L6: System design: behavior models

Limitations of functional decomposition

Behavior models
  State diagrams
  Flow charts
  Data flow diagrams
  Entity relationship diagrams
  Unified Modeling Language
Motivation

Functional design

– Functional block diagrams are good for describing systems in terms of
  • Their inputs, outputs, and transforms (function-oriented view)
  • Their hierarchical structure (modules and sub-modules)
– There are other types of system behavior that designers need to be able to understand
  • State behavior
  • Logic and flow
  • Data flow
  • Database relationships
– In this lecture we describe other design tools for describing system behavior, from old-fashion flowcharts to the all-encompassing UML
Behavior models

- **State diagrams**, to describe systems with memory
- **Flow charts**, to describe a process or algorithm
- **Data flow diagrams**, to describe the processing and flow of data
- **Entity relationship diagrams**, which model the objects (entities), their attributes, and their relationships
- **Unified Modeling Language**, a general-purpose tool to model structural and behavioral aspects of a system
State diagrams describe the behavior of systems with memory

- These are systems whose outputs to the same input can vary depending on their internal state

A state diagram consists of three elements

- States, represented as rounded rectangles
- Transitions, represented as arcs
- Initial and final states, represented by filled circles

Example (vending machine)
Flowcharts

Flowcharts describe algorithms, their steps and control

– Considered old-fashioned and simple, these are actual strengths
  • Old-fashioned means they are easily recognizable
  • Simple means they are accessible to a broader audience

– Flowcharts use specific shapes to describe various elements of the algorithm: processes, decisions, and data

– Limitations of flowcharts
  • They cannot represent the structure of the data being manipulated
  • They cannot represent concurrent processes – you need data flow diagrams for this
Example

– Can you figure out what this flowchart does?
Data flow diagrams

DFDs model the processing of data through a system

- Closely related to functional decomposition:
  - The processes in a DFD accept data, transform it, and produce an output
  - DFDs can also be successively refined (Level 0 → Level 1 → Level 2)
  - However, functional decomposition is often closer to the system implementation than DFDs, which offer a data-centric view

- Compared to flow charts, DFDs do not provide any sequencing information, and instead allow multiple concurrent processes

DFD elements

- **Processes**: describe a useful task or function—a data transformation
- **Data flow**: a data relationship between two processes
- **Data stores**: repositories for data
- **Interfaces**: external entities that use the system (sources or sinks)
Example: DFD for a video browsing system

- Video Database
- Shot Boundary Detection
  - Boundaries
  - Key Frames
- Annotation Database
- User
  - Browse Request
  - Storyboard Preview Request
  - Shot Preview Request
  - Shot
- Video
- Storyboard Preview
  - Boundaries
  - Key Frames
- Shot Preview
**Entity relationship diagrams**

**ERDs model the database(s) in a system**
- The design of a database start by describing its elements by an ERD
- Given the ERD, the database structure can be generated automatically

**ERDs contain three elements**
- **Entities**: tangible objects, roles, organizational units, devices, locations
  - The manifestation of a particular entity is known as an *instance*
  - e.g., an instance of entity *Student* would be Emily
- **Relationships**: descriptors for the relationships between entities
  - Can be modeled by an entity relationship matrix, e.g.: academic scheduling

<table>
<thead>
<tr>
<th></th>
<th>Student</th>
<th>Course</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Student</strong></td>
<td></td>
<td>Takes many</td>
<td>Majors in one</td>
</tr>
<tr>
<td><strong>Course</strong></td>
<td>Has many</td>
<td>Can require many</td>
<td>Is offered by one</td>
</tr>
<tr>
<td></td>
<td>Can be prereq for many</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Department</strong></td>
<td>Enrolls many</td>
<td>Offers many</td>
<td></td>
</tr>
</tbody>
</table>

- **Attributes**: features used to differentiate b/w instances of the entities
  - The most important are key attributes, which uniquely identify instances
  - e.g. the UIN of a *Student* is a key attribute
Unified modeling language (UML)

Created to capture best practices in OOD
- The intention of UML is to describe complex software systems
- However, some UML views can be used to provide a high-level description for many types of systems in different domains

UML diagrams offer two types of views of a system
- **Static (or structural)**: show the things in a system being modeled: objects, attributes, operations and relationships
- **Dynamic (or behavioral)**: describe how the objects interact with each other to create a functioning system
Most relevant diagrams
Static UML diagrams

Class diagram
- The most widely used UML diagram – the main building block of any OOD
- Shows classes, their attributes and operations, and relationship b/w classes
- In most cases a class has three parts
  - name at the top
  - attributes in the middle, and
  - operations or methods at the bottom

Composite Structure Diagram
- Depicts the internal structure of a class

Materials borrowed from http://creately.com/blog/diagrams/uml-diagram-types-examples/
Component Diagram

- Displays the structural relationship of components of a software system
- Mostly used with complex systems that have many components

Deployment Diagram

- Shows the hardware and its software
- Useful when software is deployed across multiple machines with each having a unique configuration
Object Diagram

– Similar to class diagrams; they also show the relationship b/w objects but with real world examples (They are used to show how a system will look like at a given time)

Package Diagram

– Shows how elements are organized into packages and dependencies between packages

Profile Diagram

– Introduced in UML 2, very rarely used
Dynamic UML diagrams

Activity Diagram

- Represent workflows in an graphical way, generally business workflow
- Sometimes used as an alternative to state machine diagrams

Use Case Diagram

- Most known behavioral UML diagrams
- Depict the actors involved in a system, different functions by those actors, and how these different functions interact
**State Machine Diagram**

- Similar to activity diagrams although notations and usage changes a bit
- Useful to describe the behavior of objects that act different according to the state they are at the moment

**Sequence Diagram**

- Shows how object interact with each other and the order those interactions occur
- Processes are represented vertically and interactions are show as arrows
Communication Diagram

- Similar to sequence diagrams but the focus is on messages passed b/w objects

Interaction Overview Diagram

- Similar to activity diagrams
- While activity diagrams shows a sequence of processes, IODs shows a sequence of interaction diagrams

Timing Diagram

- Very similar to sequence diagrams
- They represent the behavior of objects in a given time frame