95-706

Lecture 5

Patterns for Assigning Responsibilities

Richard J. Orgass
H. John Heinz III School of Public Policy and Management
Carnegie Mellon University

Agenda

- Objectives
- Interaction Diagrams
- Responsibilities and Methods
- Responsibilities and Interaction Diagrams
- Patterns
  - Expert Pattern
  - Creator Patters

Objectives, References

- Objectives
  - Define patterns
  - Learn to apply five patterns (two today)

- References
  - Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995. ISBN 0-201-63361-2.
  - Craig Larman. *Applying UML and Patterns. An Introduction to Object-Oriented Analysis and Design*. Prentice-Hall, 1998. ISBN 0-13-748880-7

Introductory Comments

- Object-oriented system consists of
  - objects sending messages to other objects to complete operations
- Contracts -- First guess at responsibilities and post conditions for system operations
  - startup
  - enterItem
  - endSale
  - makePayment
- Interaction Diagrams show a solution
  - in terms of interacting objects
  - that satisfy these responsibilities
- Patterns applied during the creation of interaction diagrams showing responsibility assignments and collaborations.

Interaction Diagrams

- Interaction diagrams are one of the most important artifacts created in object-oriented analysis and design.
- Skillful assignment of responsibilities that occurs while creating interaction diagrams is very important.
- Time and effort spend on
  - generation
  - careful consideration of responsibility assignment
- absorbs a significant part of the design effort of a project.
- Codified
  - patterns
  - principles
  - idioms
- can be applied to improve the quality of designs.

Responsibilities

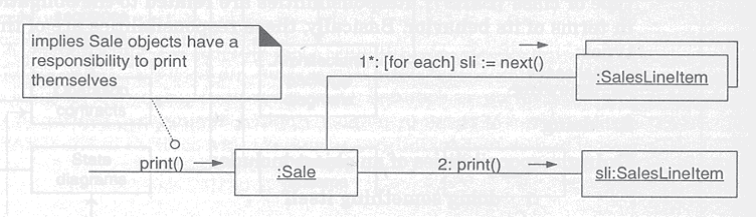
- contract or obligation of a type or class
- related to behavioral obligations of an object
- two types
  - knowing
  - doing
- Doing Responsibilities
  - doing something itself
  - initiating action in other objects
  - controlling and coordinating activities in other objects
- Knowing Responsibilities
  - knowing about private encapsulated data
  - knowing about related objects
  - knowing about things it can derive or calculate

Assigning Responsibilities to Objects

- Assignment occurs during OOD
  - Example
    - "A Sale is responsible for printing itself." (doing)
    - "A Sale is responsible for knowing its date." (knowing)
  - Knowing Responsibilities often inferred from conceptual model
    - illustrates attributes and associations
- Translation of Responsibilities into classes and methods
  - influenced by the granularity of responsibility
    - "provide access to relatlional databases" may involve
      - hundreds of classes
      - hundreds of methods
    - "print a sale" may involve
      - only a single or few methods
- Methods and Responsibilities are Different
  - responsibilities are implemented using methods

Responsibilities and Interaction Diagrams

- Diagram indicates
  - Sale objects have responsibility to print themselves
    - invoked with a print message
    - handled by a print method
  - Fulfillment of responsibility requires
    - collaboration with SalesLineItem objects asking them to print



Patterns

- Experienced developers (object-oriented and other)
  - build up a repertoire of
    - general principles
    - idiomatic solutions
  - to guide them in the creation of software
- Patterns are Codified
  - principles
  - idioms
  - given a name

Example Pattern

Pattern Name:	**Expert**
Solution:	Assign a responsibility to the class that has the information needed to fulfill it.
Problem it Solves:	What is the most basic principle by which to assign responsibilities to objects?

Patterns

- **A pattern is a named problem/solution pair that can be applied in new contexts, with advice on how to apply it in novel situations.**
- "One person's pattern is another person's primitive building block" is an object technology adage illustrating the vagueness of what can be called a pattern.
- Will skip what is appropriate to label a pattern
- Focus on pragmatic value of using the pattern style as a vehicle for presenting and remembering useful software engineering principles.

Patterns -- 2

- Patterns do not contain new ideas
  - serve to codify existing
    - knowledge
    - idioms
    - principles
  - the more honed and widely used, the better
- Patterns have suggestive names
  - Advantages
    - supports chunking
    - incorporating the concept into our understanding and memory
  - Facilitates communication
- Naming a complex idea such as a pattern is an example of abstraction
  - reducing complex form to a simple idea by eliminating detail
- Use concise names for patterns

Naming Patterns Improves Communication

- Example discussion
- Fred:
  - "Where do you think we should place the responsibility for printing a Sale? I think Model-View Separation would work well -- how about a SaleReportView?"
- Wilma:
  - "I think Expert is a better choice since it is a simple print and the Sales has all the data required in the printout -- let the Sale do it."
- Fred:
  - "Ok, I Agree."

- Chunking design idioms and principles with commonly understood names facilitates communication.

Patterns

- Discussion Summary
  - Skillful assignment of responsibilities is extremely important in OOD
  - Assignment of Responsibilities often occurs during creation of interaction diagrams
  - Patterns are named problem/solution pairs that codify good advice and principles often related to the assignment of responsibilities.
- **Patterns describe fundamental principles of assigning responsibilities to objects.**

Applying Patterns

- First Five Patterns:
  - Expert
  - Creator
  - High Cohesion
  - Low Coupling
  - Controller

- These address
  - basic, common questions
  - fundamental design issues

Setting the Stage

- Class model may define
  - dozens or hundreds of software classes
- Application may require
  - hundreds or thousands of responsibilities to be fulfilled
- During OOD, when interactions between objects are defined we make choices about the assignment of responsibilities to classes.
- Done well, systems tend to be
  - easier to understand
  - maintain
  - extend
- More opportunities to reuse components in future applications.

Expert Pattern

- Problem:
  - What is the most basic principle by which responsibilities are assigned in object-oriented design?
- Solution:
  - Assign a responsibility to the information expert -- the class that has the information necessary to fulfill the responsibility.
- Example:
  - In the point-of-sale application, some class needs to know the grand total of a sale.
- By Expert, look for that class that has the information needed to determine the total.

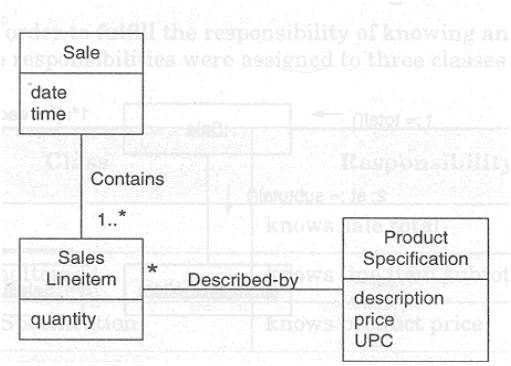
Expert Example

- In POST application, some class needs to know the grand total of a sale.
- **Start assigning responsibilities by clearly stating the responsibility.**

- By this advice, the statement is:
  - Who should be responsible for knowing the grand total of the sale?
- By Expert, we should look for that class of objects that has the information needed to determine the total.

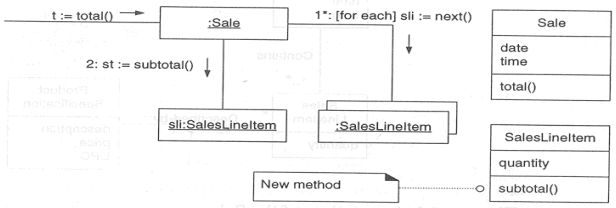
Expert Example -- 2

- Associations of the Sale class.



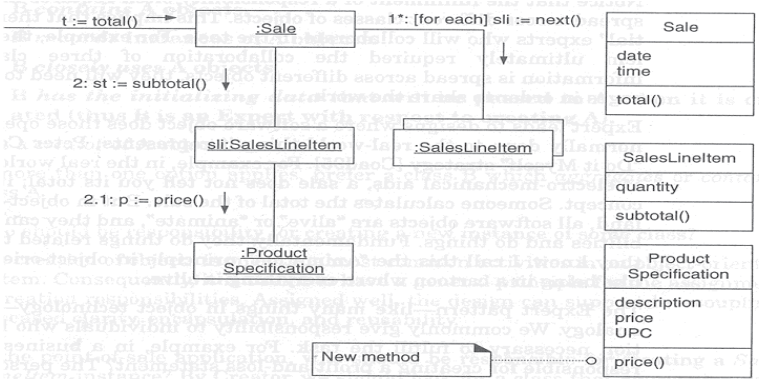
Expert Example -- 3

- What information is needed to determine the grand total?
  - Must know about all the SalesLineItem instances of a sale
  - The sum of their subtotals
- Only a Sale instance knows this, therefore, by Expert, Sale is the correct object for this responsibility.
- It is the **information expert.**
- Collaboration diagram showing result of discussions so far



Expert Example -- 4

- Collaboration Diagram to compute the sum.



Expert Example -- 5

- SalesLineItem
  - to fulfill it's responsibility of knowing and answering it's subtotal
    - must know the product price
    - ProductSpecification is Expert on answering its price
  - Therefore, a message must be sent to ProductSpecification asking its price.

Expert Example -- 6
Assigned Reponsibilities

Class	Responsibility
Sale	knows sale total
SalesLineItem	knows line item total
ProductSpecification	knows product price

Benefits of Expert Pattern

- Encapsulation is maintained
  - objects use their own information to fulfill tasks
  - supports low coupling which leads to more robust and maintainable systems
  - Low Coupling is a pattern discussed next class.
- Behavior is distributed across classes that have the required information
  - encourages more cohesive "lightweight" class definitions that are easier to understand and maintain
  - High Cohesion is supported (more in next class).

Creator Pattern Solution

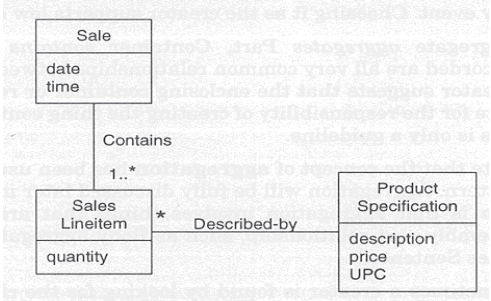
- Assign class B the responsibility for creating an instance of class A if one of the following is true:
  - B aggregates A objects.
    - Example: Sale and SaleLineItem
  - B contains A objects.
  - B records instances of A objects.
  - B closely uses A objects.
  - B has the initializing data that will be passed to A when it is created. That is, B is an Expert WRT creating A.
- If more than one case applies, prefer a class B which aggregates or contains members of class A.

Creator Pattern Problem

- Who should be responsible for creating new instances of a class?
  - One of most common activities in an OO system.
  - Useful to have general principle for assigning creation responsibilities.
- Well assigned responsibility for object creation means
  - low coupling is achieved
  - increased clairty of design
  - (much) better encapsulation and reusability

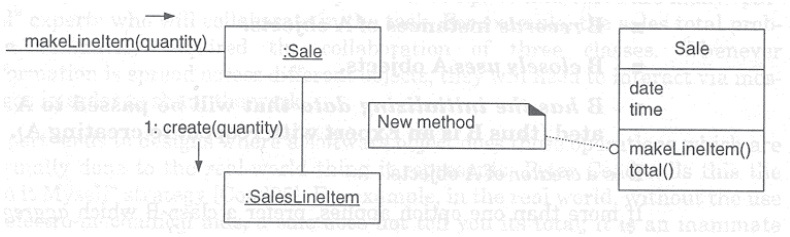
Creator Pattern Example

- Consider the point-of-sale application.
  - Who should be responsible for creating a SalesLineItem instance?
  - By creator, look for class that aggregates, contains... SalesLineItem instances.
  - Consider partial conceptual model:



Creator Pattern Example -- 2

- A Sale object contains and aggregates many SalesLineItem objects.
- Creator suggests Sale is a good candidate to be responsible for allocating SalesLineItem objects.
- Here is a collaboration diagram for the design



Creator Pattern Discussion

- Creator guides responsibility assignment by creation of objects
  - How is destruction of objects dealt with?
- Basic intent
  - find a creator which needs to be connected to created object in any event.
  - Selecting such a creator supports low coupling.
- Common Relationships between classes shown in class diagrams
  - Aggregate aggregates Part
  - Container contains Content
  - Recorder records Recorded
  - By creator pattern, all are good candidates for creators of objects