Lecture 19: Responsibility-Driven Design, Part 2

Kenneth M. Anderson
Object-Oriented Analysis and Design
CSCI 6448 - Spring Semester, 2003

Credit where Credit is Due

- Some material presented in this lecture is taken from Object Design: Roles, Responsibilities, and Collaborations. © Addison Wesley/Pearson Education, 2003. ISBN 0-201-37943-0

Responsibility-Driven Design

- Last Lecture
  - Life cycle, Stereotypes, Responsibilities
- This Lecture
  - Designing Collaborations
  - Or, once you have responsibilities, how do your objects cooperate to fulfill them?
  - We saw an example of this last time, when looking at the “get total sale” problem

Collaborations

- Collaborations represent “neighborhoods” of objects that work together to fulfill a responsibility
  - Neighborhoods may overlap, that’s okay; what’s important is that for a specific responsibility, there is a neighborhood that handles it
- This is similar to Maciaszek’s take on design, in identifying collaborations to realize use cases; here we have started with use cases to get responsibilities, and now we design collaborations to handle each responsibility

Identifying Collaborations

- Use stereotypes
- Look at individual responsibilities
- Design the details of a complex responsibility
- Design collaborations for a specific use case or event

Using stereotypes

- The roles an object plays imply certain kinds of collaborations; Based on its role, what does an object need from its neighbors and what does it offer them?
  - We need to consider
    - how an object typically fulfills its responsibilities
    - how it is used by others

Information Holders

- Information holders know facts
  - It only collaborates with objects to gain access to the information it is responsible for knowing
- Questions to identify collaborations
  - Where does its information come from?
    - Does it create the information, ask for it, get told by someone else?
  - Is any information derived? From whom?
  - Does the information persist? Who handles persistence?
  - Is information cached? From where? When do I update it?
  - Does the information need to be converted to a different form? Who handles the conversion?

Structurers

- Structurers organize information
- Questions to identify collaborations
  - Where do the structured objects come from?
  - How are the structured objects processed?
    - Does the structurer handle iteration?
  - Does the structurer persist?
  - How are structured objects accessed?
  - Is the structurer responsible for answering cumulative questions (“how many attendees?”)

Service Providers

- Service providers perform computations
- Questions for identifying collaborations
  - Who has the information required by a service provider?
  - Are services configurable? How?
  - Is any part of a responsibility prone to change? Should this responsibility be isolated in a service provider?
  - Does the application require different forms of the same service? How does the service vary?

Controllers

- Objects that make decisions and direct the actions of others are controllers; They always collaborate with others for two reasons:
  - to gather information to make decisions
  - to call on others to act
- Their focus is on decision making; not on subsequent actions
- Questions for identifying collaborations
  - Who has the information needed to make decisions?
  - Who performs the actions once a decision has been made?
  - Is the decision making process complex? Perhaps it should be distributed over multiple controllers…
  - Are there events or intermediate results that the controller must track and respond to?

Coordinators

- Coordinators exist solely to pass along information and call on others to act; their focus is on holding connections between objects and forwarding information and requests to them
- Questions for identifying collaborations
  - How does a coordinator delegate work or pass along requests?
  - How does a coordinator find its delegates?
  - Do the delegates need to know about the coordinator?

Interfaces

- Interfacers provide bridges between naturally disjoint subsystems
  - They can act as a bridge between the system and its users (user interfacers), between different neighborhoods (internal interfaces) and different software systems (external interfaces)

Interfacers, continued

- Questions for User Interfacers
  - How does a user interfacer inform the system of user actions?
  - What system objects does the interfacer know of?
  - How many states does it track?
  - How do objects register interest in state changes?
- Questions for internal interfacers
  - How does the interfacer collaborate with objects outside of its neighborhood?
  - How does it find its neighborhood?
  - How does it delegate requests?
- Questions for external interfacers
  - Will the external interfacer have to convert data into object form?
  - How does the external interfacer connect to the outside world?
  - What will the interfacer do if it can't establish a connection?

Look at Individual Responsibilities

- Asking questions about how an individual responsibility is fulfilled can lead to collaborations
  - Just as we saw with the "get total sale" example from the last lecture
    - getTotal() in the Sale object, required getSubtotal() in the LineItem object, which required getPrice() in the Product object

Design the Details of a Complex Responsibility

- Another way to identify collaborations is to decompose a complex responsibility into smaller responsibilities
- Thus, "calculate annual corporate taxes" becomes
  - Calculate applicable municipality taxes
  - Itemize income, expenses, and allowable state tax deductions
  - Calculate applicable state taxes
  - Itemize income, expenses, and allowable federal tax deductions
  - Calculate applicable federal taxes
- We will need a collaboration to step through each of these responsibilities (e.g. manage the overall process) and collaborations to perform each individual responsibility

Design collaborations for a specific use case or event

- Start with a specific use case or event and design a collaboration to handle it
- Goal is to answer questions like
  - What services are invoked between collaborators? Who is in control?
  - How and when are objects created?
  - How long and how often do they need to see each other?
  - Where are the branches in logic? Where are the decision points?
  - Do the decision makers have what they need? Where will they get their information?
  - What information holders are passed around?

Testing Collaborations

- To test a collaboration, “simulate” it
  - You can quickly find errors and omissions in your model this way
    - a simulation can identify new objects and responsibilities
    - a simulation can show that a particular object is ill-conceived and not needed
    - a simulation can identify vague responsibilities
    - a simulation can provide justification for shifting, merging, or splitting responsibilities among candidates

Planning a Simulation

- Role-play the hard parts first
  - not everything is worth simulating
- Set a goal for the simulation
  - Test ideas; Study coordination and control; develop a consistent collaboration style, etc.
- Set boundaries based on your goal
- Assign candidates to team members
  - Each person is responsible for playing the role of particular objects!
- Simulate use cases
- Invent controllers if you need them
- Test one area at a time
- Test for what you don’t know
- Limit the time spent simulating

Running a simulation

- Start with an event
  - What object should be informed of the event? Is there a CRC card that describes that object? If not, make one
  - What responsibility does the event ask the object to fulfill; has this responsibility been identified? If not, write it down
  - Who will the object collaborate with to fulfill the responsibility?
- Make sure to express the event as an “intention”
  - Not “The user clicks a button”
  - But “The user saves the file”
- Now make your objects take responsibility for the event
  - Have a physical ball represent “control” and pass the ball around as messages are exchanged

During the Simulation

- Stay at the same conceptual level
  - If a collaboration requires a switch to a different conceptual level of the system, defer the details of that sub-collaboration to another simulation
- Follow the simulation closely
  - Do the patterns of message passing make sense?
- Think Critically
  - Ask questions like “okay, this object needed this piece of information to do that; how did it get that information?”
  - Or “How did I learn of your existence? If I don’t know about you, I can’t send a message to you!”
- Sketch the collaborations
  - Using CRC cards and lines between them; or a whiteboard
- Write down what you don’t know; deal with those issues later
- Rewrite candidate cards as new responsibilities are identified