Lecture 11: Collaborations

Kenneth M. Anderson
Object-Oriented Analysis and Design
CSCI 6448 - Spring Semester, 2005

Goals for this Lecture

- Review content of Chapter 5 from the textbook
- Discuss Collaborations
  - What are they?
  - How do we find them?
  - How do we simulate them?
Solving Design Problems

- Christopher Alexander (a man who created “design patterns” for architecture) said that we should solve design problems in “the least arbitrary manner possible.”

- Wirfs-Brock and McKean state that we can achieve this by
  - designing simple, consistent communication between objects
  - designing collaborations such that changes do not ripple through the entire system under design
  - partitioning responsibilities in a reasonable manner among collaborations and when collaborations follow predictable patterns

What is Object Collaboration

- Collaborations are requests from one object to another
  - Or a group of objects working together making requests on one another

- Our analysis and design model is incomplete until we flesh out which objects will be grouped together to handle the input events of our system under design
  - We organize objects into groups to fulfill collective responsibilities
  - We also decide how objects outside the group interact with services the group provides
  - Goal: to limit the impact of change; we should be able to modify the internals of a collaboration without impacting the rest of our application
Collaboration Characteristics

- OO Design is different from procedural design
  - Objects are arranged in networks, not hierarchies
  - Procedures separate data from behavior, whereas objects combine the two
  - It's easy to “slip back” into procedural programming however
  - Give one object too much power and it finds itself surrounded by simple information holders that don't do much
  - Each object in a collaboration should have a well-defined role to play and knows which of its neighbors to ask for help

Recording Collaborations

- You can lay the foundation for collaboration design with CRC Cards
  - On the lined side of the card, you have space to list an object's collaborators
  - You can indicate the relationships between an object's responsibilities and its collaborators by drawing lines between them
Finding Collaborations

There are many ways we can identify collaborations in the system under design:

- 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 role an object plays implies 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 provide access to its information
- 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? How are structured objects accessed?
  - Does the structurer persist?
  - Is the structurer responsible for answering cumulative questions?
    - For example, a Meeting object might be able to respond to the question “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?
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?

Interfacers

- 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 interfacers) and different software systems (external interfacers)

- 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?
Interfacers, continued

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 lecture 10

- 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 get passed around?
Examples

- The book provides examples of collaborations (and how to solve problems with them) on pages 172-177
  - Collaborations might be dictated by application architecture (172)
  - Too many connections from outside to objects within a neighborhood → Use a Facade Pattern
  - Too many branches and choices
    - Use the Double Dispatch pattern
  - The double dispatch pattern is shown using a sequence diagram, we will cover this diagram and a number of other UML diagrams in lectures 13 and 14

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
  - which objects and responsibilities will be invoked
- Assign candidates to team members
  - Each person is responsible for playing the role of particular objects!

Planning a Simulation, cont.

- 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 switches to a different conceptual level of the system, defer the details that switch 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