System Models and System Specification

Beatrice Åkerblom
beatrice@dsv.su.se

System Models

"If a picture is worth a thousand words then a model is often worth 1024 lines of code when applied in the right circumstances"

Karl Wieger
(Software Requirements)

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

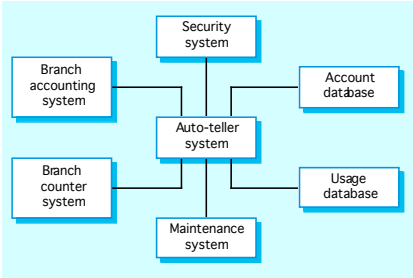
Model Types

- 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

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

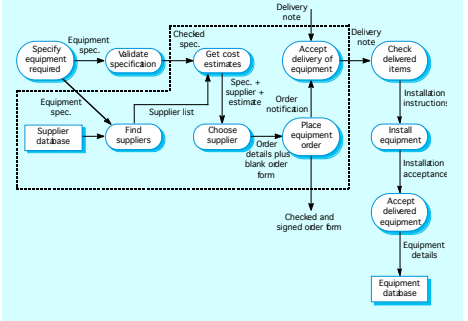
The Context of an ATM System



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

Equipment Procurement Process



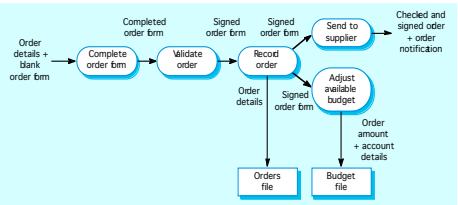
Behavioural Models

- 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

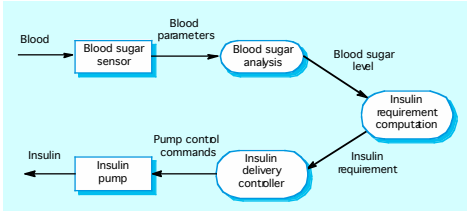
Order Processing Data Flow Diagram



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



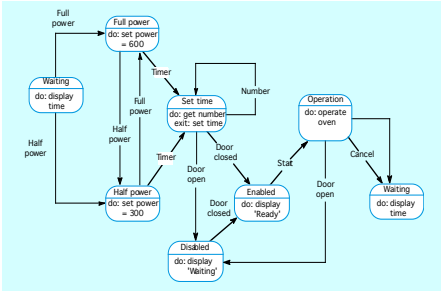
State Machine Models

- 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
- State-charts are an integral part of the UML and are used to represent state machine models

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 Model



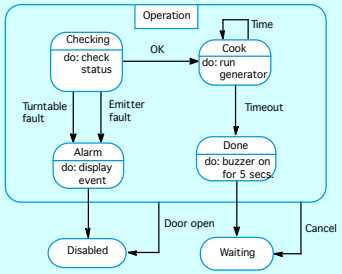
Microwave Oven State Description

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

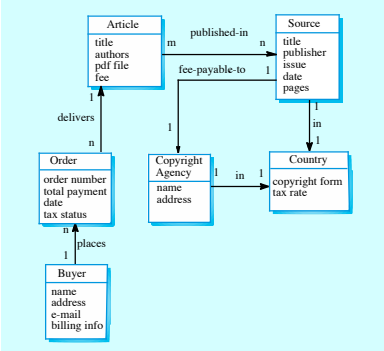
Microwave Oven Operation



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
- No specific notation provided in the UML but objects and associations can be used

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

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 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

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

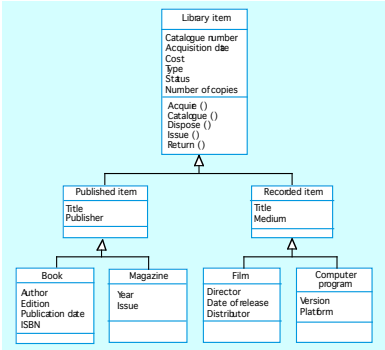
Inheritance Models

- 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

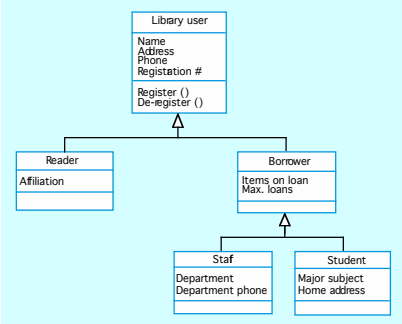
Object Models and the UML

- The UML is a standard representation devised by the developers of widely used object-oriented analysis and design methods
- It 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



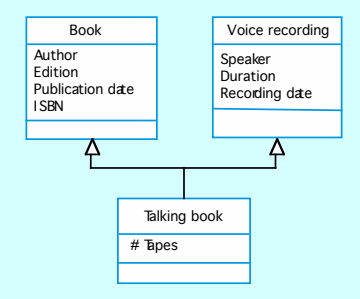
User Class Hierarchy



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
- 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

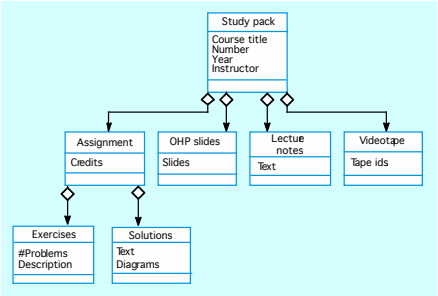
Multiple Inheritance



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

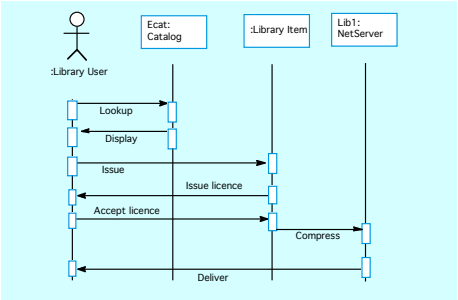
Object Aggregation



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 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

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
- The may produce too much documentation
- The system models are sometimes too detailed and difficult for users to understand

Challenges in System Specification

- What kinds of System Models should we use?
  - What kind of system are we developing?
  - Who are involved in the work, who do we need to communicate with?
- On what level of abstraction should we create our System Models?
  - What should they be used for?

Agile Modelling

"real XP:ers don't do diagrams"

an eXtreme misUnderstanding (XU)

Active Stake-holder Participation

- An expansion of XP's On-Site Customer
- Agile Modelling expands XP's On-Site Customer practice to have project stake-holders actively involved in the project

Apply The Right Artefact(s)

- A UML activity diagram is useful for describing a business process
- The static structure of your database is better represented by a physical data or persistence model
- Very often a diagram is a better choice than source code -- design alternatives can be explored more effectively by drawing diagrams than by developing code samples
- The implication is that you need to know the strengths and weaknesses of each type of artefact so you know when and when not to use them

Collective Ownership

- Everyone can work on any model, and any artefact on the project

Create Several Models in Parallel

- Each type of model has its strengths and weaknesses -- no single model is sufficient for your modelling needs
- An iterative approach with several models is often more productive than focusing on one at a time

Create Simple Content

- You should keep the actual content of your models -- your requirements, your analysis, your architecture, or your design -- as simple as you possibly can while still fulfilling the needs of your project stake-holders

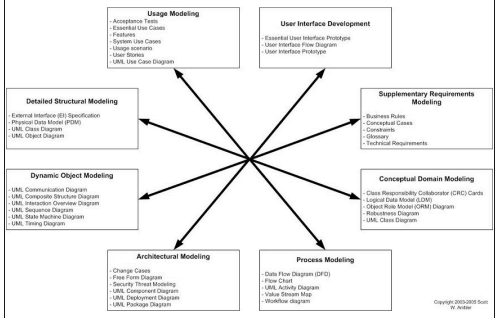
Depict Models Simply

- UML and other description “languages” provide notations for describing many different things -- most of the time you could do with just a small subset of the possibilities
- A simple model that shows the key features that you are trying to understand often proves to be sufficient.

Display Models Publicly

- Models should be displayed publicly to support open and honest communication in your team
- The modelling wall should be accessible to your development team and other project stake-holders.

Iterate to Another Artefact



- Usage
- Process
- User interface
- Supplementary requirements
- Conceptual domain
- Architectural
- Dynamic design
- Structural design

Agile Modelling -- Core Practices

- Active Stake-holder Participation
- Apply the Right Artefact(s)
- Collective Ownership
- Create Several Models in Parallel
- Create Simple Content
- Depict Models Simply
- Display Models Publicly
- Iterate to Another Artefact
- Model in Small Increments
- Model With Others
- Prove it With Code
- Single Source Information
- Use the Simplest Tools

Agile Modelling -- Supplementary Practices

- Apply Modelling Standards
- Apply Patterns Gently
- Discard Temporary Models
- Formalise Contract Models
- Update Only When It Hurts

Is Design Dead? by Martin Fowler

- Evolutionary design -- over time the design gets worse and worse
- Planned design -- the engineering approach
  - It's impossible to think through all the issues that you need to deal with when you are programming
  - Cultural differences
- XP advocates evolutionary design rather than planned design

Is Design Dead? by Martin Fowler, cont'd

- Testing and Continuous Integration
- Balance between up-front design and refactoring
- YAGNI -- "You Aren't Going to Need It"
- Simplicity
  - Runs all the Tests
  - Reveals all the intention
  - No duplication
  - Fewest number of classes or methods
- Quick design sessions
  - keep them short
  - don't address all details
  - treat the design as a sketch, not as a final design

Is Design Dead? by Martin Fowler, cont'd

- No, but the nature of design has changed. XP design looks for the following skills
  - A constant desire to keep code as clear and simple as possible
  - Refactoring skills so you can make improvements whenever you see the need
  - A good knowledge of patterns: not just the solutions but also when to use them and how to evolve into them
  - Designing with an eye to future changes
  - Knowing how to communicate the design to the people who need to understand it, using code, diagrams and above all: conversation.

End of Today's Lecture

Thanks for your attention!