Milestone 3 Report

Engicoders

H. Bird, B. Karacelik, J. Peters, J. Ropotar, A. Rybka

Department of Computer Science, University of British Columbia

COSC310: Software Engineering

Dr. Gema Rodriguez-Perez

March 8, 2024

You will need to model and analyze your use cases and requirements properly using the appropriate models with the appropriate level of granularity. For the projects, at a minimum, you will need to ensure that you have the appropriate **sequence diagrams** (supporting your use cases), and you will also need to include a **class diagram** for the different parts of the system.

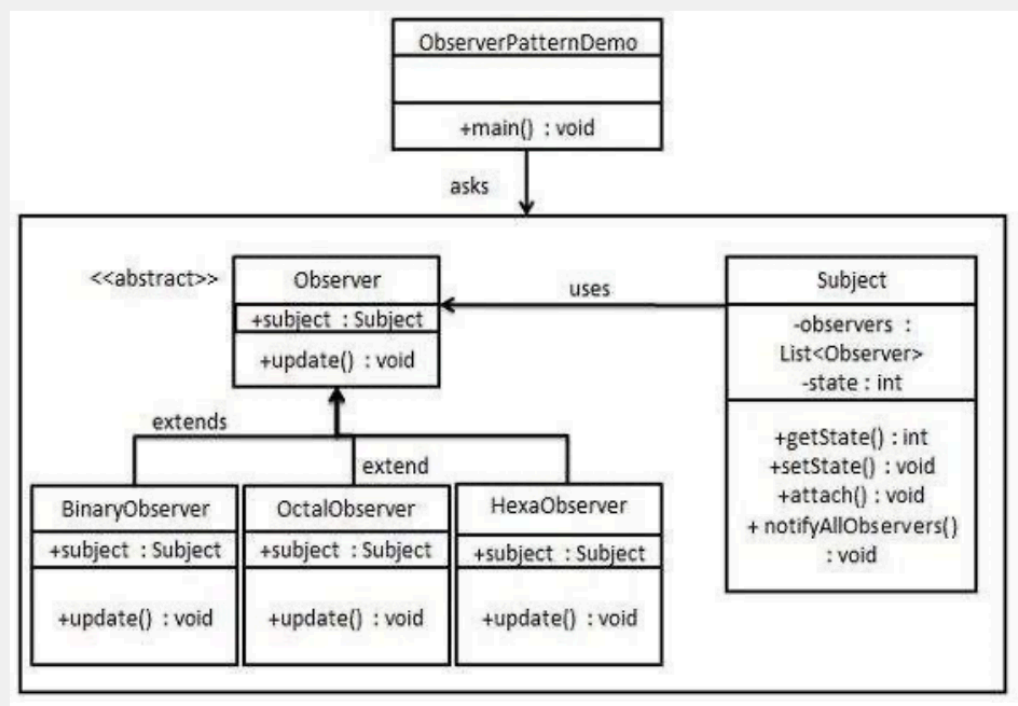
The reader should be able to understand the different entities in the system (class diagram), how they interact with respect to time (sequence), and the data that is moving throughout the system.

You will need to develop a **test plan** that will detail the requirements for testing.

Design Pattern Decisions

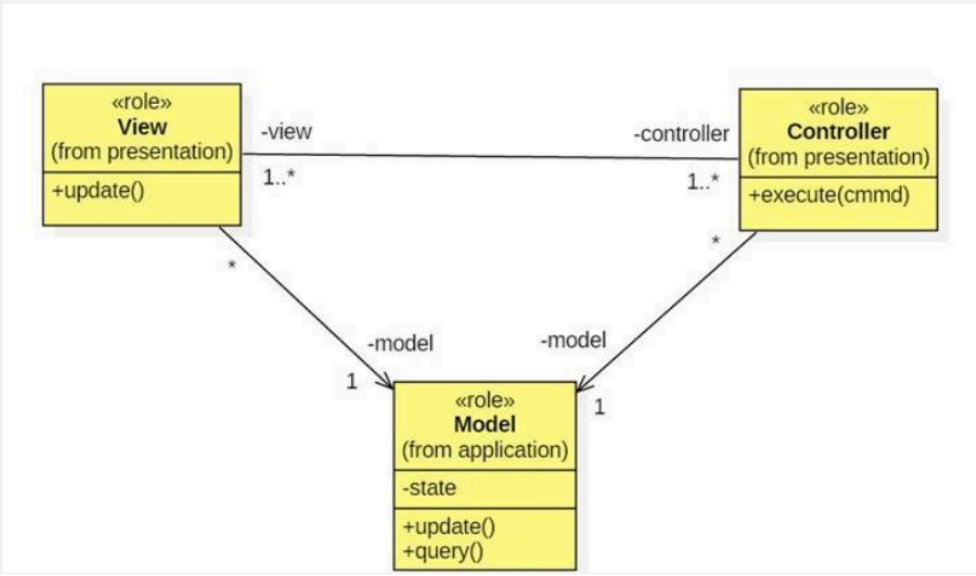
Observer:

The observer pattern is useful for the IoT web application because it allows for dynamic updating and communication between objects in the GUI and event-driven systems. This is particularly useful not only as sensor values change but as sensors are swapped. For this application there will be an abstract Subject class where a sensor object is the subject. For each subject the abstract Observer class will maintain a list of observers. These observers will include Alarm, Visualization, and Prediction classes. When the subject, or sensor, updates or is replaced the observers will then have an update() method that will be called.



MVC Pattern:

The Model-View-Controller pattern is commonly applied for web applications. In our use case it has the ability to separate the applications data, UI, and control logic into manageable modules. This means our UI will be built using flasks which will be separate from our data. Our data will be a separate class representing the queries and GET REQUEST interactions with thinkspeak. Finally, our control module will manage the sensors, alarms, and visualizations. By using MVC these separate concerns can be developed, tested, and maintained easier.


«role»
View
(from presentation)
+update()
-view
1..*
-controller
1..*
«role»
Controller
(from presentation)
+execute(cmmd)
*
*
-model
1
-model
1
«role»
Model
(from application)
-state
+update()
+query()