Spring 2012 Software Engineering Qualifying Exam

This is an open book exam. Basic calculators are allowed, but no computers or other devices that have communication capability are allowed; this means that cell phones cannot be used as calculators. There are a total of 100 points on this exam. Be sure to show your work in case your answer may deserve partial credit, but do not add spurious or frivolous content in hopes that something you say might be right. Content in an answer that is irrelevant to the question may cause point deductions.

Problem Description (all questions use this description and code)

You and your team are setting out to build a "smart home" system. A smart home has a computer system that uses devices throughout the house to sense and control the home. The two basic smart home device types are *sensors* and *controls*. These are installed throughout the house and each has a unique name and ID, location, and description. The house has a layout (floorplan) image, but is also managed as a collection of rooms. Device locations are rooms, and per-room views and functions must be supported.

Sensors are of two types: queriable and event announcer. For example, a thermostat is a queriable sensor: the computer application sends out a query and the thermostat replies with the currently measured temperature. An example of an event announcer is a motion sensor: it must immediately announce the event that motion was sensed, without waiting for a query. Controls actually control something, like the position of a window blind, the state of a ceiling fan, or whether a light is on or off. However, all controls are also queriable sensors; querying a control results in receiving the current settings of the control.

Device data (received from a sensor or sent to a control) depends on the type of device, and could as simple as one boolean flag (e.g., is door open or closed, turn light on or off), or could be a tuple of data fields (e.g., the current temperature and the thermostat setting, or fan on/off and speed). Each field (even just one) has a name.

The system will provide a "programming" environment using something like a scripting language for the user to customize their smart home environment. It should also allow graphical browsing of the current state of the house, and direct manipulation of controls (overriding any scripting control). The system must also provide some remote web-based access for use when the homeowner is traveling.

Below is some partial code from a potential system implementation; some of the methods are incomplete and have comments that act as placeholders for code that would be in that place.

```
public interface SensorObserver
{
   public void sensorChanged(Sensor changedSensor);
}

public interface Sensor
{
   public String getID();
   public boolean setID(String id);
   public boolean addObserver(SensorObserver so);
   public boolean deleteObserver(SensorObserver so);
   public SensorValue getCurrentValue(String valueName);
}

public class Thermometer implements Sensor
{
   private vector<SensorObserver> observers;
   private String id;
   private SensorValue lastReading;

   public Thermometer() { observers = new vector<SensorObserver>(); }

   public boolean setID(String id) { this.id = id; }
   public String getID() { return id; }

   public boolean addObserver(SensorObserver so) { observers.add(so); }
   public boolean deleteObserver(SensorObserver so) { observers.delete(so); }

   private void notifyChange()
   {
      for (SensorObserver so: observers) {
         so.sensorChanged(this);
      }
   }

   public SensorValue getCurrentValue(String valueName)
   {
      if (!valueName.equals("temperature"))
         return null;
      SensorValue val = /* code here to actually query the physical sensor */;
      lastReading = val;
      return val;
   }

   public void checkSensor()
   {
      SensorValue oldval = lastReading;
      SensorValue val = getCurrentValue("temperature");
      if (!val.equals(oldval))
         notifyChange();
   }
}
```

```
public class Rule implements SensorObserver
{
   private SmartVector mySensors;  // Java vector with method "find(int sensorID)"
   private Control control;        // a rule activates only one control
   private ParseTree parsedRule;

   public ErrorCode setRule(String RuleScript)
   {
      ErrorCode error = ERROR.AllOK;  // default is no error
      //
      // parse script string and verify syntax; identify
      // sensors, make sure they are valid, and attach to them
      // as an observer. For each sensor token in the script, we do:
      //
         sensor = System.findSensor(id);
         if (sensor != null) {
            mySensors.add(sensor);
            sensor.addObserver(this);
         } else {
            error = ERROR.UnknownSensorID;
            break;
         }
      // finally, after all is done, we return our status
      return error;
   }

   public void sensorChanged(Sensor changedSensor)
   {
      evaluateRule(changedSensor);
   }

   private void evaluateRule(Sensor changedSensor)
   {
      // walk through the rule as stored in parsedRule,
      // each time we come to a sensor value being referred to
      // (as id.valueName) we do:
         if (id == changedSensor.getID())
            tmpSensor = changedSensor;
         else
            tmpSensor = mySensors.find(id);
         value = tmpSensor.getCurrentValue(valueName);
         // value is then used as appropriate in the rule
      // after rule is fully evaluated we do:
      if (ruleResult == true) {
         // fire necessary actions on control
      }
   }
}
```

Questions

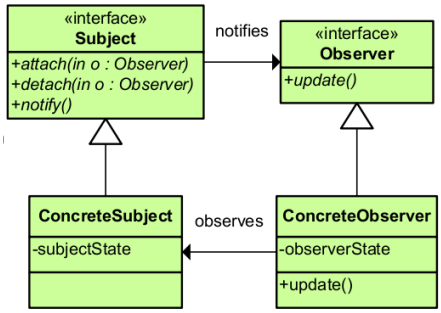
[20pts] 1. Create a domain model for this problem, modeling the domain information structure in a UML class diagram. Include classes and their relevant attributes, and relationships between classes, including name, cardinality, and direction where appropriate. Remember that some of what is described above might not be relevant to a *problem domain* model.

[10pts] 2. Pick one software development process style (e.g., waterfall, spiral, or others) that you would prefer your team to use, and **explain** why. What benefits would this process give you? What assumptions are you making about your team? What would this process style be good at, and what would it be not so good at? (Note that it is the explanation that is the important part of the answer, and is where the problem points will be earned.)

[10pts] 3. What are two potential risks that could jeopardize the success of your project? **Explain** why you chose them.

[10pts] 4. Pick one software architectural style that you think would be useful for at least part of this system and **explain** how it would help. You can choose any known architectural style from any reference, **except** client-server.

Problems 5 and 6 make use of the Observer design pattern:



(from Design Pattern Card, (C) 2007 Jason McDonald, www.mcdonaldland.info)

[10pts] 5. In the code above the Observer pattern is used, yet some of the suggested implementation details in the pattern figure are changed. **Describe** how the classes in the code above map to the classes in the figure, **explain** where the code differs from the suggested pattern, and **explain** why the differences might be a good idea.

[10pts] 6. In looking at the code above it would seem that many sensors might have very similar code in parts of their class when implementing the Sensor interface. Thus we might consider making the Sensor interface an abstract class. **Explain** the benefits and potential drawbacks. As part of your explanation, describe which code in the Thermometer class would be moved into the Sensor class.

[10pts] 7. Consider sensor values and the variables that hold sensor values (some for very short times, some for longer times). **Explain** in terms of dataflow coverage testing why from an individual variable view, the code shown here is fairly easy to test for dataflow coverage.

[10pts] 8. If we take a more abstract view by thinking of "sensor values", Then the code might actually be considered very hard to test. **Explain** why. (Hint: view the code from an object-oriented viewpoint rather than just a static "lines of source code" view, and consider what would be happening in a fully deployed system.)

[10pts] 9. For an overall view of this part of the system, pick one of the statements "this code is easy to test" or "this code is hard to test", and justify why your selection is the proper view of this code.