Modern Method Software Engineering

Home work 4

Sike Huang	**850414-A394**	**sike.huang@gmail.com**	**0762320173**
Shanbo Li	**840810-A478**	**shanboli@Gmail.com**	**0704646157**

2007-10-07

Question 2: Consider closed and open architectures (Fig 6-10 and Fig 6-11).

1. Compare design goals which are achieved and which could be difficult to meet in each (use bullets, first analysis design goals for Open and then similarly discuss the same for closed architecture).

Answer:

To open architecture, the main design goal is **Runtime efficiency**. Any layer can invoke operations from any layers below. In general, the openness of the architecture allows developers to bypass the higher layers to address performance bottlenecks. But an open architecture may very complex, so it is not easy to change a layer without touch other layer. And it leads to make it difficult to maintain the architecture.

To closed architecture, the main design goals are **High maintainability** and **flexibility**. Any layer can only invoke operations from the immediate layer below. It is very easy to change a layer on closed architecture. But for one layer can only access the layer below, there may be some bottlenecks that cannot be avoid.

2. Under which scenario (set of design goals) you will consider developing an Open architecture and under which developing a Closed architecture.

Answer:

Open architecture

Desktop application

Design goals:

- Operating system independent (java)
- Desktop application
- Based on Java Swing
- High performance required
- No database needed

Closed architecture

A website application

Design goals:

- Interactive with users
- Database involved
- Database changeable
- Web layer art engineer do not know database acknowledgement

Question 4: Consider eBay auction. Customers browse and view products and make an offer before end-date of given auction or buy theatrical on 'But-it now price'. Users are also registered with system and create profiles etc. etc. ….. System registers buyers, displays them the items active in auction etc. etc. …. You can check eBay's process. Formalize the general usage of eBay in terms of actors you can identify. And give a very brief usage scenario for each identified actor. (These will form assumptions for your answer)

Your task is to design an access control policy or discuss already in-use access control policy on eBay.

Answer:

Scenario name	**UserRegister**
Participating actor instances	Shanbo: seller
Flow of events	1. Shanbo has some secondhand book and want to sale them. 2. He thought it should be a good way to sale the books via internet. He wants to sale books with eBay. 3. To be a seller, he has to register first. He registers and applies to sale books. 4. Now Shanbo can start to sale books on ebay.

Scenario name	**AddItem**
Participating actor instances	Shanbo: seller
Flow of events	1. Shanbo add some books to eBay. 2. He sets the end-date and buy it now price.

Scenario name	**BuyItem**
Participating actor instances	Sike: buyer
Flow of events	1. Sike needs a book named OOSE 2. He wants to buy it from eBay so that he can get a low price. 3. He searches the book and found a seller named Shanbo gives a very low price. 4. He chooses buy-it-now, before he could continue, eBay asks him to login first. 5. He logins with his account, and continues on his shopping. 6. He pay the money to PayPal. Input his address and wait for the book at home.

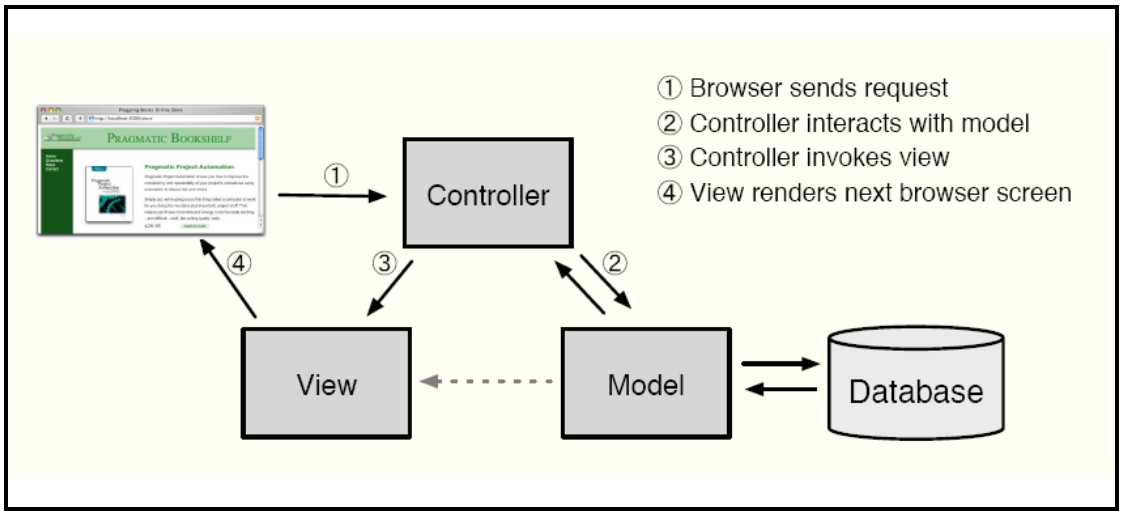
Scenario name	**ItemDeliver**
Participating actor instances	Shanbo: seller Sike: buyer
Flow of events	1. eBay notifies Shanbo that a buyer want to buy his book and already transfer

	money to PayPal. 2. Shanbo deliver the book to Sike. And wait for Sike's confirm so that he can get the money from PayPal.

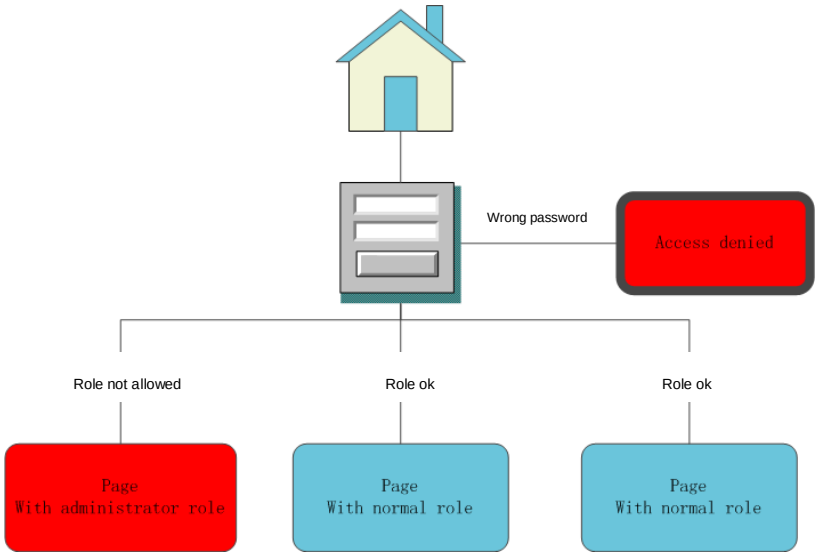
Scenario name	**ConfirmItem**
Participating actor instances	Sike: buyer Shanbo: seller
Flow of events	1. Sike received the book which post from Shanbo 2. Sike satisfies with the book and confirms that he has received the book to eBay. 3. PayPal transfers the money to Shanbo's bank account.

My access control policy:

My goal is that all access to any pages must to be authorized. Without authorization user can not access the certain page. As the MVC model shown below, we stored users with role-control in the database. Each user has a password. The password has to be stored with encryption, so that even the database manager cannot see the user's password.



When a user tries to login, the controller will encrypt the password which he just inputted. And compare the encrypt password with the password in database. If it is not correct, the user will be rejected. Otherwise, if the password is correct, his name and role will be stored in his cookie. And every page has a filer. Before displays the page's content, the filter will check if the user's role is allowed to access this page.



Question 6: Tetris field comprises of a collection of stones, some representing empty fields, and some representing parts of a particular Tetris block, whenever a Tetris block lands, it dissembles into its individual stones, which are considered independent from then on. It makes sense, therefore, to represent each stone as an individual object in an object-oriented Tetris application. Doing so naively results in a large amount of objects being allocated, deleted, and moved all the time. What design pattern can be applied for optimization, taking advantage of the fact that the stones are really very similar?

Answer:

To draw the stones we can use *Factory Method* pattern.

The factory method pattern is an object-oriented design pattern. Like other creational patterns, it deals with the problem of creating objects (products) without specifying the exact class of object that will be created. The factory method design pattern handles this problem by defining a separate method for creating the objects, which subclasses can then override to specify the derived type of product that will be created.

So**, Factory method pattern** can be used to draw different style of stones. We can introduce new identifiers for new kinds of Shapes, or we can associate existing identifiers with different shape objects. Template method pattern can be applied for optimization

To control the movement we can use *Template Method* pattern

A template method defines the program skeleton of an algorithm. The algorithm itself is made abstract, and the subclasses override the abstract methods to provide concrete behavior.

Using Template method for class MoveController, which is the base class of LeftKeyController, RightKeyController, UpKeyController and DropController. For handling any movement, it has to perform the following actions: erase(), action(), draw(). Although different event has different action(), the three steps have to be done in the same order, that's why Template Method is used here.

Question 8: List at least 3 general dangers and benefits of using design patterns.

Answer：

Benefits：

- Reuse the core of the solution
  - It is just why we need design patterns.
- Make the interfacing between many modules or classes more manageable.
  - Like Façade Pattern.
- Reduce the Complexity of Models
  - Like Composite pattern

Dangers:

- Functionality may lost in lower level of classes
  - Like Façade Pattern
- Design pattern may lead to more class generate, that means more objects will be generated at run time, which consume more memory resources to some degree and will slightly slow down the running speed.
- Some pattern novices understand Singleton first and believe that since it has been given a pattern name they should use it more often.
- Sometimes they indicate a language deficency. Many of the GoF patterns are rarely applied in

Question 10: Based on the problem description for "Advert Consultancy Inc" (Introduced in Question 5 of Homework 2), and you results from homework 2 and homework 3, and the examples starting on pages 288, 337 and 380 of the textbook, Develop a system design and object design document which includes:

1. Subsystem decomposition structure (using class diagrams).

2. Mapping subsystems to processors and components - hardware/software mapping (using UML deployment diagrams).

3. Persistent storage solution for the problem.

4. Access control, global control flow and boundary conditions.

5. Applying design patterns to designing object model for the problem (using class diagrams).

6. Writing contracts for noteworthy classes

Question 12: Consider a sorted binary tree structure for storing integers. Write invariants in OCL denoting that

- All nodes in the left subtree of any node contain integers that are less than or equal to the current node, or the subtree is empty.
- All nodes in the right subtree of any node contain integers that are greater than the current tree, or the subtree is empty.
- The tree is balanced.

Answer:

Predefined:

```
class TreeNode {
        public int item;              // The data in this node.
        public TreeNode left;         // Pointer to the left subtree.
        public TreeNode right;        // Pointer to the right subtree.
        public int length;            // the length of tree
}
```

context BinaryTree **inv**:
TreeNodes -> forAll (b:TreeNode | b.left.item <= b.item or b.left == null)
context BinaryTree **inv**:
TreeNodes -> forAll (b:TreeNode | b.right.item >= b.item or b.right ==null)
context BinaryTree **inv**:
TreeNodes -> forAll (b:TreeNode | abs(b.left.length – b.right.length) <= 1 || b.length ==0)