Chapter 7. System models

- Abstract descriptions of systems whose requirements are being analysed

Objectives

- To explain why it is important to model the context of a system
- To describe the concepts of behavioural modelling, data modelling and object modelling
- To introduce some of the notations used in UML and how to use it for developing system model
- To show how CASE workbenches support system modelling

Topics covered

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

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

Structured methods

- Structured 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 a structured method(model editor, automate system documentation, some model checking)

Method weaknesses

Structured analysis methods have 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
- The may produce too much documentation➔Essence of the system requirement may be hidden
- The system models are sometimes too detailed and difficult for users to understand

Model types

System model types may be produced during analysis process:

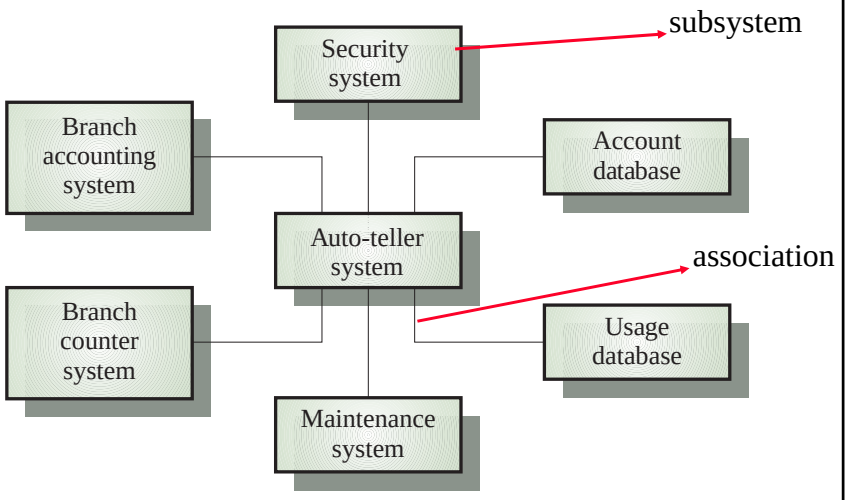
- **Data processing model** showing how the data is processed at different stages
- **Composition model** showing how entities are composed of other entities in an E-R diagram
- **Architectural model** showing principal sub-systems
- **Classification model** showing how entities have common characteristics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

Context models

- Context models are used to illustrate the boundaries of a system
- Social and organisational concerns may affect the decision on where to position system boundaries

  ➔ may be determined by non-technical factors
- Architectural models show the a system and its relationship with other systems
- High-level architectural models are expressed as simple block diagrams. Each subsystem is represented by a named rectangle. Lines indicate the association between sub-systems

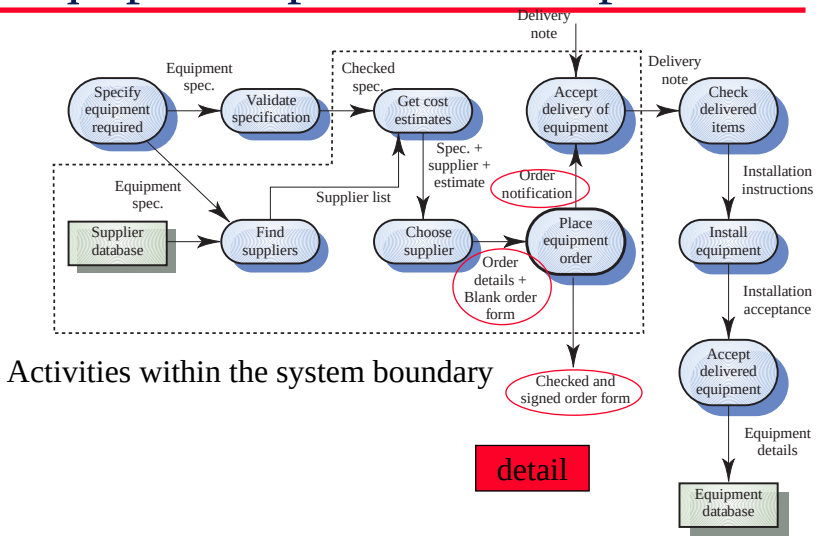
The context of an ATM system



Process models

- Simple architectural models are supplemented by process model to show the process activities and supports by data flow model
- 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

Equipment procurement process



Activities within the system boundary

detail

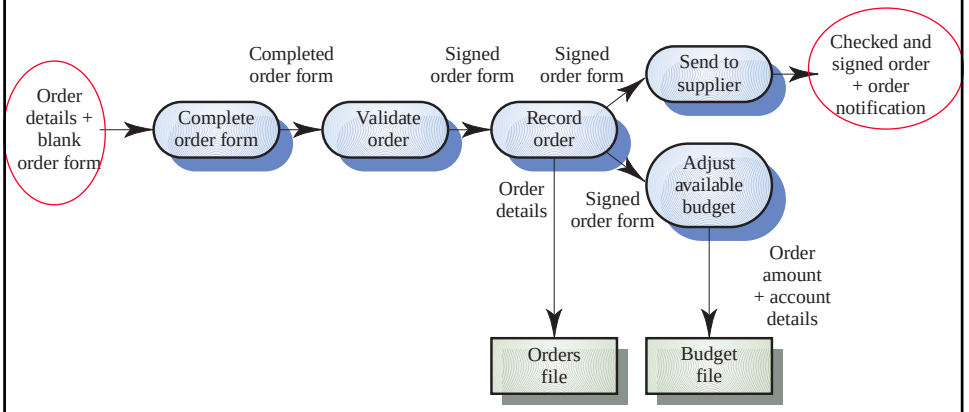
Behavioural models

- Behavioural models are used to describe the overall behaviour of a system
- Two types of behavioural model are shown here
  - Data processing models that show how data is processed as it moves through the system(**data flow**)
  - State machine models that show the systems response to events(**control flow**)
- Both of these models are required for a description of the system's behaviour

Data-processing models

- **Data flow diagrams** are used to model the system's data processing (Yourdan/Demarco or Gane/Sarson 1978)
- These show the processing steps as data flows through a system
- Intrinsic part of many analysis methods
- Simple and intuitive notation that customers can understand
- Show end-to-end processing of data

Order processing DFD



back

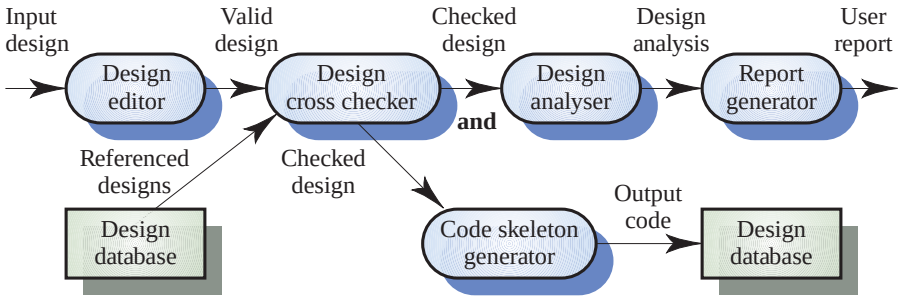
Data flow diagrams

- 3 basic graphical components for representing DFD:
  - Rounded rectangle for **processing step or subsystem**
  - Arrow with data name for **data flow**
  - Rectangle for **data stores or data sources**
- 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

Data Flow Diagram

- Top-down design process
- Information about system is normally acquired about several levels at the same time
- Lower-level models may be developed first and then to create a more general model
- Data flow model can show how different systems and subsystems exchange information

CASE toolset DFD



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
- Harel's Statecharts are an integral part of the UML
- A stimulus can trigger a transition from one state to another state

Microwave oven operation

- Select the power level(Full or Half)
- Input the cook time
- Press start button and the food is cooked for the given time

➔Rounded rectangle represents the system states

➔Labeled arrow represents stimuli which force the system to change state

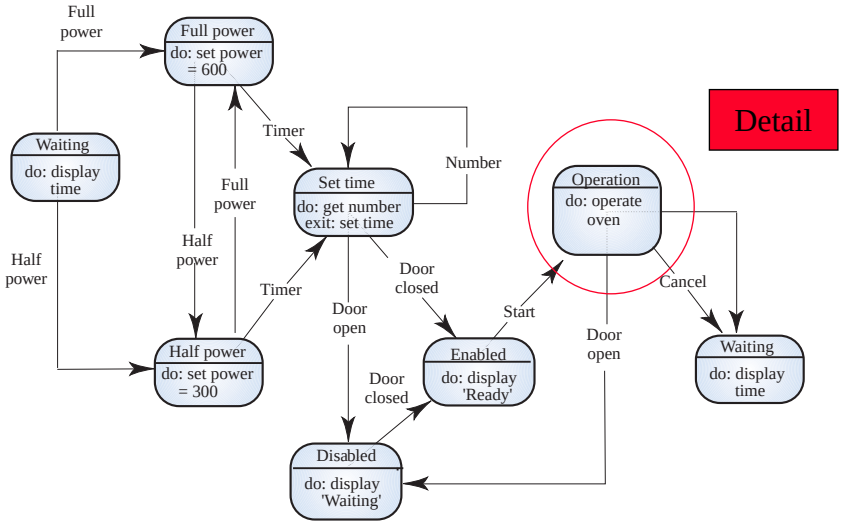
Microwave oven state description

	State	Description
1	Waiting	The oven is waiting for input. The display shows the current time.
2	Half power	The oven power is set to 300 watts. The display shows alf power'.
3	Full power	The oven power is set to 600 watts. The display shows ull power'.
4	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.
5	Disabled	Oven operation is disabled for safety. Interior oven light is on. Display shows 'Not ready'.
6	Enabled	Oven operation is enabled. Interior oven light is off. Display shows 'Ready to cook'.
7	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?w hile buzzer is sounding.

Microwave oven stimuli

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

Microwave oven model



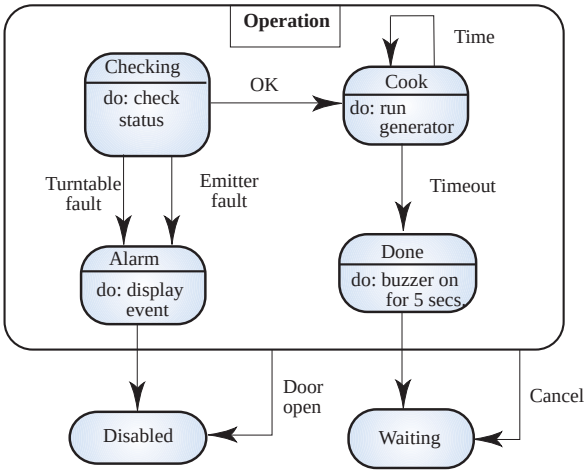
State machine table for microwave oven

Stimulus / State	1	2	3	4	5	6	7	8
1	2	3						
2		3	4					
3	2		4					
4				4	5	6		
5						6		
6							7	
7					5		1	1

Statecharts

- Allow the decomposition of a model into sub-models
- 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
- For large system model, a super-state which encapsulates a number of separate states is used for structuring the states structure

Microwave oven operation

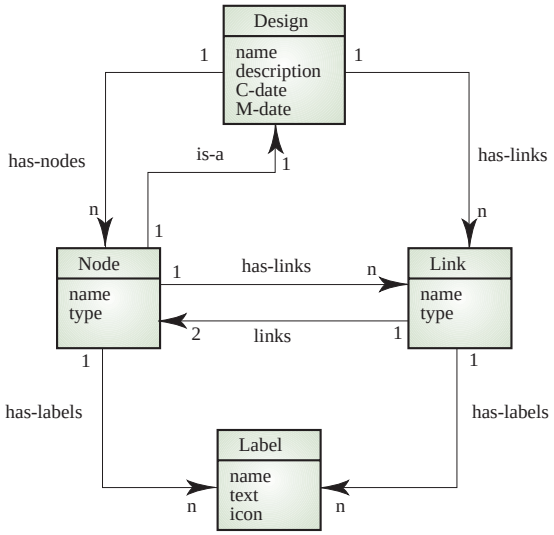


back

Semantic data models

- Used to describe the **logical structure of data** processed by the system➔describe the db structure
- **Entity-relation-attribute** model sets out the entities in the system, the **relationships** between these entities and the **associated attributes**('76 Chen)
- Widely used in database design. Can readily be implemented using relational databases in **3NF**
- Can also be used to implement **OODB**
- No specific notation provided in the UML but **objects** and **their relationships** can be used

Software design semantic model



DD detail

Software design semantic model

- Designs are **directed graph** that is consisted of a set of **nodes** of different types connected by **links** represented the **relationships** between design nodes
- Process the **entities**, the **logical attributes** and their **relationship**
- ERA model **lacks detail description** and should be used to supplement the information which are kept in the **data dictionary** or **repository**

Data dictionaries

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

Data dictionary entries

back

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 name 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** may be used to represent both the **system data** and their processing
- An **object class** is an abstraction over a set of **objects** with common **attributes** and the **services**(operations) provided by each object
- Object models simplify the transition from OOA to OOD and OOP but should be supplemented by the **data-flow models** to show the end-to-end **data processing** in the system

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

Object models

Various object models may be produced

- Inheritance models
- Aggregation models
- Interaction models

- Object models focus how objects can be classified and inherit attributes & operations from other objects
- Aggregation models show how objects are composed
- Simple behaviour models show object interaction

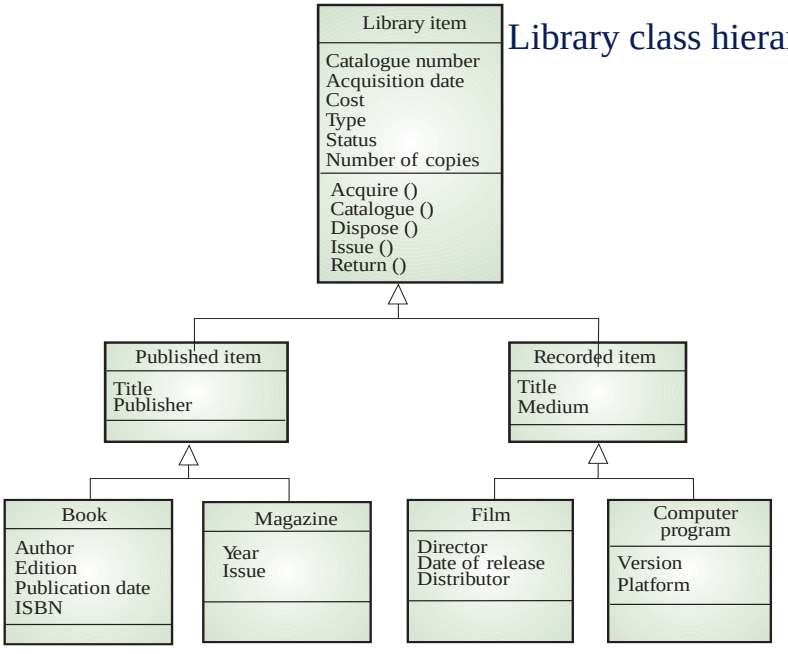
Inheritance models

- 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** is a difficult process if duplication in different branches is to be avoided

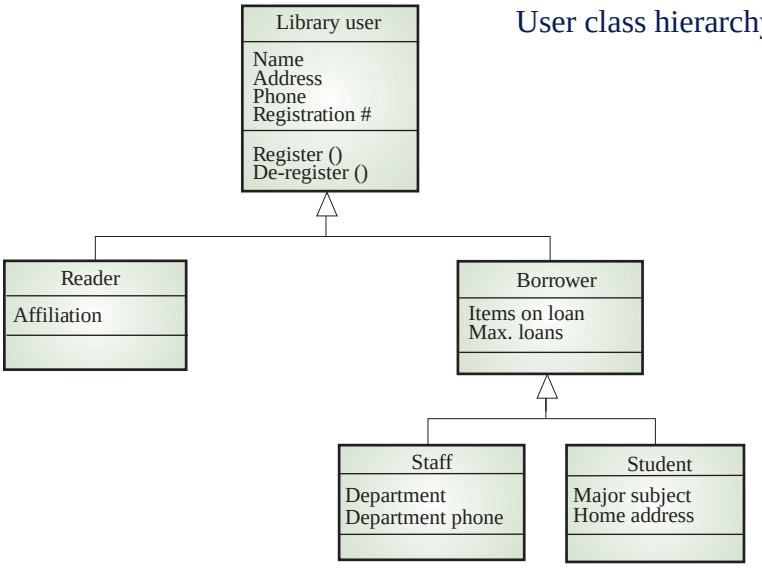
The Unified Modelling Language

- 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 relationship** and is shown '**upwards**' rather than 'downwards' in a hierarchy

Library class hierarchy



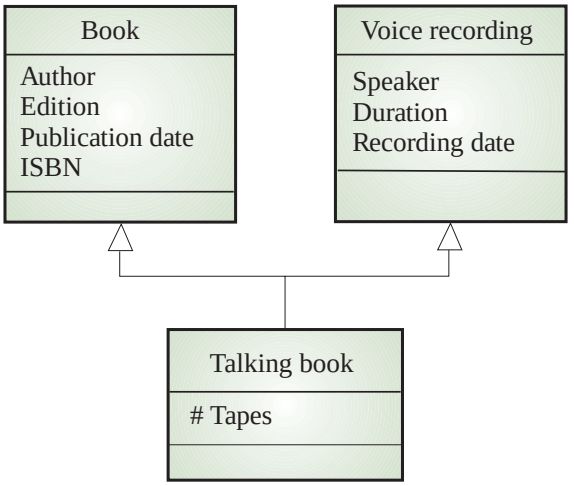
User class hierarchy



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
- Design an **inheritance graph** to show the object structure
- Can lead to **semantic conflicts** where **attributes/services** with the same name with different meanings in different super-classes
- Makes class hierarchy reorganisation more complex

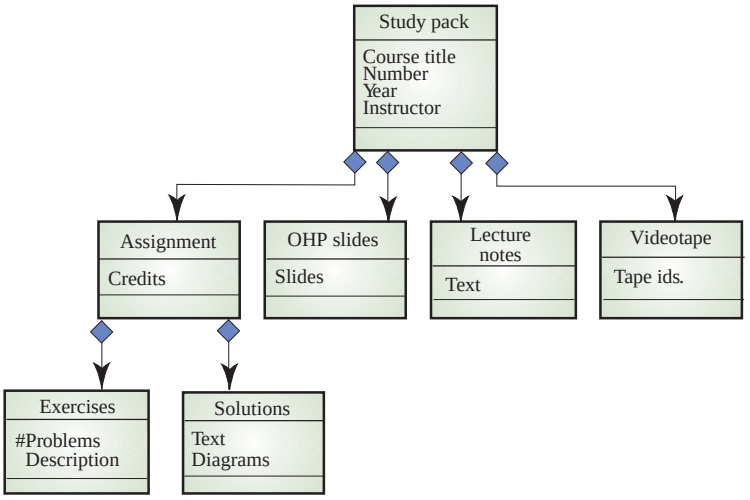
Multiple inheritance



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
- Relationship:
  - Has-a (attribute)
  - Part-of (aggregation objects)
  - Is-a (inheritance)

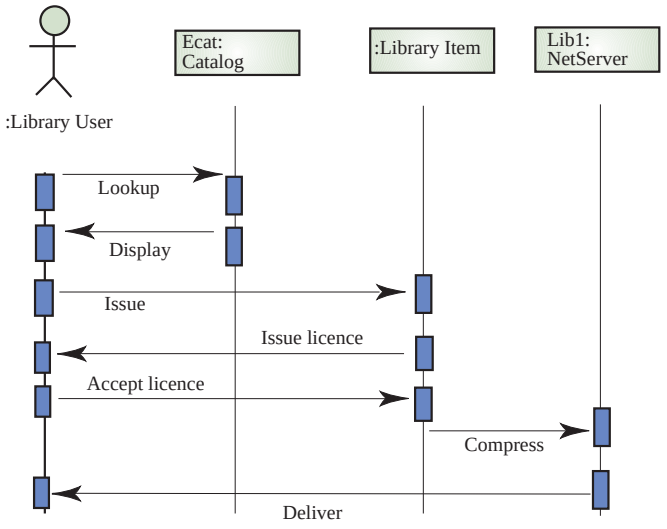
Object aggregation



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

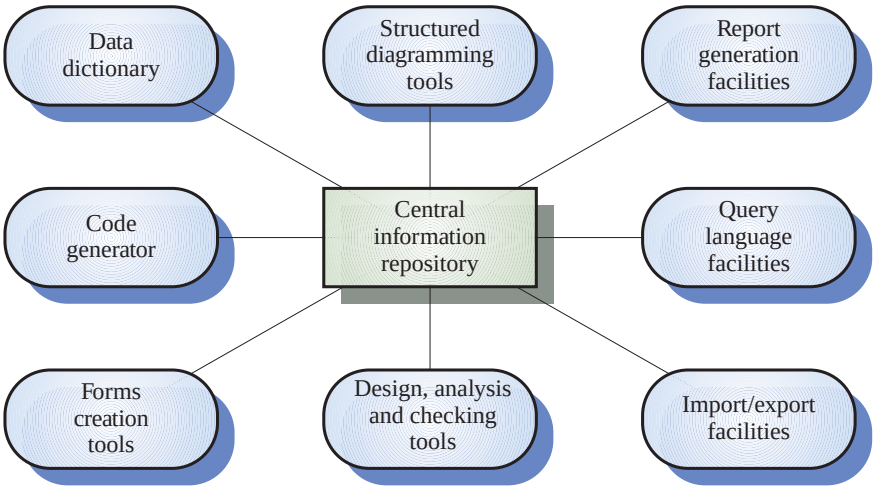
Issue of electronic items



CASE workbenches

- A coherent set of tools that is designed to support related software process activities such as analysis, design or testing
- Analysis and design workbenches support system modelling during both requirements engineering and system design
- These workbenches may support a specific design method or may provide support for a creating several different types of system model

An analysis and design workbench



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

CASE Workbenches

- Tools are integrated through a **shared repository** which structure is proprietary to the **workbench vendor**
- It is difficult for users to **add their own tools** to modify the tools that are provided by workbench vendor

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
- CASE workbenches support the development of system models

HomeWork

- 7.2
- 7.3
- Draw your project'd DFD into at least 3 levels
- 7.4
- 7.7
- Draw your project's state diagram at least 3 levels