Object-Oriented Design

Lecturer: Raman Ramsin

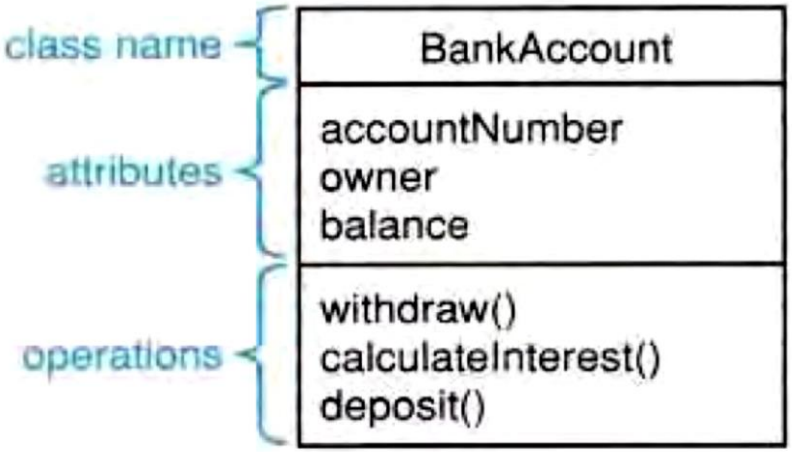
Lecture 7: Finding Analysis Classes

Analysis Workflow: *Analyze a Use Case*

- The *analysis workflow* consists of the following activities:
  - □ Architectural analysis
  - □ **Analyze a use case**
    - **Outputs:**
      - □ **analysis classes**
      - □ **use case realizations**
  - □ Analyze a class
  - □ Analyze a package

Analysis Classes: Typical Structure

- Analysis classes represent a crisp, well-defined abstraction in the problem domain.
- Analysis classes include:
  - □ a set of high-level candidate attributes
  - □ a set of high-level operations



Good Analysis Classes

- What makes a good analysis class?
  - □ Its name reflects its intent.
  - □ It is a crisp abstraction that models one specific element of the problem domain.
  - □ It maps to a clearly identifiable feature of the problem domain.
  - □ It has a small, well-defined set of responsibilities:
    - a responsibility is a contract or obligation that a class has to its clients;
    - a responsibility is a semantically cohesive set of operations;
    - there should only be about three to five responsibilities per class.
  - □ It has high cohesion - all features of the class should help to realize its intent.
  - □ It has low coupling - a class should only collaborate with a small number of other classes to realize its intent.

Bad Analysis Classes

- What makes a bad analysis class?
  - A functoid - a class with only one operation.
  - A stand-alone class - each class should be associated with a small number of other classes with which it collaborates to deliver the desired benefit.
  - An omnipotent class - a class that does everything (classes with "system" or "controller" in their name *may* need closer scrutiny).
  - A class with a deep inheritance tree - in the real world inheritance trees tend to be shallow.
  - A class with low cohesion.
  - A class with high coupling.
  - Many very small classes in a model – merging should be considered.
  - Few but large classes in a model – decomposition should be considered.

Class Identification Techniques

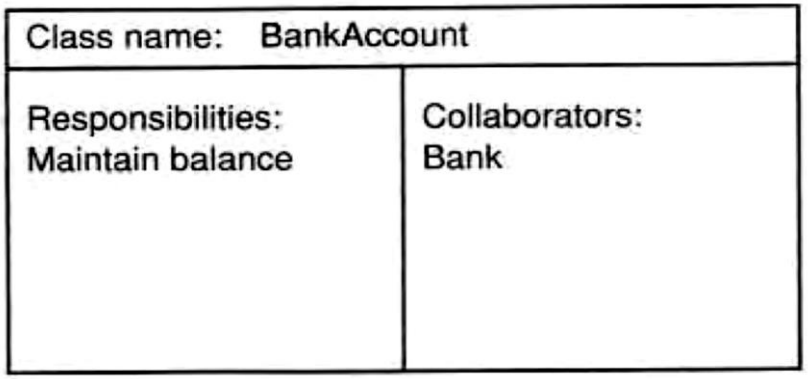
- Noun/Verb Analysis (*Grammatical Parsing*)
- CRC Analysis
- Use-Case-Based Analysis
- Real-World Analysis

Noun/verb analysis (*Grammatical Parsing*)

1. Collect as much relevant information about the problem domain as possible; suitable sources of information are:
   - □ The requirements model
   - □ The use case model
   - □ The project glossary
   - □ Any other document (architecture, vision documents, etc.)
2. Analyze the documentation:
   - □ Look for nouns or noun phrases - these are candidate classes or attributes.
   - □ Look for verbs or verb phrases - these are candidate responsibilities or operations.
3. Make a tentative allocation of the attributes and responsibilities to the classes.

CRC Analysis – CRC Cards

- CRC – Class, Responsibilities, and Collaborators
- Important things in the problem domain are written on CRC Cards. Each Card has three compartments:
  - □ class - contains the name of the class
  - □ responsibilities - contains a list of the responsibilities of that class (the functions it performs and even the information it is responsible to keep and provide)
  - □ collaborators - contains a list of other classes with which this class collaborates in order to fulfill the responsibilities



Class name: BankAccount	
Responsibilities: Maintain balance	Collaborators: Bank

CRC Analysis Procedure – Phase 1

- The participants are OO analysts, stakeholders, and domain experts.
- Phase 1: *Brainstorm - gather the information:*
    1. Explain that this is a true brainstorm.
        1. All ideas are accepted as good ideas.
        2. Ideas are recorded but *not* debated.
    2. Ask the team members to name the "things" that operate in their business domain - for example, customer, product.
        1. Write each thing on a sticky note; it is a candidate class, or attribute of a class.
        2. Stick the note on a wall or whiteboard.
    3. Ask the team to state responsibilities that those things might have; record these in the responsibilities compartment of the note.
    4. Working with the team, identify classes that might work together; record collaborators in the collaborators compartment of the note.

CRC Analysis Procedure – Phase 2

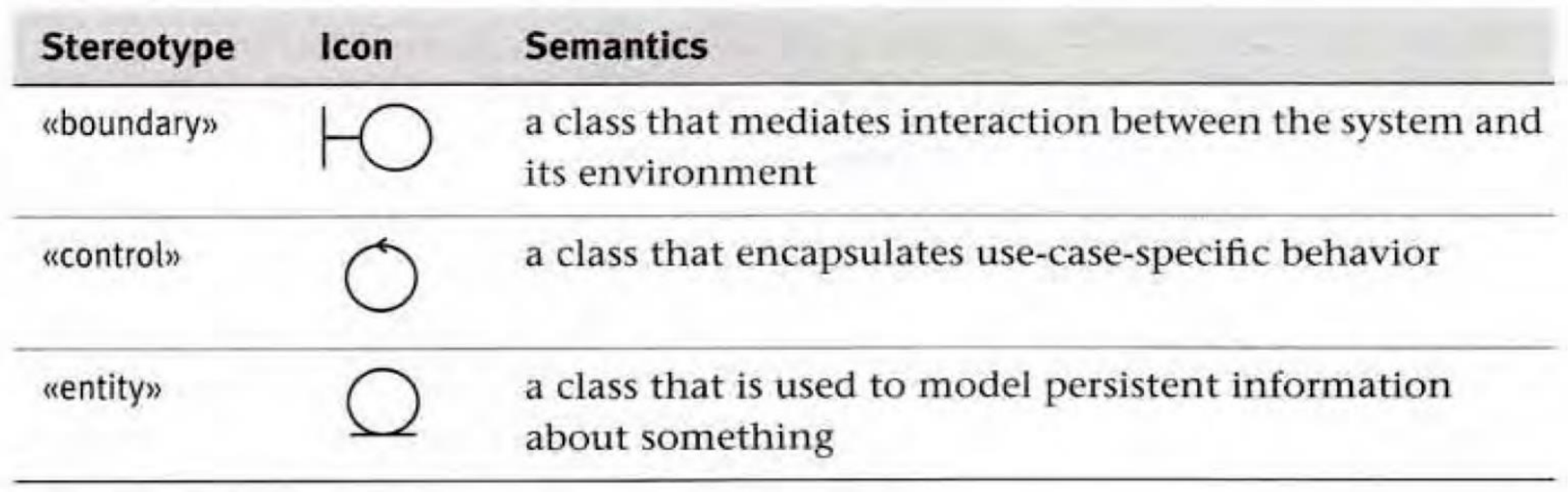
- The participants are OO analysts and domain experts.
- Phase 2: *Decide which sticky notes should become classes and which should become attributes:*
  - □ Analysis classes *must* represent a crisp abstraction in the problem domain. Certain sticky notes will represent key business concepts and clearly need to become classes.
  - □ If a note logically seems to be a *part* of another note, this is a good indication that it represents an attribute.
  - □ If a note doesn't seem to be particularly important or has very little interesting behavior, see if it can be made an attribute of another class.
  - □ If in doubt about a note, just make it a class.

Use-Case-Based Analysis

- Complements other techniques
- Starts from an initial list of classes
- List of classes is perfected and refined based on use cases:
  - Behavioral models are built showing use case realizations
  - Classes are identified based on the objects needed for use case realizations: the list of classes should provide instances which implement the behavior needed for the use cases;
    - New classes will be added if needed
    - Changes will be made to existing classes if required for use case realization

Use-Case-Based Analysis – Using RUP stereotypes

- RUP stereotypes can be used to focus analysis activity on three types of class



Stereotype	Icon	Semantics
«boundary»	├○	a class that mediates interaction between the system and its environment
«control»	↻	a class that encapsulates use-case-specific behavior
«entity»	○̲	a class that is used to model persistent information about something

Real-World Analysis

- Explore the real world for classes:
  - Candidates: physical objects, paperwork, interfaces to the outside world, and conceptual entities;
    - Physical objects: Things such as aircraft, people, and hotels may all indicate classes.
    - Paperwork: Things like invoices, orders, and bankbooks may all indicate possible classes; beware of paperwork supporting the redundant business processes that the new system might be trying to replace.
    - Known interfaces to the outside world: Things such as screens, keyboards, peripherals, and other systems can be a source of candidate classes, especially for embedded systems.
    - Conceptual entities: Things that are crucial to the operation of the business but are not manifest as concrete things; such as enrollment, educational program, and alarm condition.

Analysis Model

- Create a first-cut analysis model:
  - compare the results of different methods with the results of an examination of other sources of classes.
  - resolve synonyms and homonyms.
  - differences between the results of the different techniques indicate areas of uncertainty.
  - consolidate results into a first-cut analysis model.

Reference

- Arlow, J., Neustadt, I., *UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design*, 2nd Ed. Addison-Wesley, 2005.