Chapter 7

System Models
Abstract descriptions of systems whose requirements are being analysed

Model

Model is an abstraction of a system aimed at simplifying the reasoning about the system by omitting irrelevant details.

Modeling

An activity during which participants construct an abstraction of a system by focusing on interesting aspects and omitting irrelevant details.

What is interesting or irrelevant depends on the task in which the model is used.

Topics covered

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

System models

- The most important aspect of a system model is that it leaves out detail.
- A system model is an abstraction (instead of an alternative representation) of the system being studied.

System modelling

- Models are used to communicate with stakeholders.
- 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
Examples of system model

- 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

- A context model is used to illustrate the boundaries of a system. It shows the system and its relationship with other systems.
- Social and organizational concerns may affect the decision on where to position system boundaries

The context of an ATM system

![ATM system context diagram]

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

Behavioural models

- They are used to describe the overall behaviour of a system
- Two types of behavioural model:
  - Data flow models
  - State machine models
- Both of these models are useful in describing the system’s behaviour
### Data-flow models

- Data flow diagrams are used to model the system’s data processing. They show the processing steps involved as data flow through a system.

### 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

### CASE toolset DFD

#### State-machine models

A state-machine model is useful in describing how a system responds to a sequence of inputs or stimuli.

### 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 reactive software 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.
Microwave oven state description

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waiting</td>
<td>The oven is waiting for input. The display shows the current time.</td>
</tr>
<tr>
<td>Full power</td>
<td>The oven power is set to 600 watts. The display shows 'Full power'.</td>
</tr>
<tr>
<td>Half power</td>
<td>The oven power is set to 300 watts. The display shows 'Half power'.</td>
</tr>
<tr>
<td>Set time</td>
<td>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.</td>
</tr>
<tr>
<td>Disabled</td>
<td>Oven operation is disabled for safety. Interior oven light is off. Display shows 'Not ready'.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Oven operation is enabled. Interior oven light is on. Display shows 'Ready to cook'.</td>
</tr>
<tr>
<td>Operation</td>
<td>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.</td>
</tr>
</tbody>
</table>

Microwave oven stimuli

<table>
<thead>
<tr>
<th>Stimulus</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Half power</td>
<td>The user has pressed the half power button</td>
</tr>
<tr>
<td>Full power</td>
<td>The user has pressed the full power button</td>
</tr>
<tr>
<td>Timer</td>
<td>The user has pressed one of the timer buttons</td>
</tr>
<tr>
<td>Number</td>
<td>The user has pressed a numeric key</td>
</tr>
<tr>
<td>Door open</td>
<td>The oven door switch is not closed</td>
</tr>
<tr>
<td>Door closed</td>
<td>The oven door switch is closed</td>
</tr>
<tr>
<td>Start</td>
<td>The user has pressed the start button</td>
</tr>
<tr>
<td>Cancel</td>
<td>The user has pressed the cancel button</td>
</tr>
</tbody>
</table>

State charts

The descriptive power of a state-transition diagram is rather limited. The language of state chart is an extension of that of state-transition diagram.
Microwave oven operation

- Operation
  - Check: do check status, turntable fault, door display event, buzzer on for 5 secs.
  - Cook: do run generator, event, check status, buzzer on, door open, cancel.
- Time
  - Waiting

Data models

- 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

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 organizational knowledge linking analysis, design and implementation
- Many CASE workbenches support data dictionaries

Modeling with UML

- System development focuses on:
  - functional model
  - object model
  - dynamic model
- Five UML notations:
  - Use case diagrams
  - Class diagrams
  - Sequence diagrams
  - Statechart diagrams
  - Activity diagrams
The functional model
represented in UML with use case diagrams, describes the functionality of the system from the user's point of view.

The object model
represented in UML with class diagrams, describes the structure of a system in terms of objects, attributes, associations, and operations.

The dynamic model
represented in UML with sequence diagrams, statechart diagrams, and activity diagrams, describes the internal behavior of the system.

Use case diagrams
Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from an external point of view. A use case describes a function provided by the system that yield a visible result for an actor. An actor describes any entity that interacts with the system.

Example use case diagram

Class diagrams
Class diagrams describes the system in terms of objects, classes, attributes, operations, and their association.
Example class diagram

SimpleWatch

PushButton

Display

Battery

Time

Sequence diagrams

Sequence diagrams are used to formalize the behavior of the system and to visualize the communication among the objects. They are useful for identifying additional objects that participate in the use cases. A sequence diagram represents the interactions that take place among the objects involved in a use case.

Example sequence diagram

Statechart diagrams

Statechart diagrams describe the behavior of an individual object as a number of states and transitions between these states. A state represents a particular set of values for an object.

Example statechart diagram

Activity diagrams

An activity diagram describes a system in terms of activities. Activities are states that represent the execution of a set of operations. The completion of these operations triggers a transition to another activity. They are similar to data-flow diagrams or control-flow diagrams.
Example activity diagram

Examples of data dictionary entries

<table>
<thead>
<tr>
<th>name</th>
<th>descriptions</th>
<th>type</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>has-labels</td>
<td>1:N relation between entities of type Node or Link and that of type Label.</td>
<td>Relation</td>
<td>5/10/04</td>
</tr>
<tr>
<td>Label</td>
<td>Holds info about nodes or links. Labels represented by an icon and text.</td>
<td>Entity</td>
<td>8/12/04</td>
</tr>
<tr>
<td>Link</td>
<td>1:1 relation between design entities represented as nodes.</td>
<td>Entity</td>
<td>12/3/04</td>
</tr>
<tr>
<td>name (label)</td>
<td>Each label has a name that identifies the type of label.</td>
<td>Attribute</td>
<td>12/9/04</td>
</tr>
<tr>
<td>name (node)</td>
<td>Each node has a name that must be unique within a design.</td>
<td>Attribute</td>
<td>1/4/05</td>
</tr>
</tbody>
</table>

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
- 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 recognized as a difficult process requiring a deep understanding of the application domain
- Object classes reflecting domain entities are reusable across systems

Inheritance models

- Organize 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 specialized 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 generalization and is shown ‘upwards’ rather than ‘downwards’ in a 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 reorganization more complex

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 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 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

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 sequences of inputs.
Key points (continued)

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