

Term Project Part 4: Detailed Design

The purpose of this exercise is to give you practice specifying design details. It will help you see how the various UML models come together for use by developers and give you practice specifying algorithms



February 21, 2024

Boston University

Mahim Choudhury

1/26/2024

Table of Contents

[6.1 Updated Class Model 2](#_Toc159597113)

[Figure 1: Updated class model 3](#_Toc159597115)

[A6.1 (ChatGPT regarding Updated Class Model) 4](#_Toc159597117)

[6.2 Activity Diagram 4](#_Toc159597118)

[Figure 2 Activity Diagram for monitoring safety 4](#_Toc159597119)

[6.2.1 Focused Sequence Diagram 5](#_Toc159597120)

[6.3 Pseudocode 6](#_Toc159597121)

[Appendix 7](#_Toc159597122)

[Evaluation 11](#_Toc159597123)

MET CS682: ASSIGNMENT 6

Term Project Part 4: Detailed Design

*Mahim Choudhury*

The purpose of this exercise is to give you practice specifying design details. It will help you see how the various UML models come together for use by developers and give you practice specifying algorithms.

1. OBSERVE LIMITS; 2. USE AI GENERATION: as before 3. OUTSIDE RESEARCH; 4. NAME YOUR FILE; 5. EVALUATION; 6. HINTS: as before

Continue developing your broader system focused on the *EmbeddedAI* sub-system by specifying UML design for it as required below, making the same assumptions listed in project parts 1-3 (assignments 3-5). Continue with the scope and specifications that you have already chosen for *EmbeddedAI*, *however f*eel free to extend the scope as needed by the demands of this assignment.

Your solution should focus on software-intensive aspects.

The last section contains numerous hints.

# Updated Class Model

Update your class model to a *detailed* design. Include classes, attributes, and methods appropriate for the activity diagram and pseudocode requested in the sections below. If additional classes are needed, try to limit them to two. It is not necessary to show all attributes and methods—only those that you consider important in communicating your detailed design. If possible, use colors to show updated design elements. In no more than three sentences give one example of how your updates have either supported or changed a design goal which you selected in the previous assignment.

A screenshot of a computer

Description automatically generated

Figure 1: Updated class model

As noted above by orange color code, only few methods were added with a new class named Motion. It inherits the Sensors class and works as class for Iot device. The addition of the embeddedAI is to clarify how some of the functions will be used in the activity and sequence diagram later. Notice I added new relationships between embddedAI class and two other packages i.e. Interaction and Detection. It illustrates how and what methods will be used to analyze interaction and detection data by the embeddedAI package within the system. These methods don’t deviate my original design goal rather enhances some features. The detectMotion() denoted in purple in the Detection class will be used later for activity diagram and pseudocode. The method from detection class analyses the motion detection and saves it. I believe the labels and verbs used in the class diagram are clear and understandable. Also, the methods and attributes are sufficient to carry out my original goal. The method does communicate with Sensor class and Safety class as mentioned in the class model.

## A6.1 (ChatGPT regarding Updated Class Model)

Show your most relevant prompt to chatGPT and the response which was most relevant to your solution.

ChatGPT didn’t recommend any changes.

# Activity Diagram

A diagram of a flowchart

Description automatically generated

### Figure 2 Activity Diagram for monitoring safety

The diagram above only focuses on detectMotion() method from the Detection class. If it gets data from the motion sensor it proceeds to the next step, otherwise the method exits. It checks again if the data received is valid or not by checking the camera and then proceeds to the next step. If the data is invalid it exits the method. Next up is the Analyze data portion which is done through the help of the microservice “EmbeddedAI” server. After analyzing the data, if there is any real detection, it lists them by order of occurrence and then send them to the safety class to process them by severity, ultimately ending the method’s job. If it detects there is no motion, it then exits the method without notifying the safety system. Note that I could have mentioned the notification class here, however, I think the detectMotion() doesn’t directly interact with it and thus there was no need for it.

## Focused Sequence Diagram

A diagram of a diagram

Description automatically generated with medium confidenceFigure 3: Focused Sequence Diagram

The Diagram above breaks down the “Analyze data” activity from the activity diagram. The scope is very limited and based on the requirements given, I tried to keep it as minimal as possible. It starts with shopwing how the Analyze data activity is triggered. It is only triggered when the camera sensor verifies the validity of the motion sensor data. Once verified, the Detection class sends the motion data to the AI analysis class it analyzes the motion data and sends it back. The Detection class then sends the analyzed data to the safety class where it can order them by occurrence and severity. What it does after that is out of Analysis activity and thus not mentioned in the diagram. Safety class also send an acknowledgement message to the Detection class that confirms the last detection was taken care of.

# Pseudocode

Provide pseudocode for the method that you outlined in 2. This method may call on other functions, possibly methods from objects of other classes.

Your pseudocode, including comments, should not exceed three quarters of a page (using 12-point type). To help you develop the logic, you may want to use design techniques you are now familiar with (use cases, sequence diagrams, specific design goals, etc.).

function detectMotion():

// Check if motion sensor data has been collected

IF NOT MotionSensorDataCollected THEN //check if any data was collected

return NULL; //efficiently exit the method without proceeding further.

ELSE

// Validate the collected data

IF NOT validateMotionData(): //if the data is not validated

return NULL; //efficiently exit the method without proceeding further.

ELSE

// Analyze the motion sensor data

motionDetected = analyzeMotionData(MotionSensorData)

// Check if motion is detected

IF motionDetected:

listMotionOccurrences(MotionSensorData) // List data by occurrence

SafetySystem.sendMotionData(MotionSensorData) // Send the motion data to the safety system

ELSE

Return NULL;

ENDIF

ENDIF

ENDIF

# Appendix

Since this Term Project Part 4 builds on Parts 1 - 3, provide the overall mission statement here. If you like, you may also provide prior parts you feel are helpful in supporting your work above. This section is not graded.

Scope:

Scenario:

Imagine a Smart Home Automation System designed to manage various aspects of home living, such as lighting, temperature control, security, and appliance automation. This system utilizes IoT devices and sensors to enable remote monitoring and control via mobile apps or voice commands.

Scope: The scope entails integrating EmbeddedAI into an existing Smart Home Automation System to enhance user experience, efficiency, and security. This integration will leverage generative AI like ChatGPT and IoT elements to provide personalized assistance, intelligent automation, and proactive home management capabilities. Key focus areas include personalized home assistance, natural language interaction, predictive home automation, and enhanced security and safety measures.

The project will involve analyzing user behavior, preferences, and environmental data collected by IoT sensors to offer tailored recommendations and automate routine tasks. It will enable natural language interactions with EmbeddedAI, allowing users to communicate with their smart home system using voice commands or text input. Predictive capabilities will anticipate user needs and potential issues, while enhanced security measures will detect anomalies and trigger automated responses to ensure the safety of the home environment.

Initial Use Case:

|  |  |  |
| --- | --- | --- |
| **Use case Name** | Pet activity monitoring | |
| **Actor:** | Pet owner | |
| **Description:** | The integrated EmbeddedAI system can aid the smart home system to empower the pet owners with the ability to leave their pets home without worry. AI-enabled monitoring system enhances pet safety and allows pet owners to stay connected with their pets, even when they are not physically present at home. | |
| **Pre-condition:** | The system is online and connected to IoT devices. | |
| **Step #** | **Actor** | **System** |
| **1** | Pet Owner activates pet monitoring mode | System initiates AI-enabled pet monitoring functionalities, including smart cameras, motion sensors, and environmental sensors, to track pet movements and behaviors. |
| **2** | Pet Owner receives activity alert | System utilizes AI algorithms to detect pet movements, sounds, or irregular behaviors, and sends real-time alerts and notifications to the pet owner's mobile device. |
| **3** | Pet Owner views live monitoring feed | System provides access to live video streams and environmental data through the mobile app, allowing the pet owner to observe their pets' activities and surroundings remotely. |
| **4** | Pet Owner interacts with pets remotely | System enables the pet owner to engage with their pets through interactive features, such as remotely operated toys, treat dispensers, or two-way audio communication channels. |
| **5** | Pet Owner ensures pet safety | System allows the pet owner to monitor environmental conditions, such as temperature, humidity, and air quality, and take necessary actions to ensure the safety and comfort of their pets. |
| **Alternate Courses:** | [System Alt 6] If the system detects unusual behavior or potential hazards, it alerts the pet owner and provides guidance on mitigating risks or contacting emergency services.  [System Alt 7] If the pet owner is unavailable to respond to alerts, the system can activate pre-configured safety protocols, such as contacting designated emergency contacts or veterinary services. |  |

4.4 Documenting Classes and Relationships

Using the format below, select three most important nontrivial business classes and one non-business design class from the class model and the format below, explain its importance for your design, and its relationships with other classes. Select classes which are complex enough to require explanation.

4.4.1 First Business Class Selected: User

Importance for the Design: The “User” class defined in the class model diagram is one of most important business classes in the system. Although the main beam of light in this system is focused around the implementation of Iot devices and AI analysis, ultimately the destination is the User. User class defines what the system should know about the human users of it. User has their own id, username. It also allows them to create new users and update their information. The class has some relationships with other important classes which will be described later. It is the root of regular user, emergency contact, interactions with pet, and selecting what dispenser to use for interaction. Although the AI can help in this area, I still believe the pets need their human friend to interact with them instead of a cloud.

Relationship with other classes: The “User” class in the class diagram shows that it is inherited by two other sub classes named “EmergencyContact” and “Interaction”. From my previous experience, I could tell the inheritance of classes means they are the same class with additional features. Meaning the EmergencyContact and interaction classes already have the attributes of “User” class and then they have their own unique attributes and behavior. In this case, the interaction class have it’s own ID and the EMergencyContact has two more attributes, Email and Cell and their behaviors. For example, each object of Interaction class can have it’s own specific userID, username, and type id. Similarly, EmergencyContact can have it’s userID, username plus the email and cell per instances of the class itself. Interaction class has it’s own relationship with “Dispenser” class, which is also a branch coming out of the root “User” class.

4.4.2 Second Business Class Selected: Sensors.

Importance for the Design: The “Sensors” class is another important business class for. For this system, it works as the hub class for all the IoT devices to provide the data to the AI for analysis. The class is designed to hold information about the IoT devices such as Id, location, and behaviors like activate(), deactivate(). There could be more attributes and behaviors, however, we are only focusing on some. It is clearly visible in the class diagram that this class has many relationships with some other important classes in the system. This emphasizes the importance of this class in this system. Although it can connect to many other IoT devices, the diagram is only showing the thermostat, and camera as the main two sensors. The relationships are described in the next section.

Relationship with other classes: The “Sensors” class has aggregation relationship with Thermostat and Camera classes. From my understating from past learning, aggregation relationship means that on class is part of or has another class. For the Thermostat, it can control one and only one thermostat and a thermostat can only be controlled by that one sensor class. I chose only one thermostat because in most houses, there is only one thermostat to control the temperature. For the Camera class however, the Sensor class can control 1 to many cameras since a house can have many of them. Cameras can only be controlled by that one Sensor class though. The Aggregation relationship also tell us that the Thermostat and Camera classes can exist without the Sensors class. It has the same relationship with “PetMonitoring” class. However, the “detection” class cannot exist without the Sensor class and therefore, it has a composition relationship with it.

4.4.3 Third Business Class Selected: PetMonitoring.

Importance for the Design: The “PetMonitoring” class is another most important business class for this system. This class works as the main brain of the pet monitoring feature of the smart home system. It connects to various important classes and works as the manger. It holds information of the pet(s), monitors their activity, detects anomalies, checks the system health, etc. Without the “PetMonitoring”, the main purpose of this pet monitoring feature of smart home system will not function as required. It is required to communicate with other classes to gather data and work with it. The many relationships it has with other classes are described in the section below.

Relationship with other classes: The “PetMonitoring” class has aggregation relationship with “Sensors”, “Safety”, “User”, “Pet”. This way, it can get information from them and their related classes. It can have 1 to many Sensors, Users, and Pets. Those classes however, can only be associated with only one “PetMonitoring” class. The “PetMonitoring” class can be associated with 0 to 1 Safety class. The reason I chose 0 to 1 is because I think even without the information from safety class, it can still function with the activity detected from the sensors and keep recording it. On the other hand, the “Safety” class should only be associated with the on and only “PetMonitroing” class.

4.4.4 Non-Business Class Selected: <<Design>>.

Importance for the Design: As depicted by “<< >>”, the Design class is the non-busineess class for this system. It is pictured as the main home screen from where the user can interact with the system to enable pet monitoring mode. It can also display valuable information such as graphs of previous usage, pets weekly sleep time, etc from previous monitoring. Although it is a non-business class, it is important to the user from user interface of the pet monitoring system’s standpoint. It determines how user-friendly and easy to use pet monitoring feature would be to the users. It has a dependency with “PetMonitoring” class which is described in the section below.

Relationship with other classes: The dependency described in the class model is pretty self-explanatory. It displays the “PetMonitoring” class. As mentioned earlier, it need the data from the “PetMonitoring” class in order to display the information in the GUI. “PetMonitoring” class gather the data from various other classes across the design but it all needs be presented to the user through he non business class <<Design>>.

# Evaluation

