CS 2704

Topic: Identifying Class Responsibilities

Outline

• Objects and responsibilities
• Identifying Attributes
• Identifying Responsibilities

Goal

• Have identified potential objects in problem domain
• Want to eliminate objects that have no role

Objects

• Definition from problem domain perspective
• An object is an abstraction of something in the problem domain, reflecting the capabilities of a system to keep information about it, interact with it, or both
• Object may be composed of other objects
• Object may be real or abstract (conceptual)

Responsibilities

• Responsibility of an object is
  – Set of public services (behaviors) object provides
  – Attributes necessary to services

Identifying Attributes

• Look for adjectives and possessive phrases in requirements document
• Find general description of object
• Determine what parts of description are applicable to problem domain
• If more worried about efficiency than flexibility, find minimal description for problem
Eliminating Attributes

- If attribute considered independently, make object instead of attribute
- If value depends on context of object, then treat as qualifier (?)
- Attributes of relationships should not be attribute of object
- Eliminate minor details that do not affect method
- Attributes that are unrelated to others may indicate object should be split into (at least) two objects

Specifying Attributes

- Attribute should be atomic (simple)
- Eliminate attributes that
  - can be calculated from others
  - address normalization, performance, object identification (implementation issues)
- Verify attributes make semantic sense together

Identifying Services

- Look for verbs in requirements – usually defines services of object of sentence
  “Person hits the ball”
- Look at user scenarios – different ways system can be used
- Look at each feature – require services of many objects

Use-Case Diagram

- Show how system fits into application domain – show user scenarios
- Actors - external entities that interact with system
- Use-case - description of system behavior from user point of view
- Scenario - particular instance of use-case

Example Use-Case

- Use-case diagrams used in analysis (before design)
- Elaborate on use-cases by written description
- Show response to use-case with event-trace diagram – helps identify responsibilities of objects
Event-Trace Diagram

Objects

method
value

Focus of Control

• Show time period that object has “control”

Example Diagram

Circulation
Catalog
Patron List
Circ List

Specifying Services

• Name service to match external request for service
• Identify argument list

Evaluating Class Designs

• Is class general enough for future needs?
• Does class have too much or too little responsibility?
• Are responsibilities unused or unrelated?
• Are responsibilities equally abstract?
• Are names appropriate?

Eliminating Objects

• Don’t need objects unneeded for application
• Test:
  – Object must provide service for system feature or other object in system
  – Object should have multiple attributes (some exceptions)
• Result will probably not include all objects used in implementation