Object-Oriented Design (B3)

• goals:
  - decomposition into classes
  - definition of functionality
• Booch methodology:
  - identify classes
  - identify functionality
  - identify relationships
• graphical notation e.g. UML
Responsibility-Driven Design (B3)

• Wirfs-Brock and Wilkerson c.1990
• driven by specification and delegation of responsibilities
• responsibilities delegated to classes
  – class responsible for its operations
• characterize desired behaviour
  – of system
  – of components
Analysis (U16)

- identify desired behaviour
  - what, not how
- use cases
  - major functionalities of the system
  - a set of related scenarios
- actors
  - interact with system
- example: credit-card validation system
  - actors: customer, store, bank
  - use cases:
    - perform card transaction (customer, store)
    - manage account (customer, bank)
Design (U4,U5,B3)

- identify components
  - classes, patterns, etc.
- associate behaviour with classes
  - responsibilities
  - cohesion
  - coupling
- identify relationships
  - association (“uses”)
  - aggregation (“has a”)
  - inheritance (“is a”)
Assignment of Responsibilities (B3)

• walk through scenarios
  - activities become responsibilities
• CRC cards
  - component, responsibility, collaborator
  - physical representation
• example: The Interactive Kitchen Helper
The Interactive Kitchen Helper (B3.4)

- Greeter component - offers user 5 choices
  1. Browse the database of existing recipes, without reference to any particular meal plan
  2. Add a new recipe to the database
  3. Edit or annotate an existing recipe
  4. Review an existing plan for several meals
  5. Create a new plan of meals

<table>
<thead>
<tr>
<th>Greeter</th>
<th>Collaborators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display informative initial message</td>
<td>Database Manager</td>
</tr>
<tr>
<td>Offer user choice of options</td>
<td>Plan Manager</td>
</tr>
<tr>
<td>Pass control to either</td>
<td></td>
</tr>
<tr>
<td>Recipe Database Manager</td>
<td></td>
</tr>
<tr>
<td>Plan Manager for processing</td>
<td></td>
</tr>
</tbody>
</table>
Recognizing Relationships (U5)

• association
  - “uses”
  - CRC collaborators
  - 1-way or 2-way

• aggregation
  - “has a”
  - one class contains or manages objects of another class

• inheritance
  - “is a”
  - every object of one class conforms to another with additional properties
Coupling and Cohesion (B3)

- cohesion
  - how closely related are the responsibilities of a unit?
- coupling
  - how dependent are units on each other?
  - number of associations
- system should be highly cohesive, loosely coupled
Information Hiding
(B3)

• implementation hidden behind simple interface
• interface: what
• implementation: how
• Parnas’s principles:
  - all required information and no more
  - user: for use of a component
  - developer: to carry out responsibilities of a component
Unified Modelling Language (U7, U23)

- Booch, Rumbaugh, Jacobson c. 1995
- standard language for graphically modelling a system
- structural modelling:
  - class diagram
  - internal structure diagram
  - collaboration diagram
  - component diagram
  - use case diagram
- dynamic modelling:
  - state machine diagram
  - activity diagram
  - sequence diagram
  - communication diagram
Use Case

• scenario – a description of specific use of the system by a user, e.g.
  Joe goes to the box-office to buy a ticket to the Ron Sexsmith concert. The clerk checks the availability. Joe chooses a seat. The clerk asks for payment. Joe hands his credit card. The clerk enters the credit card information. The system validates & bills the credit card. The system prints the ticket. The clerk gives Joe the ticket.

• use case – generalization of a set of scenarios that start with the same request to achieve the same user goal
  - goal – the business value to the user
  - system – application and hardware used by user
  - actor – an external entity that interacts with the system

• in this case, the system is the box office software & hardware, the goal is buying a ticket(s) and the actor is the clerk.
Use Case Template *(Lee & Tepfenhart, Figure 3-6)*

<table>
<thead>
<tr>
<th><strong>Use Case Name</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong>: A one or two sentence description of the use case</td>
</tr>
<tr>
<td><strong>Actors</strong>: Identifies the actors participating in the use case</td>
</tr>
<tr>
<td><strong>Includes</strong>: Identifies the use cases included in it</td>
</tr>
<tr>
<td><strong>Extends</strong>: Identifies the use cases that it may extend</td>
</tr>
<tr>
<td><strong>Pre-Conditions</strong>: Identifies the conditions that must be met to invoke the use case</td>
</tr>
<tr>
<td><strong>Details</strong>: Identifies the details of the use case – the action</td>
</tr>
<tr>
<td><strong>Post-conditions</strong>: Identifies the conditions that are assured to hold at the conclusion of the use case</td>
</tr>
<tr>
<td><strong>Exceptions</strong>: Identifies any exceptions that might arise in the execution of this use case</td>
</tr>
<tr>
<td><strong>Constraints</strong>: Identifies any constraints that may apply</td>
</tr>
<tr>
<td><strong>Variants</strong>: Identifies any variations that might hold for the use case</td>
</tr>
<tr>
<td><strong>Comments</strong>: Provides any additional information that might be important in this use case</td>
</tr>
</tbody>
</table>
Buy Tickets

Description: A customer purchases tickets for an event from a clerk

Actors: Clerk

Includes: Make Charges

Extends: none

Pre-Conditions: Event must be scheduled

Details:
1. Clerk requests availability for number of tickets for event
2. Box-Office displays availability
3. Clerk selects seats
4. Box-Office indicates cost
5. Clerk selects credit card purchase and enters card number
6. Box-Office makes charges to credit card
7. Box-Office prints tickets

Post-Conditions: Selected tickets are sold

Exceptions: Credit card not validated

Constraints: None

Variants: Cash purchase

Comments: None
Use Case Diagrams (U16, U17)

- shows how use cases are related to each other and to the actors
- shows which actors participate in which use cases
- actors: "roles" played by users
- use cases: a transaction among actors and the system
- associations
  - inclusion – one use case is a part of another
  - extension – specialized version of the case
- system boundary
- example: Box-Office
Use Case Diagram (U_2.3-6)

- **Box-Office**
  - **Buy Tickets**
  - **Buy Subscription**
  - **Make Charges**
  - **Survey Sales**

- **Actors**:
  - **Clerk**
  - **Kiosk**
  - **Credit card Service**
  - **Supervisor**

- **Relationships**:
  - Association
  - <<include>> relationship
  - Use case
  - System boundary
  - <<include>>
Class Diagrams (U4,U5,U8)

- depict static structure
- classes:
  - name
  - attributes
  - operations
  - responsibilities
- relationships:
  - association
  - aggregation and composition
  - inheritance
- advanced features: multiplicity, named relationships, dependencies, visibility
- examples
Relationships (U5-1)

- **Dependency**
  - Window to Event

- **Generalization**
  - ConsoleWindow, DialogBox, Control derived from Window

- **Association**
  - ConsoleWindow and DialogBox associated

- **Methods**
  - open(), close(), move(), display(), handleEvent()
Structural Relationships (U5-10)

- **School**: 1..*  
  - **Department**: 1  1..*  
    - **Instructor**: 0..1  
      - **Chairperson**: 0..1  
        - **Student**: 1..*  
          - **Course**: 1..*  
            - **Attends**: *  
              - **Teaches**: 1..*  
            - **Has**: 1  1..*  
              - **Assigned To**: 1..*  
              - **Member**: *
Modeling Simple Collaborations (U8-2)

PathAgent
- seek path
- avoid obstacles

Motor
- Move(d:Direction; s:Speed)
- stop()
- resetCounter()
- Status status()
- Integer distance()

CollisionSensor
- 1

Driver
- *

SteeringMotor
- 1

MainMotor
- 1

Motor
- 1

Responsibilities
- seek path
- avoid obstacles
Dynamic Modelling (U7, U18)

• modelling the behaviour of a system
• use case diagrams
• collaboration diagrams
  – flow of function calls between objects
  – emphasis: structural relationship
• sequence diagrams
  – interaction of objects as messages are passed
  – emphasis: timing
• state diagrams
  – sequence of states for an object
  – states, events and transitions
• activity diagrams
  – flowcharts
  – emphasis: flow of control from activity to activity
Sequence Diagrams

- objects
- messages
- object lifeline
- focus of control
- time-ordering of messages
- timing constraints
- examples: ODBC, phone call
Sequence Diagram (U18-2)

- **Objects**
  - `c : Client`
  - `{transient}`
  - `p : ODBCProxy`

- **Actions**
  - `<<create>>`
  - `setActions(a, d, o)`
  - `setValues(d, 3.4)`
  - `setValues(a, “CO”)`
  - `committed`
  - `<<create>>`
  - `<<destroy>>`

- **Time Line**

- **Lifeline**
  - `focus of control`
  - `committed`

© M. Winter
Sequence Diagram (U18-4)

Sequence Diagram:

- **s**: Caller
- **r**: Caller
- **Switch**: 
  - `setDialTone()`
  - `* dialDigit(d)`
  - `dialing`
  - `connect(r)`
  - `{dialing.executionTime < 30 sec }
  - `routeCall(s, n)`

- **c**: Conversation
  - `<<create>>`
  - `connect(r, s)`
  - `ring()`
  - `liftReceiver`

- **r**: Caller
  - `connect(s)`