Lecture 20: Intro. to Design (Part 3)

Kenneth M. Anderson
Object-Oriented Analysis and Design
CSCI 6448 - Spring Semester, 2002

Credit where Credit is Due

- Some material presented in this lecture is taken from section 6 of Maciaszek's "Requirements Analysis and System Design". © Addison Wesley, 2000

Goals for this Lecture

- Cover the tutorial presented in Section 6.3 of the textbook
  - Discuss
    - Package Design
    - Component Design
    - Deployment Design
    - Collaboration Design

Last Lecture

- Covered
  - Detailed Design
    - Collaborations
    - Collaboration Diagrams
    - Extend analysis models to contain technical details
      - For each collaboration, define
        - » Structural part (subset of class diagram)
        - » Behavioral part (collaboration diagram)
    - Collaborations *realize* use cases and operations

Tutorial in Design Modeling

- Purpose: to continue developing the analysis models created in Section 2.2
  - adding technical detail to transform these models into a software design
- Demonstrate concepts introduced in chapter 6
  - The tutorial will cover architectural issues and detailed design issues

Step 1: Package Design

- Packages can be used to cluster related models together
  - In analysis: cluster related use cases
  - In design: cluster related classes
- As such, Maciaszek distinguishes between
  - Use case packages and Class packages
- Packages are typically used in large system design
  - small systems often can be understood without packages, since use cases provide a sufficient breakdown of concepts in such situations

Package Design, continued

- Class packages evolve during design as boundary, control, and database classes are identified
  - Entity classes were already identified during analysis
- Use case packages while helpful in early design stages are eventually replaced by the class package model

Package Design Example

- We return to the example of the system that allows the ordering of computer configurations via a web site
  - To start package design, we develop a use case package model that groups the use cases identified in figure 2.24 on page 52
    - The resulting model is shown on page 223
- Note: this model accomodates future use cases; as we develop additional use cases, this package structure provides a way to categorize them

Package Design, continued

- Next, we develop an initial class package model
  - Maciaszek suggests "impersonating the system" and imagine what needs to be done to accept a customer's order for a configured computer
- Figure 6.25 on page 225 shows the resulting package structure
  - The "customers", "computers", and "orders" packages come from the classes identified on page 59; all other packages were derived from the process above
- Note: Maciaszek asserts that package dependencies are non-transistive; but I don't belive it!
  - Taking his example of changes to customer not percolating to the GUI classes, what if we add a new attribute to our customer class? We will need a change in the GUI to ask for this piece of information

Step 2: Component Design

- Components are physical parts of the system; they cannot be separated from the implementation platform
  - Our example is a web application that will require at least a web server and a database server
  - On page 226, Maciaszek discusses common components of a web application, including scripts, cookies, applets, ODBC or JDBC, etc.

Component Design, continued

- Similar to the "impersonate a system" technique, Maciaszek develops an initial component design by steping through the process of ordering a computer and looking for "cohesive functional units"
  - On page 228, the diagram contains components for listing and displaying products, configuring a computer, making a purchase and tracking an order

Step 3: Deployment Design

- Once components have been designed, we must determine how they will be deployed
  - This is highly context dependent
  - Maciaszek does only a cursory job of covering this topic in section 6.3.3
    - His deployment diagram is underwhelming to say the least! (page 230)
- Issues to be determined include, what types of nodes are present in the system and how the components map to them
  - In addition, issues such as security, load balancing, backups, etc.

Step 4: Collaboration Design

- As discussed in the previous lecture, collaborations help to realize use cases and operations
  - here we are performing detailed design
  - the previous three steps were addressing architectural design
- The first step is to elaborate use cases; after analysis use cases are unlikely to contain enough information to design collaborations

Collaboration Design, continued

- To prepare for this step, Maciaszek recommends the use of a case tool for storing use case documents
  - in particular, being able to use a case tool's ability to number and renumber requirements is extremely useful
- Our purpose is to add information to the use case that include system-level demands while still maintaining an actor perspective

Use Case Elaboration

- As an example, we return to the use case "Order Configured Computer" shown on page 53 and extend it to the use case displayed on pages 233-236
- Comments
  - Some aspects of ui design are present
  - Many details defered in analysis, such as the contents of an order form, make their appearance
  - Maciaszek's use case violates Cockburn's "Do not 'check whether'" guideline! :-)
  - The elaborated use case reads more like a traditional requirements/design document

Elaborate all Use Cases

- Note: we must elaborate all use cases in the same manner
  - The idea being that with the elaborated use cases we have a better chance at identifiying collaborations that will help realize the use case
- Once we have elaborated the use cases and identified collaborations, we can move to designing the structural and behavioural part of each collaboration

Structural Design of Collaborations

- Figure 6.30 shows the structural part of a collaboration that will realize the "Order Configured Computer"
  - Note, it borrows two classes from the "Build Computer Configuration" use case which is not shown in the text book
- Comments on class diagram
  - Each class is categorized into boundary, control,entity, and database packages using a prefix notation, e.g. e_Configuration
  - Instantiation relationships are shown that show which objects will cause other classes to be instantiated, e.g. which objects create other objects
  - Special stereotypes are used to indicate special types of objects that are specific to the web application domain

Behavioral Design of Collaborations

- We now combine the information contain in the elaborated use case with the structural part of the collaboration to create the behavioral model of the collaboration
  - On page 239 is a collaboration diagram that does this for the "Order Configured Computer" use case
    - again I think it is unfortunate that Maciaszek does not number the messages in his collaboration diagram
- In practice, these models are only going to be created with iteration; indeed, I would recommend starting with sequence diagrams and elaborating them to include design details
  - once the sequence diagram is finalized, you can then create a collaboration diagram to make sure that all associations between classes have been specified

In summary

- Design consists of
  - architectural design
    - package design
    - component design
    - deployment design
  - detailed design
    - use case elaboration
    - collaboration design
      - structural and behavioral

In summary, continued

- Design, like analysis, will be highly iterative
  - At first, we will create partial architectural designs
    - indeed we may skip this part at first to focus on use case elaboration
  - Then we will start on use case elaboration
    - before elaborating all use cases, we may then start on collaboration design
- We then continue to iterate, elaborating use cases, designing their collaborations, mapping classes to packages, packages to components, and components to node
- Once we are done, we are ready to move on to implementation; there we can organize our implementation efforts around use cases, implementing their associated collaborations until the entire system is implemented