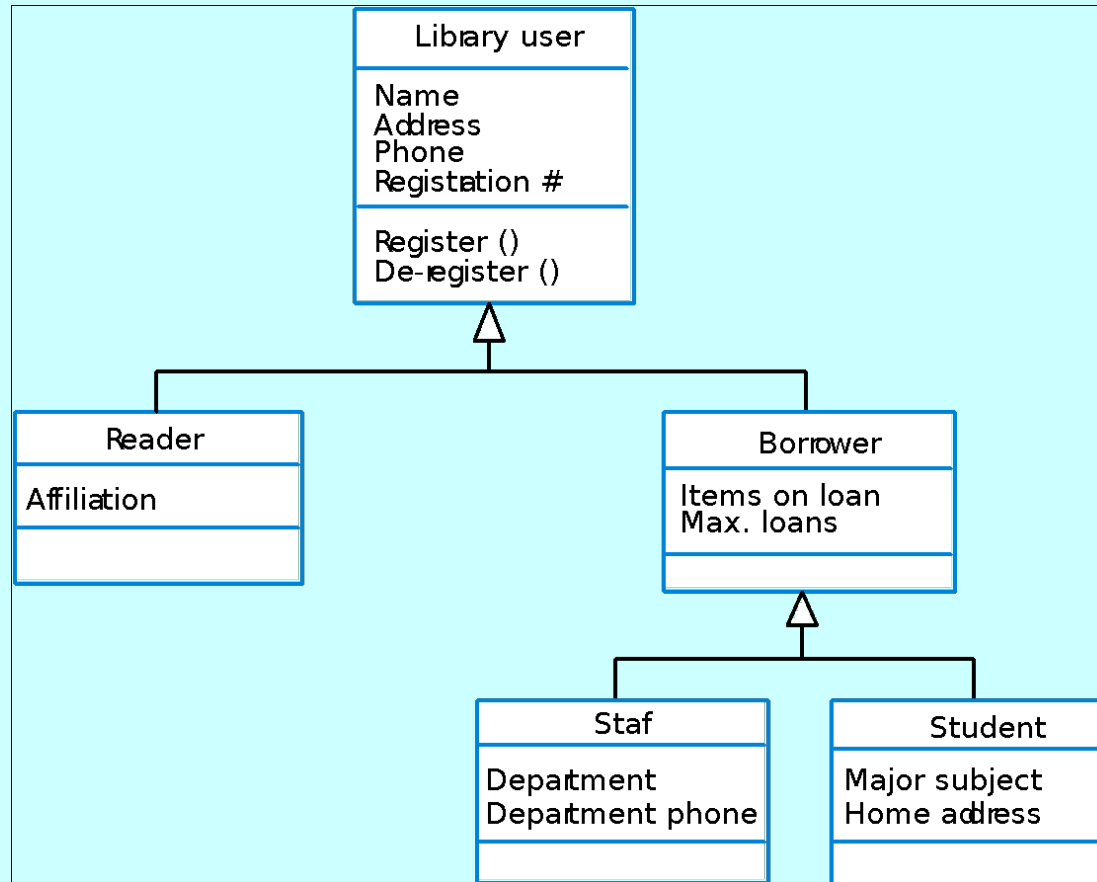
# UML notation

- 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. Associations may be annotated with the type of association or special arrow symbols may be used to denote the association type (e.g. inheritance is a triangular white arrowhead)
- Inheritance is referred to as generalisation and is shown 'upwards' rather than 'downwards' in a hierarchy.

# Library class hierarchy



# User class hierarchy

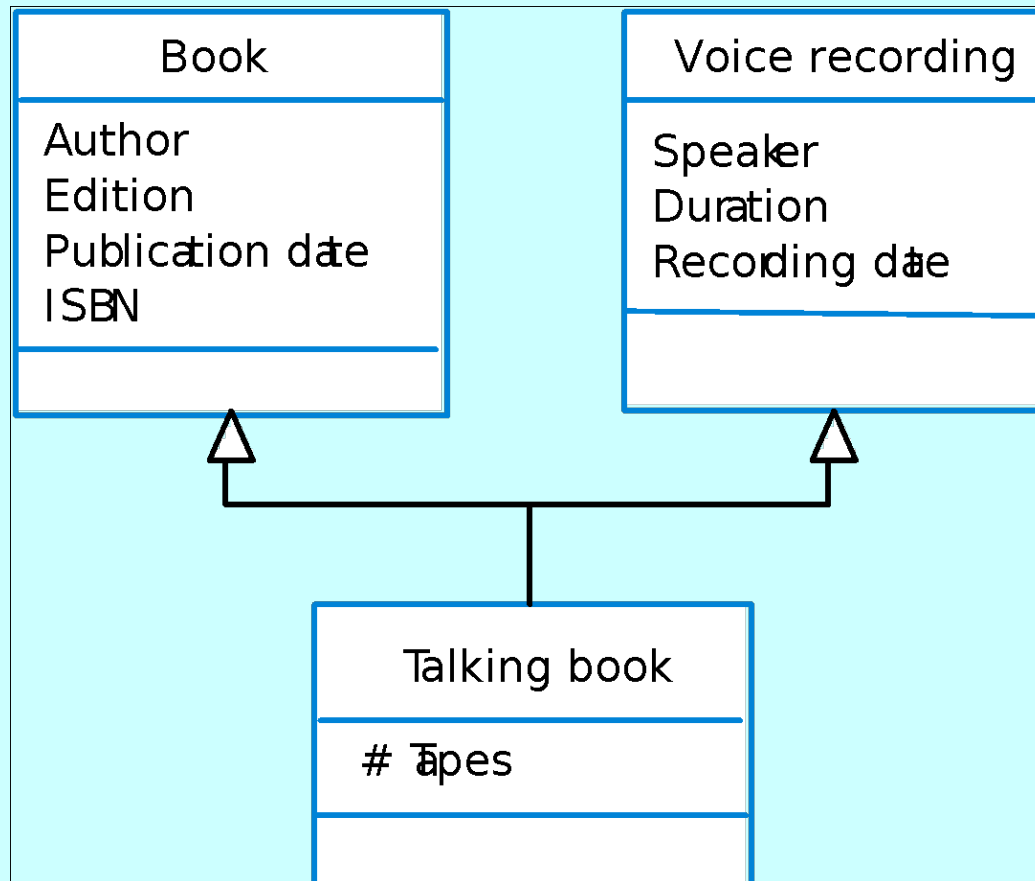


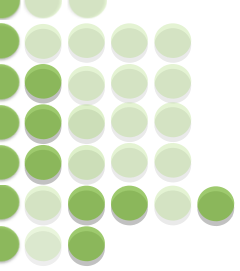


# Multiple inheritance

- 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.
- This can lead to semantic conflicts where attributes/services with the same name in different super-classes have different semantics.
- Multiple inheritance makes class hierarchy reorganisation more complex.

# Multiple inheritance

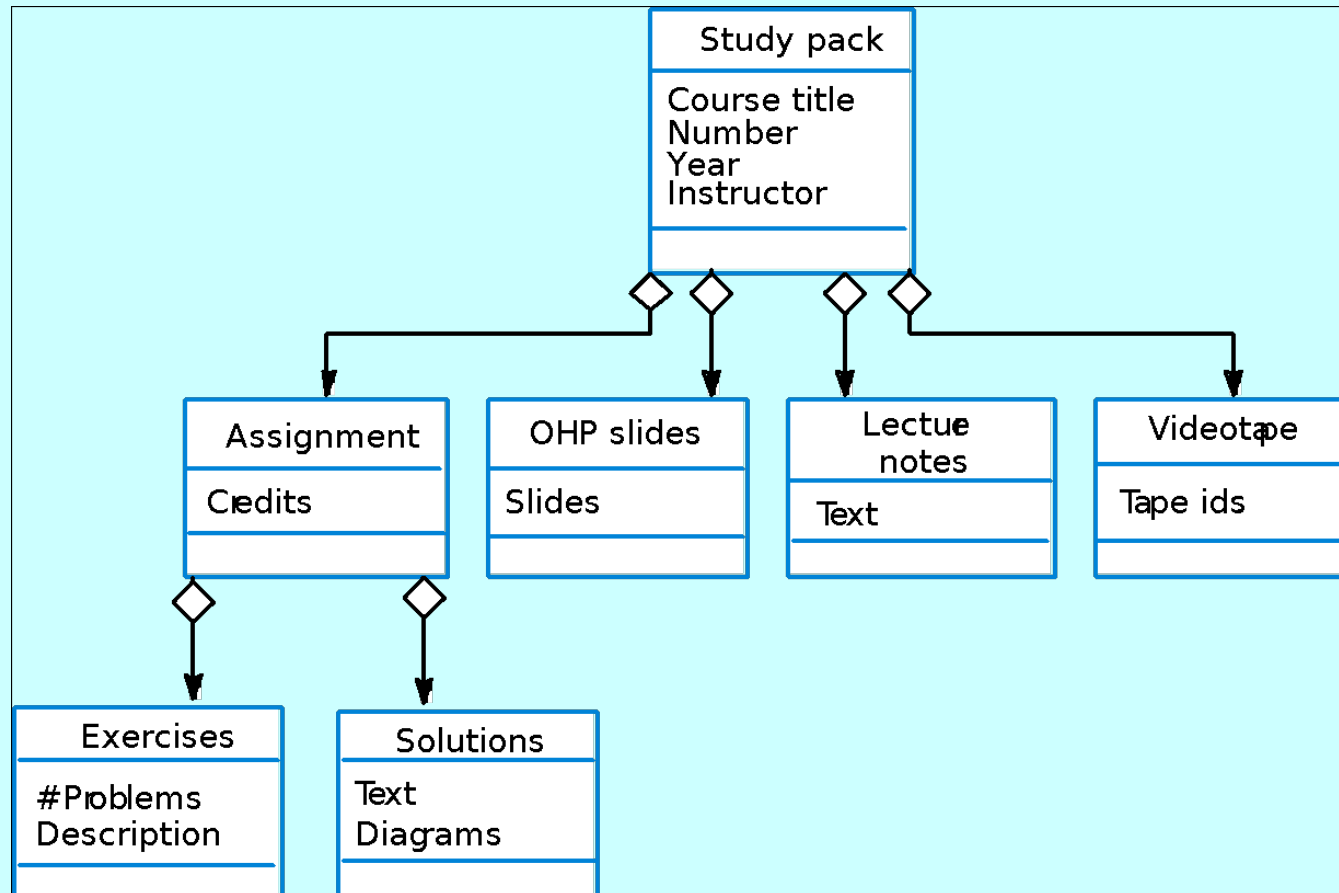


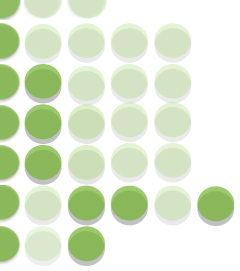


# Object aggregation

- An aggregation model shows how classes that are collections are composed of other classes.
- Aggregation models are similar to the part-of relationship in semantic data models.

# Object aggregation



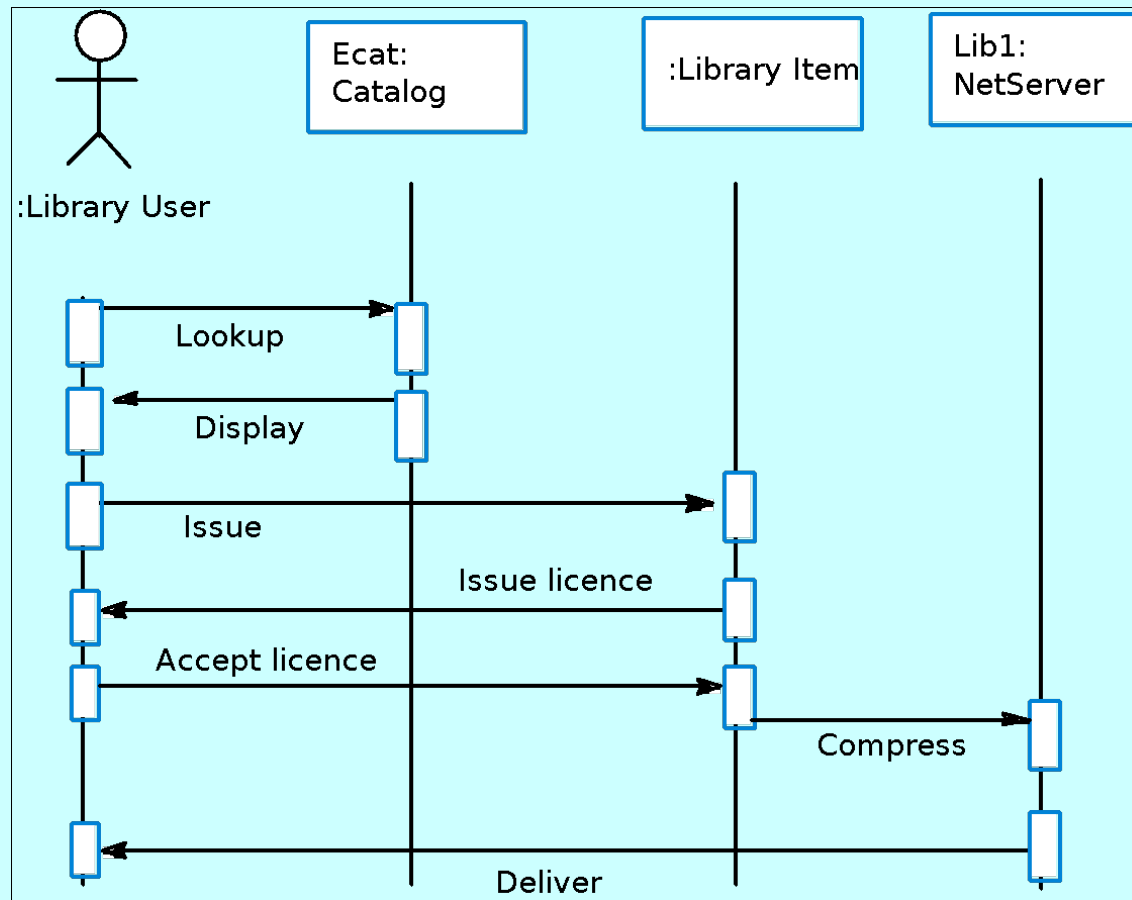


# Object behaviour modelling

- A behavioural model shows the interactions between objects to produce some particular system behaviour that is specified as a use-case.
- Sequence diagrams (or collaboration diagrams) in the UML are used to model interaction between objects.



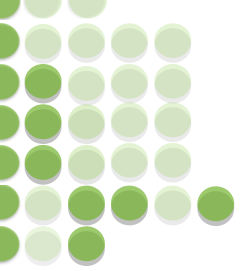
# Issue of electronic items





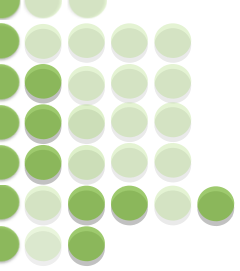
# Structured analysis and design

- Structured analysis and design methods incorporate system modelling as an inherent part of the method.
- Methods define a set of models, a process for deriving these models and rules and guidelines that should apply to the models.
- CASE tools support system modelling as part of the structured method.



# Method weaknesses

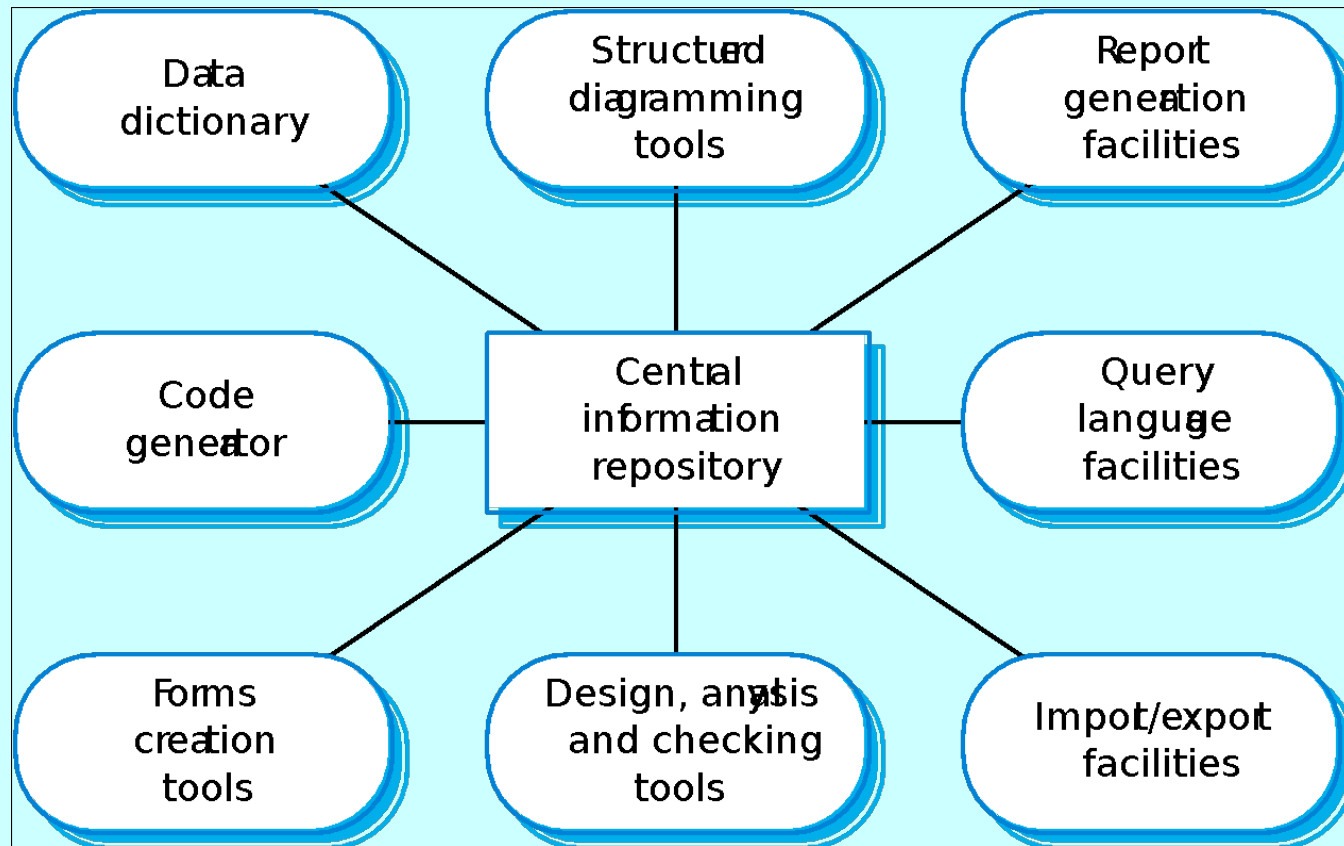
- They do not model non-functional system requirements.
- They do not usually include information about whether a method is appropriate for a given problem.
- They may produce too much documentation.
- The system models are sometimes too detailed and difficult for users to understand.



# CASE workbenches

- A coherent set of tools that is designed to support related software process activities such as analysis, design or testing. May be incorporated in an IDE such as ECLIPSE.
- Support system modelling during both requirements engineering and system design.
- UML workbenches provide support for all UML diagrams and, usually, some automatic code generation. Java, C++ or C# may be generated.

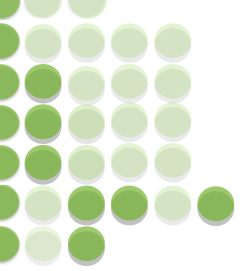
# An analysis and design workbook





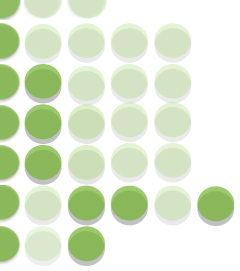
# Analysis workbench components

- 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



# Summary

- 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.
- Sequence models show the interactions between actors and the system objects that they use.
- Structured methods provide a framework for developing system models.



EDUCATION FOR A BETTER LIFE



EDUCATION FOR A BETTER LIFE