Requirements Analysis Overview

- Requirements analysis results in an analysis model of 3 parts
  - functional model: use cases & scenarios
  - analysis object model: class & object diagrams
  - dynamic model: statechart & sequence diagrams (later in the unit)
- Investigate the problem domain as far as possible before moving to the solution domain (for design & implementation)

Actors, Objects and Classes

- What is the difference between an actor and a class and an object?
- Actor:
  - An entity outside the system to be modeled, interacting with the system ("Pilot")
- Class:
  - An abstraction modeling an entity in the problem domain, inside the system to be modeled ("Cockpit")
- Object:
  - A specific instance of a class ("Joe, the inspector").
Object: entity that has state and a defined set of operations which operate on that state.
- state – a set of object attributes
- operations – on that state; services to other objects

Class: abstraction of a set of objects with the same attribute, operations, relationships and semantics

Object Class Definition: a template for creating objects – declares all attributes, and operations

Class Example
- Name: employee
- Attributes: name, address, dateofbirth, employeenum, department, manager, salary, status, taxcode
- Operations: join, leave, retire, changedetails

Outline of Lecture 3
What is an object?
Identifying objects
  Grammatical Analysis
  CRC method
  B&D heuristics for object identification

Example Scenario: Publications Manager
“READ: A reader starts up the system, chooses new search parameters, and submits a search request. The system returns a list of papers, after which the reader either revises her search and the cycle is repeated, or the reader prints or saves the list and exits the system.”
Grammatical Analysis Method (1)

"READ: A reader starts up the system, chooses new search parameters, and submits a search request. The system returns a list of papers, after which the reader either revises her search and the cycle is repeated, or the reader prints or saves the list and exits the system."

Grammatical Analysis Method (2)

"READ: A reader starts up the system, chooses new search parameters, and submits a search request. The system returns a list of papers, after which the reader either revises her search and the cycle is repeated, or the reader prints or saves the list and exits the system."

Grammatical Analysis Method (3)

- NOUNS may represent
  - Objects and Attributes
- VERBS may represent
  - operations or services
- The grammatical method throws up noise – it needs to be used alongside other methods
- See also Abbot 1983 (B&D table 5-1)

B&D Table 5-1

<table>
<thead>
<tr>
<th>Part of speech</th>
<th>Model component</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proper noun</td>
<td>Instance</td>
<td>Alice</td>
</tr>
<tr>
<td>Common noun</td>
<td>Class</td>
<td>Field officer</td>
</tr>
<tr>
<td>Doing verb</td>
<td>Operation</td>
<td>Creates, submits</td>
</tr>
<tr>
<td>Being verb</td>
<td>Inheritance</td>
<td>Is a kind of</td>
</tr>
<tr>
<td>Having verb</td>
<td>Aggregation</td>
<td>Has, includes</td>
</tr>
<tr>
<td>Model verb</td>
<td>Constraints</td>
<td>Must be</td>
</tr>
<tr>
<td>Adjective</td>
<td>Attribute</td>
<td>Incident description</td>
</tr>
</tbody>
</table>
CRC: for teaching O-O thinking

- Idea: it is difficult when learning OO programming not to “think globally”
- Solution: immerse the learner in the “object-ness” of the material
- Concrete manifestation of object design: Class-Responsibility-Collaboration cards
- Written on 4x6 index cards
- Developed by Beck & Cunningham 1989

Class-Responsibility-Collaborator Model

- the class name of an object
  - creates a vocabulary for discussing a design
- responsibilities of an object
  - identify problems to be solved
  - a handful of short verb phrases, each containing an active verb
- collaborators of an object are
  - other objects which will send or be sent messages in the context of satisfying responsibilities

CRC Example

HotDraw: a drawing editor

Class-Responsibility-Collaborator Model

B&D Heuristics for Identifying Objects

- For each scenario or use case, try to identify objects
- Entity Objects: persistent data
- Boundary Objects: interaction between actors and system
- Control Objects: tasks performed by users and supported by the system
Heuristics to Identify ENTITY objects

- recurring nouns in the use cases e.g. Incident
- real-world entities the system needs to track e.g. FieldOfficer, Dispatcher, Resources
- real-world activities the system needs to track e.g. EmergencyOperationPlan
- terms that developers or users need to clarify in order to understand the use case e.g. EmergencyReport
- data sources or sinks e.g. Database, Printer
- always use the user’s terms for entity objects

Heuristics to Identify BOUNDARY objects

- Forms and windows needed to enter data into the system e.g. EmergencyReportForm
- Notices and messages that the system uses to respond to the user e.g. AcknowledgementNotice
- Do not model visual aspects of the interface with boundary objects (use screen mock-ups instead)
- Always use the user’s terms, and not implementation technology terms

Heuristics to Identify CONTROL objects

- COs hold state information for a period of time
- Try one control object per use case (or subflow)
- OR Identify one control object per actor in the use case e.g. ReportEmergencyControl (FieldOfficer) ManageEmergencyControl (Dispatcher)
- Life span of a control object is the extent of the use case or of a user session – the use case must have clear entry and exit conditions

Describing Objects

- Any of these three techniques may be used to identify objects, classes, methods and relationships.
- In the following lecture, we will examine ways to describe and document these objects in a consistent manner.