Pattern-Oriented Software Architecture

by
Roman Punskyy.

Who am I ?

- I am a 3rd year undegraduate
- on MSci Computer Science degree

- You can contact me by email
- R.Punskyy@cs.ucl.ac.uk

Outline

- What is a Pattern?
- Characteristics
- Descriptions
- Categories
- MVC pattern

What is a Pattern ?

- "A ***pattern for software architecture*** describes a particular recurring design problem that arises in specific design contexts and presents a well-proven generic scheme for its solution. The solution scheme is specified by describing its constituent components, their responsibilities and relationships, and the ways in which they collaborate." [Buschmann].

Characteristics of Patterns (1)

- A pattern describes a solution to a recurring problem that arises in specific design situations.
- Patterns are not invented; they are distilled from practical experience.
- Patterns describe a group of components (e.g., classes or objects), how the components interact, and the responsibilities of each component. That is, they are higher level abstractions than classes or objects.

Characteristics of Patterns (2)

- Patterns provide a vocabulary for communication among designers. The choice of a name for a pattern is very important.
- Patterns help document the architectural vision of a design. If the vision is clearly understood, it will less likely be violated when the system is modified.
- Patterns are building blocks for the construction of more complex designs.

Characteristics of Patterns (3)

- Patterns provide a conceptual skeleton for a solution to a design problem and, hence, encourage the construction of software with well-defined properties.
- Patterns help designers manage the complexity of the software. When a recurring pattern is identified, the corresponding general solution can be implemented productively to provide a reliable software system.

Description of Patterns

- Context
  - The Context section describes the situation in which the design problem arises.
- Problem
  - The Problem section describes the problem that arises repeatedly in the context.
- Solution
  - The Solution section describes a proven solution to the problem.

Categories of Patterns (1)

- Architectural Patterns
  - An ***architectural pattern*** expresses a fundamental structural organization schema for software systems. It provides a set of predefined subsystems, specifies their responsibilities, and includes rules and guidelines for organizing the relationships between them." [Buschmann]

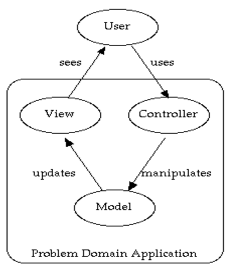
Categories of Patterns (2)

- Design Patterns
  - "A ***design pattern*** provides a scheme for refining the subsystems or components of a software system, or the relationships between them. It describes a commonly-recurring structure of communicating components that solves a general design problem within a particular context." [Buschmann]

Categories of Patterns (3)

- Idioms
  - "An ***idiom*** is a low-level pattern specific to a programming language. An idiom describes how to implement particular aspects of components or the relationships between them using the features of the given language." [Buschmann]

The Model-View-Controller Pattern



MVC Pattern

- **Model :** The core of the application. This maintains the state and data that the application represents. When significant changes occur in the **model**, it updates all of its **views**
- **Controller :** The user interface presented to the user to manipulate the application.
- **View :** The user interface which displays information about the **model** to the user. Any object that needs information about the **model** needs to be a registered **view** with the **model**.

How it all works in Java?

- A Model consists of one or more classes that extend the class java.util.Observable. This superclass will provide the register/notify infrastructure needed to support a set of views.
- The views are built of AWT or SWING components. However, views must implement the java.util.Observer interface.
- the controllers are the listeners in the Java event structure.

Steps …

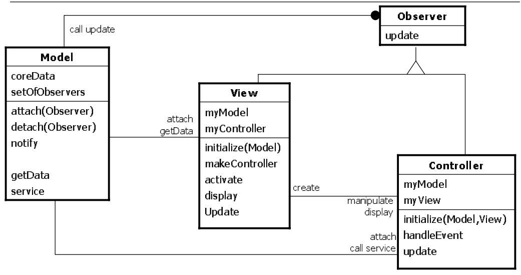
- write a Model that extends java.util.Observable
  - class accessors to get informtion about its current state
  - mutators to update the state
- create one or more views
  - Each view must implement the java.util.Observer interface and hence implement the update method

Steps …

- The Object in the second parameter will be used to receive additional information if passed.
- Eg.

```
interface Observer {
    void update (Observable t, Object o);
}
```

UML Diagram



MVC advantages (1)

- **Clarity of design**
  - by glancing at the model's public method list, it should be easy to understand how to control the model's behaviour.
  - When designing the application, this trait makes the entire program easier to implement and maintain.
- **Efficient modularity**
  - allows any of the components to be swapped in and out as the user or programmer desires - even the model!
  - Changes to one aspect of the program aren't coupled to other aspects, eliminating many nasty debugging situations
  - Development of the various components can progress in parallel.

MVC advantages (2)

- **Multiple views**
  - the application can display the state of the model in a variety of ways, and create/design them in a scalable, modular way.
  - Views are using the same data, they just use the information differently.
- **Ease of growth**
  - controllers and views can grow as the model grows.

MVC advantages (3)

- **Distributable**
  - with a couple of proxies one can easily distribute any MVC application by only altering the startup method of the application.
- **Powerful user interfaces**
  - using the model's API, the user interface can combine the method calls when presenting commands to the user.

Summary

- Patterns helps in developing software with known properties
- Help find concise solutions to design problems
- Can be implemented in any programming language
- Already widely used in application and business domain.

References

- Pattern-Oriented Software Architecture by Buschmann, Meunier, Rohnert, Sommerlad, Stal [John Willey & Sons]
- J. Bergin, Pace University
  - http://csis.pace.edu/
- H. C. Cunningham, Mississippi University
  - http://www.cs.olemiss.edu/