CSCI 253

Object Oriented Design:
Observer Pattern
George Blankenship

Overview

Creational Patterns
- Singleton
- Abstract factory
- Factory Method
- Prototype
- Builder

Structural Patterns
- Composite
- Façade
- Proxy
- Flyweight
- Adapter
- Bridge
- Decorator

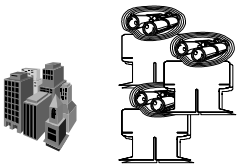
Behavioral Patterns
- Chain of Respons.
- Command
- Interpreter
- Iterator
- Mediator
- Memento
- Observer
- State
- Strategy
- Template Method
- Visitor

The Elements of a Design Pattern

- A pattern name
- The problem that the pattern solves
  - Including conditions for the pattern to be applicable
- The solution to the problem brought by the pattern
  - The elements (classes-objects) involved, their roles, responsibilities, relationships and collaborations
  - Not a particular concrete design or implementation
- The consequences of applying the pattern
  - Time and space trade off
  - Language and implementation issues
  - Effects on flexibility, extensibility, portability

The Observer Pattern: The Problem

Assume a one to many relationship between objects, when one changes the dependents must be updated



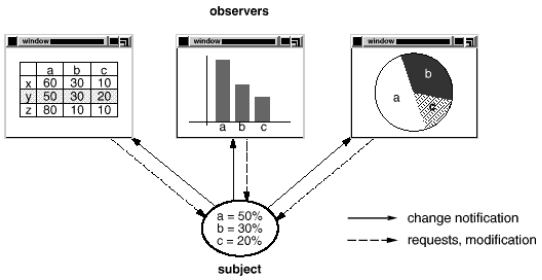
Subject Observers

- different types of GUI elements depicting the same application data
- different windows showing different views on the same application model

Also known as : Dependants, Publish-Subscribe

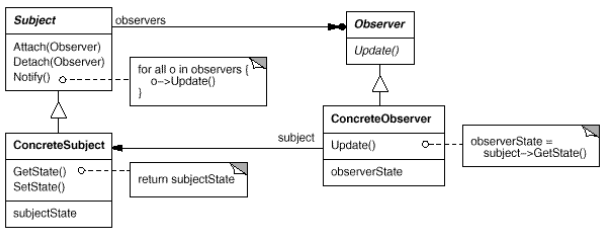
Observer Pattern George Blankenship 4

Observers



Observer Pattern George Blankenship 5

The Observer Pattern: Structure



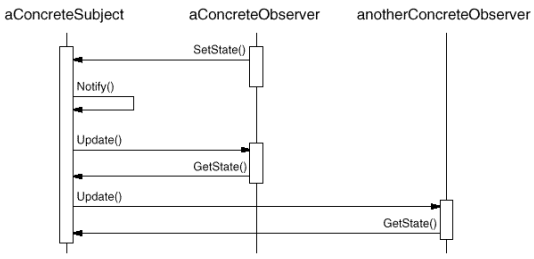
Observer Pattern George Blankenship 6

The Observer Pattern: Participants

- Subject: knows its observers, provides an interface for attaching (subscribe) and detaching (unsubscribe) observers and provides a *notify* method that calls *update* on all its observers
- Observer: provides an *update* interface
- ConcreteSubject: maintains a state relevant for the application at hand, provides methods for getting and setting that state, calls notify when its state is changed
- ConcreteObserver: maintains a reference to a concrete subject, stores a state that is kept consistent with the subject's state and implements the observer's *update* interface

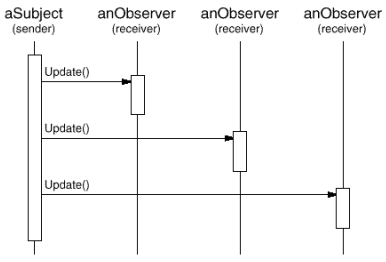
Observer Pattern George Blankenship 7

Participant Map



Observer Pattern George Blankenship 8

Participant Relations



Observer Pattern George Blankenship 9

The Observer Pattern: Collaboration

- ConcreteSubject notifies its Observers whenever a change occurs that could make its observers' state inconsistent
- After being informed of a change in the ConcreteSubject, a ConcreteObserver may query the Subject for information concerning its state and then reconcile its own state with that of the Subject
- The change in the ConcreteSubject can be initiated by one of the Observers or by some other application object

Observer Pattern George Blankenship 10

The Observer Pattern: Consequences

- Abstract and minimal coupling between Subject and Observer: the subject does not know the concrete class of any observer, concrete subject and concrete observer classes can be reused independently, subject and observer can even belong to different abstraction layers in the system
- Support for broadcast communication: the notification a subject sends does not need to specify a receiver, it will broadcast to all interested (subscribed) parties
- Unexpected updates: observers don't have knowledge about each others presence, a small operation might cause a cascade of spurious updates

Observer Pattern George Blankenship 11

The Observer Pattern: Implementation

- Map subjects to their observer: the Subject keeps explicit references to the Observers it should notify or some associative lookup is installed; memory/time trade off must be made
- Observing more than one subject can make sense in some situations
- Who triggers the updates (i.e. who calls notify):
  - have all state changing operations on Subject call notify after the subject's state is changed; consecutive operations cause several consecutive updates which may not be necessary and is inefficient
  - make clients responsible for calling notify at the right time; clients get the added responsibility to call notify which makes errors likely

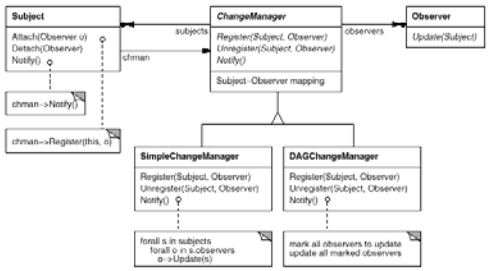
Observer Pattern George Blankenship 12

The Observer Pattern: Implementation

- When deleting a Subject the Observers should be notified so that they can reset their Subject reference
- Make sure that Subject is self consistent before calling notify
- Update efficiency can be improved when the observers register for specific events of interest
- Use appropriate protocol such as the pull and the push models
- Encapsulate complex update semantics

Observer Pattern George Blankenship 13

Implementation



Observer Pattern George Blankenship 14