CSC301 Introduction to Software Engineering

Assignment 2

Details

Topics: object oriented design (OOD)
Weight: 10%
Given: June 6th, 2008
Due: June 22nd @ 11:55pm, 2008
Work submission: submit the design file to your team cvs repo. Name the file teamXdesign.pdf *where X is your team number*.
Deliverables:

- Due June 22nd: UML OOD
- Due June 25th (*in class*):
  1. Assignment cover sheet signed by all team members
  2. A closed envelope from each team member with
     a. team member evaluation form filled for other team members.
     b. team member evaluation summary form.

Assignment Description

Who are you?
A software architecture team.

Inputs:

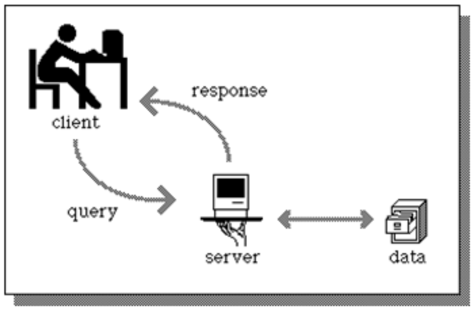
- ➢ A new set of screen shots detailing updates to the interface of the media application.
- ➢ OOA document (the one you made in assignment 1).

Task:
In this assignment, your task is to produce an object oriented design which includes:

1) Subsystems definition
2) Classes and interfaces in each subsystem
3) Attributes and operations for each indentified class
4) Preconditions and post-conditions for each operation

You must provide detailed class diagram for the design. You are free to include other types of UML diagrams. Remember that you will be passing this document to the development team and if they misunderstand your design, they will not produce a correct implementation.

Context:



The media application is a client server application. Your scope is limited to the client application. The server will be designed by a different company (*we will call it the server-company*). The client and server will communicate using text-based messages written in XML format. The server-company has already defined the transactions it supports (listed in appendix A) and the message format (sample shown in appendix B – *don't worry about the details of these messages – these are just listed as examples*). In your design, you do not need to define classes for all those transactions, instead – pick a category, e.g. content transactions and show how it will be incorporated in your design.

Object Oriented Design

Encapsulation:
An important property of good OOD is employing 'encapsulation': the idea that only the procedures that form part of an object are able to access its data. Contrast with a more traditional scheme, in which any part of the program can access any piece of data. Encapsulation accords with the idea of mirroring the real world: for example, to alter a plane's height, you must go through the appropriate procedures with the flight controls -- you can't just reach for the altimeter and turn it a few thousand feet up or down. The principal benefit of encapsulation is that the designer of an object has complete control over its internal workings, and can therefore provide guarantees about its behavior (to the designers of the other objects with which it works). It's easier to build a large system from such well-defined pieces. Secondarily, the internal design of an object can be altered without the changes necessarily propagating to the rest of the program. Used well, encapsulation can provide good controls on the costs of software updates.

Polymorphism:
Careful definition of the interfaces to objects is the key to 'pluggability' (sometimes called 'polymorphism' or 'substitutability'), which is the most powerful benefit of object oriented design. By choosing one or another implementation, we can assemble different systems: just in the same way that we can create different PCs or hi-fi systems by plugging in different compatible implementations of the various parts. Taking this idea to the limit, we can make kits of objects with many interoperable pieces, which can be rapidly assembled into a variety of end products..

Loosely coupled-highly cohesive:
loosely coupled-highly cohesive is an attribute of good OOD, and it refers to an approach to designing interfaces across classes/subsystems to reduce the interdependencies across/between them – in particular, reducing the risk that changes within one class will create unanticipated changes within other classes. This approach specifically seeks to increase flexibility in adding classes, replacing classes and changing operations within individual classes.

Design Patterns:
Design patterns have two major benefits. First, they provide you with a way to solve issues related to software development using a proven solution. The solution facilitates the development of highly cohesive modules with minimal coupling. They isolate the variability that may exist in the system requirements, making the overall system easier to understand and maintain. Second, design patterns make communication between designers more efficient. Software professionals can immediately picture the high-level design in their heads when they refer the name of the pattern used to solve a particular issue when discussing system design. In this assignment, we expect you to at least use the following patterns: command, object pool, singleton, facade, iterator, observer, and chain-of-responsibility. You might find other patterns useful as well.

UML Software

Here is a non-exhaustive list of some UML tools you could use to do this assignment.

- MagicDraw (http://www.magicdraw.com/)
- Dia (http://live.gnome.org/Dia)
- ArgoUML (http://argouml.tigris.org/)
- UMLet (http://www.umlet.com/)
- Violet (http://horstmann.com/violet/)
- TOPCASED (http://topcased-mm.gforge.enseeiht.fr/website/modeling/uml/index.html plugin for Eclipse. beware: install all pre-requisite plugins first)
- Visio (http://office.microsoft.com/en-us/visio/default.aspx)
- Poseidon UML (http://www.gentleware.com/).

Since we are allowing you to use any UML software. You are going to submit a pdf file (not the file saved by the tool). To do that, you will have to generate a pdf! You will need to install the following free PDF-generation software: http://www.cutepdf.com/products/cutepdf/Writer.asp This will install a pdf printer driver on your computer. Use the print menu to print the design using the installed printer driver.

Appendix A:

MEMBERSHIP TRANSACTIONS

1.Register
2.Login
3.Logout
4.Retrieve-password
5.Retrieve-profile
6.Edit-profile

CONTENT TRANSACTIONS

7. Add-content
8. Delete-content
9. Edit-content
10. Retrieve-my-content
11. Retrieve-content-licensed-by-me
12. Retrieve-content-licensed-from-me
13. Search-Video-Content-by-Tag
14. Search-Audio-Content-by-Tag
15. Search-Content-by-Tag
16. Search-Video-Content-by-Keyword
17. Search-Audio-Content-by-Keyword
18. Search-Content-by-Keyword
19. License-Content

C. PRODUCTION TRANSACTIONS

20. Retrieve-my-productions
21. Add-Channel
22. Delete-Channel
23. Edit-Channel
24. Add-Schedule
25. Retrieve-Schedule-For-Viewer
26. Retrieve-Schedule-For-Producer
27. Edit-Schedule
28. Retrieve-Channel-History
29. Retrieve-Channel-Stats
30. Add-Ad-Contract

D. SUBSCRIPTION TRANSACTIONS

31. Add-Subscription
32. Delete-Subscription
33. Retrieve-My-Subscriptions
34. Search-Video-Channel-by-Tag
35. Search-Audio-Channel-by-Tag
36. Search-Channel-by-Tag
37. Search-Video-Channel-by-Keyword

38. Search-Audio-Channel-by-Keyword
39. Search-Channel-by-Keyword
40. Start-Viewing
41. Stop-Viewing

E. ADS TRANSACTIONS

42. Add-Ad
43. Delete-Ad
44. Edit-Ad
45. Retrieve-My-Ads
46. Retrieve-Ads-proposed-by-me-and-not-accepted
47. Retrieve-ads-proposed-by-me-and-accepted
48. Add-Ad-Proposal
49. Edit-Ad-Proposal
50. Delete-ad-proposal
51. Retrieve-ads-proposed-to-me-and-not-accepted
52. Retrieve-ads-proposed-to-me-and-accepted

F. TAGS TRANSACTIONS

53. Generate-top50-content-tags
54. Generate-top50-channel-tags
55. Generate-top50-ad-tags
56. Generate-newest20-content-tags
57. Generate-newest20-channel-tags
58. Generate-newest20-ad-tags

G. BALANCE TRANSACTIONS

59. Retrieve-payment-received-from-payer
60. Retrieve-payment-received-from-all
61. Retrieve-payment-made-to-payee
62. Retrieve-payment-made-to-all
63. Retrieve-payment-made-to-website
64. Retrieve-membership-fees
65. Retrieve-storage-units-fees
66. Retrieve-broadcast-bandwidth-fees

Appendix B:

Sample client request to server for registering on website:

```
<register>
        <user-name>jSmith</user-name>
        <password>9999</password>
        <first-name>John</first-name>
        <last-name>Smith</last-name>
        <salutation>Mr.</salutation>
        <birth-date>1975-11-06</birth-date>
        <sex>m</sex>
        <email-id>jSmith@gmail.com</email-id>
        <address>
                 <street>100 College</street>
                <city>Toronto</city>
                <state>Ontario</state>
                <zipcode>M4X1K3</zipcode>
                <country>Canada</country>
       </address>
       <membership-type>channel producer</membership-type>
       <gui-language>English</gui-language>
       <paypal-email>jSmith@gmail.com</paypal-email>
       <homepage-content>some content</homepage-content>
       <storage-unit>256K</storage-unit>
       <credit-card-information>
              <name-on-card>John Smith</name-on-card>
              <number-on-card>456709871234567</number-on-card>
              <expiry-date>2009-11-05</expiry-date>
              <card-type>visa</card-type>
       </credit-card-information>
</register>
```

Sample server response to the above client request:

```
<register-response>
       <email-id>jSmith@gmail.com</email-id>
</register-response>
```

Sample client request to server to retrieve ads posted by a user:

```
<retrieve-my-Ads>
        <email-id>jSmith</email-id>
</retrieve-my-Ads>
```

Sample server response to above request:

```
<retrieve-my-Ads-response>
        <search-results>
           <record id = 1>
              <ad-information>
                <ad-title>Ad1</ad-title>
                <tags>ad1 Toronto scenes</tags>
                <description>Toronto scenes</description>
                <URL>
                    <fileURL>www.ad1jsmith.com</fileURL>
                    <previewURL>www.ad1preveiw.com</previewURL>
                    <imageURL>www.ad1image.com</imageURL>
                </URL>
                <fileSize>256K</fileSize>
                <encoding-bandwidth>128K</encoding-bandwidth>
                <mediaType>Vide</medaType>
                <permissionType>Public</permissionType>
          </ad-information>
        </record>

     </search-results>
</retrieve-my-Ads-response>
```

Sample client request to server to add content:

```
      <add-content>
                  <email-id>jSmith@gmail.com</email-id>
                  <title>Content1</title>
                  <tags>toronto scenes jsmith</tags>
                  <description>Toronto Scenes</description>
                  <URL>
                         <fileURL>www.content1Smith.com</fileURL>
                         <previewURL>www.content1preview.com</previewURL>
                         <imageURL>www.content1image.com</imageURL>
                  </URL>
                  <fileSize>256K</fileSize>
                  <encoding-bandwidth>128K</encoding-bandwidth>
                  <mediaType>Video</mediaType>
                  <PriceInfo>
                        <viewerCount>1000</viewerCount>
                        <pricePerViewer>20.00</pricePerViewer>

                  </priceInfo>
       </add-content>
```

Sample server response to above request:

```
      <add-content-response>
                   <title>Content1</title>
      </add-content-response>
```