New York University
Computer Science Department
Courant Institute of Mathematical Sciences

Course Title: Software Engineering
Instructor: Jean-Claude Franchitti
Course Number: g22.2440-001
Session: 6

(Individual) Assignment #3 – Design Modeling and Architectural Analysis

I. Due

Monday April 2 2018, at the beginning of class.

II. Objectives

1. Learn how to approach modeling using UML 2.0/3.0.
2. Learn how to perform an architectural evaluation and approach architectural analysis and design
3. Learn how to use architectural, design, and implementation patterns.
4. Learn how to perform architectural analysis.

Disclaimer: The artifacts created in this assignment will be used to drive forward subsequent technical planning in terms of deliverables, and most importantly the actual form of the object model you will use to represent the architecture. It is therefore important to create and record the artifacts as formally as possible.

III. References

1. Slides and handouts posted on the course Web site.

IV. Software Required

1. UML 2.0-Compliant Editor (e.g., Sparx Systems's Enterprise Architect, Omondo UML Eclipse plugin, etc.)

V. Assignment Part I - Design Modeling:

1. Problem Statements:

See Problem Scenarios #1 and #2 outlined in Assignment #2. In addition to the questions listed in this section, you should make sure that you follow the coverage and consistency heuristics discussed in class when it comes to using UML 2.0 notations to model your application architectures. Therefore, you should introduce

UML notations as needed in addition to the ones discussed in this section. You should also create separate analysis and design models as needed and use patterns and heuristics as appropriate. The next assignment will focus on a more elaborate use of patterns. Please document how your models are created using coverage and consistency and other architectural heuristics as this will be taken into account heavily in the grading process.

2. Brainstorming and CRC Cards:

   For each problem scenario in section 1 above, "brainstorm" out a list of object candidates that model objects you consider key abstractions necessary to solve the problem. You may want to use 4"x6" index/CRC cards to assist with this modeling exercise. If you decide to put together a CRC session, it is fine to work with your group partner(s) as CRC sessions are typically done in groups.

   Questions:

   a. For each candidate ask the following important "anthropomorphic" questions :
      - "What do I know"? – what data will the object candidate encapsulate?
      - "What do I do"? – what methods (functions) will the object support?
   b. Assign each of these candidates an index card. In the responsibilities section document the data/methods you have identified. Using the set of primary scenarios developed in the previous exercise walk through each step, identifying how your objects will implement the flow described within the scenario. Need to add a responsibility as you walk through a scenario? Go ahead! Need to add a new object? Remove a responsibility? Remove or combine objects? Again change at this stage is encouraged! Please document the steps you followed as part of your homework solution for this question.

3. Class Diagrams:

   For each problem scenario in section 1 above, use the set of object candidates identified in the previous exercise and develop a set of class diagrams that will document the classes and their relationships in a more formal manner.

   Questions:

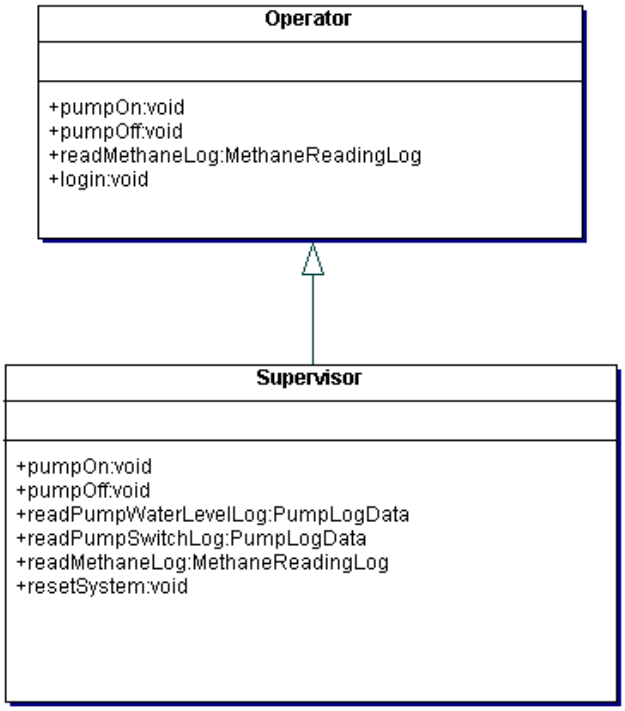
   a. In your diagrams consider the following design issues:
      i. For each class what types should the data attributes be?
      ii. What are the method signatures for a class i.e. parameters and return types?
      iii. How / should the diagrams be partitioned into packages?
   b. What relationships exist between each class, are there Composition? Aggregation? Generalization-Specialization?

4. Sequence Diagrams:

   For each problem scenario in section 1 above, use the flow you have identified at the use case and CRC level between objects and develop a set of sequence diagrams to represent the flow of control from a given actor(s) and its execution of the methods you have identified with your emerging model.

5. Inheritance versus Composition:

   Consider the following partitioning of the roles described in the first problem scenario outlined in section 1:



   Assume the following implementations:

   Operator

   - *pumpOn* – activates the pump (if within min and max water levels)
   - *pumpOff* – deactivates the pump (if within min and max water levels)
   - *readMethaneLog* – allows the operator to read the methane log for up to 24 hours, with time specified as a parameter
   - *login* – user supplies username and password to log in

   Supervisor

   - *pumpOn* – activates the pump (at any time)

- *pumpOff* – deactivates the pump (at any time)
- *readPumpWaterLevelLog* – allows the supervisor to read the log entries generated when the pump is switched on or off due to being triggered by the high of low water sensor, for up to 30 days, with time specified as a parameter
- *readPumpSwitchLog* – allows the supervisor to read the log entries generated when the pump is switched on or off by an operator or supervisor, for up to 30 days, with time specified as a parameter
- *readMethaneLog* – allows the supervisor to read the methane log for up to 30 days, with time specified as a parameter
- *resetSystem* – returns the pump to automatic functionality, switches off any activated alarms
- *addNoteToLog* – allows the supervisor to add a note to a log, supplied as a parameter

For the purposes of example let us add two new roles to this model.

Administrator

An administrator is responsible for setting all the configuration parameters in the system. These include:

a. Adding a user
b. Setting a user role
c. Switching the pump on and off as per the supervisor role
d. Resetting the system
e. Adding a note to the log that captures the details associated with a pump being switched on or off by a user (to add a maintenance entry)
f. Setting the maximum water level
g. Setting the minimum water level
h. Setting the maximum methane level
i. An administrator may *not* view any reports

Service Engineer

A service engineer is responsible for maintaining and testing the pumping system and sensors. This includes the following functions:

a. Switching the pump on and off as per the supervisor role
b. Resetting the system
c. Adding a note to the log that captures the details associated with a pump being switched on or off by a user (to add a maintenance entry)

Questions:

a. How can the inheritance hierarchy be re-factored to capture these new roles, does the new hierarchy make sense as an “is-a” typology?

b. Use composition to redesign the role set presented above, Operator, Supervisor, Administrator, Service Engineer.
c. Discuss the relative advantages and disadvantages of each approach.
d. Come up with a similar context and answer the same questions for the second scenario outlined in section 1.

6. Architectural Evaluation Discussion:

For each problem scenario in section 1 above, choose an "architecturally interesting" part of your object model. For the selected set of classes address the following questions:

a. What alternatives with respect to cohesion and coupling could you have made with this part of the model? What were the advantages and disadvantages of each possibility?
b. In your use of inheritance or composition, how would you restructure the model to use one over the other? What would be the advantages and disadvantages of each alternative?
c. Critique your model

Note: This part of your homework may be used as a basis for a short presentation to the instructor and a follow-on discussion with the class.

VI. Assignment Part II – Architectural Analysis:

1. Problem Statements:

See Problem Scenarios #1 and #2 outlined in Assignment #2.

2. Architectural Capabilities (mine pump scenario only):

Fault Detection

Your system relies heavily upon the correct operation of hardware, specifically pumps sensors and alarms. These need to be closely monitored to ensure that they are all responsive during normal operations

Questions:

a. Design a "heartbeat" monitor to examine each piece of hardware in your system – update your class diagrams with the new architecture, and provide a simple sequence diagram to indicate how this will operate.
b. Design a "ping" monitor to examine each piece of hardware in your system– update your class diagrams with the new architecture, and provide a simple sequence diagram to indicate how this will operate.

c. Design an exception class hierarchy that captures and propagates faults for all types of hardware in the system.
d. Should these designs be generic (i.e., for all types of system hardware? Or specialized to meet individual operating requirements?)

Fault Recovery

Sensors are typically inexpensive to replace but are prone to failure (Mean Time Between Failure 8000 hours), as a consequence multiple "instances" of a given server type can exist within a specific mine shaft.

Questions:

a. Based on one of the detection strategies you designed above, specify, using class and sequence diagrams build, recovery strategies using Shadowing, Voting and Replication based architectures.
b. Report your findings.

Performance and Concurrency

In order to partition and handle the incoming system events, refactor your architecture to include the presence of "active objects". These will be processes/threads that respond to incoming events (such as interrupt driven sensors) or those that periodically poll resources (such as sensors that are polled for readings).

Questions:

a. Introduce active objects into your class diagram in order to handle the dynamic aspects of your system, e.g. sensor input, alarm triggering, pump actions, user input and so on.
b. Assign each active object a relative priority based on the following levels:
    1. DAEMON – run me in the background if there are spare CPU cycles.
    2. LOW – run me if there are no higher priority events waiting.
    3. MEDIUM – run me if there are no HIGH priority events waiting.
    4. HIGH – run me first!
c. What happens if there are multiple HIGH priority processes waiting, how should these be handled?

d. What arbitrates who should have priority? Introduce this to your architecture. What Priority should it run at? How should it work?

Security

The application provides a role based security model, verified by a simple user/password access control scheme.

Questions:

a. Add the following to your architecture:
    1. Classes to handle the login procedure, this will validate the user based on a supplied username/password stored in a database and assign them the appropriate role. Describe the dynamic behavior using a sequence diagram.
    2. Classes to administer the security database, creating, updating, editing usernames, passwords and role.
    3. Classes to encrypt / decrypt usernames and passwords using an encryption algorithm chosen by the customer from the following list (Blowfish, DES, Triple-DES).

Validity/Testability

As a safety-critical/mission-critical application your company has decided to provide architectural hooks into the system to facilitate programmatic testing. As a consequence you will be required to undertake the following:

e. Provide a class and illustrate sequence diagram to provide recording/playback capabilities for the following interface(s):
    1. Commands sent to the pump
    2. Information sent/polled by each sensor
    3. Commands sent to the Evacuation Alarm
f. Provide specialized interfaces for your basic abstractions that allow access to all private attributes for unit test purposes. Do not allow these to be accessed by "normal" clients ensuring system operation (***hint*** : use two UML interfaces, one for normal operation, one for test purposes only.)

3. Abstract Factory Design Pattern (mine pump scenario only):

After an analysis of the market your company has decided to provide device drivers for a range of different platforms, to enable your application to communicate with each type of sensor supported by the application (high water, low water, methane sensor). These are listed as follows:

1. Linux USB Driver

2. Linux Serial Port Driver
3. Linux Wireless Driver
4. Win32 USB Driver
5. Win32 Serial Port Driver

Questions:

c. Define a class diagram that shows the Abstract Factories and Product Hierarchies necessary to implement these variations for every type of Sensor supported. Make sure each factory is also defined as a *Singleton* (why? What would be the advantage of this?)

4. Composite Design Pattern (mine pump scenario only):

Your company has decided to provide an "environmental simulator" used to simulate events that the system may receive in order to test installations and diagnose outstanding software or hardware problems. As part of this simulator you will be required to build an object structure used to model expressions in PSL, the proprietary *Pump Simulator Language*.

a. This language has the following primitives:
   i. High Water Event – represents a maximum water level reading from a high water sensor
   ii. Low Water Event – represents a minimum water level reading from a low water sensor
   iii. Methane Alarm – represents a dangerous methane reading from the methane sensor
   iv. Supervisor Switches Pump On – represents the supervisor switching the pump on manually
   v. Supervisor Switches Pump Off – represents the supervisor switching the pump off manually
   vi. Run for N minutes – runs the pump for N minutes, this must follow an event from (i)-(iv)
   vii. Reset – resets all alarms, switches off the pump
b. These primitives can be combined into higher level constructs called "workflows". A workflow is a composite that may hold one or more primitives, or other workflow objects.

Using the composite pattern, develop the following tactical solutions:

Questions:

a. Develop a class diagram showing the components of the language and how they interrelate.
b. Develop a number of sequence diagrams that show example PSL processing scenarios.

In order to handle user interactions in the presentation layer your architecture team has decided to handle individual user actions in separate *action* objects. An action will receive a user request, unpack input variables, call the appropriate business logic tier interface(s), get back the results and send them as output back to the user.

a. An Action object must support the following functionality:
    i. StartTransaction – will start a transaction every time the action object is called
    ii. endTransaction – will end (and commit) the transaction when output is returned to the user
    iii. onError – will rollback the transaction if an error occurs
    iv. doWorkflow – will do the necessary action
b. These actions can be nested, i.e. it is possible to have an action object made up of smaller action object. These "child" actions are simply executed in sequence.

Using the composite and template method patterns do the following:

Questions:

a. Develop a class diagram showing the structure you have designed
b. Develop two sequence diagrams
    i. Show a normal transaction
    ii. Show a transaction where error (and rollback occurs)
c. Hint : For the superclass in the composite you will need an abstract class not an interface…why?
d. For composite action objects how can I add logic to the execution of nested "leaf" or "composite" actions i.e. *if X then do action Y else action Z.*

5. Adapter Design Pattern (mine pump scenario only):

Many pumps typically run "hot" and require airflow cooling systems to keep them within safe operating temperatures. These systems are expensive to operate and so should run "in step" with the operation of the pump.

Questions:

a. Build a class diagram that provides a class/interface for the cooling system. This interface provides two methods:
    i. Switch (on/off), turns the cooling system on or off based on the parameter, when switched off the pump will run for an additional 20 minutes in order to cool the pump down following operation.
    ii. Emergency Off – switches off the cooling system immediately

b. Build an adapter that will operate the pump and cooling system simultaneously
c. Model the dynamic behavior of the adapter using a sequence diagram, show scenarios that demonstrate both normal on/off operation and the behavior exhibited when the methane alarm is activated.

6. Observer Design Pattern (mine pump scenario only):

A pump will generate a diagnostic message every 30 seconds that reports the following information:

- The current temperature of the pump
- The amount of water pumped (in cubic cm) since the last reading
- The current time

This information is required to be sent to a user interface, a mail message must be sent to a dedicated email inbox, and it must be placed in a dedicated pump reading database.

Questions:

a. Make the appropriate changes to your pump class, use an observer pattern to model the scenario described above using a class diagram. Produce a sequence diagram to illustrate how this pattern will work when the pump generates a diagnostic message.

7. Visitor Design Pattern (mine pump scenario only):

Each sensor and pump in the system will also be required to have the following two data attributes:

a. Last Serviced – the calendar data it was last serviced on
b. Next Service Due – the time by which it must be serviced (an offset of N months from the Last Serviced date)

In addition a pump will have the following information:

c. Pump Location (e.g. Shaft 6)
d. Logical Pump Name (e.g. “Old Ironsides”)

Using the visitor pattern you are required to build a simple reporting addition that will…

Questions:

a. Generate a report showing all the sensor and pump information on request from a supervisor.

8. Architectural Patterns:

For each problem scenario in section 1 above, using the session 8 presentation(s) on architectural patterns as a "pattern catalog" select one *or more* architectural patterns as the basis for your final system. For each pattern selected undertake the following:

Questions:

a. Using UML *packages*, partition your existing class diagram(s) to conform to the architectural configuration suggested by your chosen patterns.
b. What additional classes will your packages require in order to meet the requirements of this architecture? e.g.: How is distributed communication managed? How are common resources efficiently shared?

9. JEE Implementation Patterns:

For each problem scenario in section 1 above, using a component technology of choice (e.g. JEE or .NET), use the presentation as a "pattern catalog" to identify how the following tactical solutions can be used to facilitate the use of your existing object model as a basis for a JEE/.NET solution:

Questions:

c. Business Delegate
d. Service Locator
e. Session Façade
f. Intercepting Filter
g. Fast Lane Reader

VII. Deliverables

1. Electronic:
   Your assignment files (**Assignment3 *files***) must be emailed to the TA. The files must be sent by the beginning of class. After the class period, the homework is late. The email clock is the official clock.
2. Written:
   Printout of your assignment solution. The cover page supplied on the next page must be the first page of your assignment file
   Fill in the blank area for each field.

NOTE:
The sequence of the hardcopy submission is:

1. **Cover sheet**
2. **(Part I) Brainstorming, modeling, and architectural evaluation documentation for each scenario**
3. **(Part I) UML 2.0/3.0 analysis and design object model for each scenario**

4. **(Part II) Brainstorming, modeling, and architectural analysis documentation for each scenario as required**
5. **(Part II) UML analysis/design object models for each scenario**

Name _____ Username: _____ Date: _____
(last name, first name, username is SID)
Section: _____

Assignment 3 Assessment

Assignment Layout

o Assignment is neatly assembled on 8 1/2 by 11 paper.
o Cover page with your name (last name first followed by a comma then first name), email, and section number with a signed statement of independent effort is included.
o (Part I) Brainstorming, modeling, and architectural evaluation documentation is satisfactory.
o (Part I) UML 2.0/3.0 analysis and design object models are satisfactory.
o (Part II) Brainstorming, modeling, and architectural analysis documentation is satisfactory.
o (Part II) UML analysis/design object models are satisfactory.
o File names are correct.

(Part I) Brainstorming, Modeling, and Architectural Evaluation

o Word document.
o Completeness of analysis and design for each problem scenario.
o Appropriateness of architectural analysis and design comments for each problem scenario.

(Part I) UML 2.0/3.0 analysis object models

o XMI file.
o Completeness and conciseness of analysis and design object model for each scenario.
o Appropriateness of modeling choices for each problem scenario.

(Part II) Brainstorming, Modeling, and Architectural Analysis

o Word document.
o Completeness of analysis/design for each problem scenario.
o Proper application of patterns for each problem scenario.
o Appropriateness of architectural analysis comments for each problem scenario.

(Part II) UML analysis object models

o XMI file(s).
o Completeness and conciseness of analysis/design object model(s) for each scenario.

o Appropriateness of modeling choices for each problem scenario.

Total points _____
Professor Comments:

Affirmation of my Independent Effort: _____
(Sign here)