

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](#_Toc159510352)

[Figure 1: Updated class model 3](#_Toc159510354)

[A6.1 (ChatGPT regarding Updated Class Model) 3](#_Toc159510356)

[6.2 Activity Diagram 4](#_Toc159510357)

[Figure 2 Activity Diagram for monitoring safety 5](#_Toc159510358)

[6.2.1 Focused Sequence Diagram 6](#_Toc159510359)

[6.3 Pseudocode 6](#_Toc159510360)

[Appendix 6](#_Toc159510361)

[References 6](#_Toc159510362)

[Evaluation 7](#_Toc159510363)

[Hints 8](#_Toc159510364)

[Overall Assignment Notes and Grading Criteria 8](#_Toc159510365)

[Class Model 8](#_Toc159510366)

[Activity Diagram 8](#_Toc159510367)

[Pseudocode 9](#_Toc159510368)

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

Select a non-trivial method (function) from your class model above and draw an activity diagram for it. The diagram may reference other functions, possibly from objects of other classes. The latter should appear in your class model; however, you do not need to elaborate on these referenced methods beyond their (expressive!) names and, if needed, clarifying comments. Consider numbering the activities to coincide with the pseudocode steps described in Section 3.

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

Select a non-trivial activity from your activity model above and draw a sequence diagram for it. The scope should be limited to no more than 3 message calls. The diagram will reference other functions from objects of other classes and should appear in your class model. You may place the sequence diagram in the Activity Diagram in section 6.2 next to the activity you are expanding.

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.).

Your pseudocode replaces this.

# Appendix

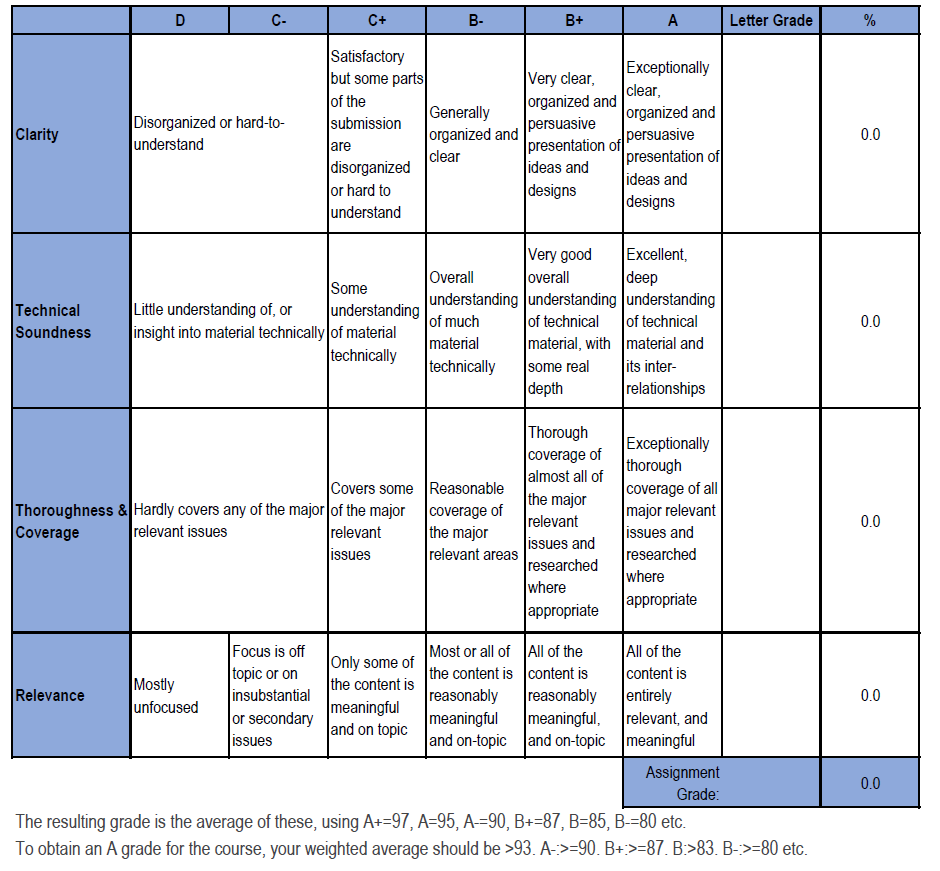
Since this Term Project Part 4 builds on Parts 1 - 3, provide the overall mission statement here. If you like, you may also provide prior parts you feel are helpful in supporting your work above. This section is not graded.

# References

[1] your first reference replaces this

[2] …

# Evaluation



**Please do not include Hints section in your solution.**

# Hints

## Overall Assignment Notes and Grading Criteria

* You may use Visio, LucidChart, or another design tool of your choice (please check with your facilitator in advance if not using Visio or LucidChart).
* The module notes are a primary source for explanations and examples; we also encourage you to do outside reading and research to gain additional perspective.
* **Clarity:** Are diagrams clear to read? (e.g., avoid overlapping lines, non-polished designs).
* **Relevance**: Make sure to consider the right scope for the activity diagram and pseudocode. This is not system or use case level such as one you put together previosuly. This is specific to a single method which you have identified as important in your design to expand on.
* **Relevance**: We develop activity diagrams and pseudocode to help us with important and complex functionality. Make sure to choose a significant method for the activity diagram/pseudocode. Significant means something important to the application you are designing and your design (i.e. something you have been focusing on all along, or something that you have expanded on in this assignment).
* **Thoroughness and Coverage:**
  + Refer the parts back to design goals at every opportunity. For example, if you add a class, make sure it does not contradict goals of design you have selected previously.
  + Consider what to place in the Appendix. For example, it may be helpful to add a sequence diagram to complement your activity diagram and pseudocode—make sure to choose appropriate scope.
* **Consistency and Clarity:** Is your design consistent? For example, all function calls and attributes listed in pseudocode are provided in class model.

## Class Model

* **Clarity:** Is it clear how your added (if applicable) classes/methods/attributes relate to the existing model? Add explanatory notes if not obvious. For example, do these new additions appear in the pseudocode?
* **Technical Soundness:** At this point, cohesion, coupling, and encapsulation choices really matter. Make sure the right attributes and methods belong to the right class, explain what may not be obvious.
* Additional hints and notes on approaching class modeling are covered in Assignment 4.

## Activity Diagram

* A good place to start is to review the “Activity Diagrams” section in Module 4.
* Activity diagrams are discussed on pages 129-138 in the text.
* It may be helpful to develop/update a use case for the function you decide to work with, which you are welcome to include in the Appendices section.
* **Clarity**: Make sure your diagram is clearly labeled, e.g., are your decision points clear?
* **Technical Soundness:** Don't confuse activity diagrams with state diagrams—you can show parallel processes and decision choices. They are not the same as data flow diagrams either, because activity diagrams indicate the flow of control rather than of data.
* **Thoroughness and Coverage:** Does your activity diagram cover all reasonable branches?  This is a good opportunity to add robustness.

## Pseudocode

* Module 6 provides two ways to approach writing Pseudocode within the “Specifying Methods and Algorithms in Detail” section.
* It may be helpful to develop a use case or sequence diagram for the function outlined in part 2, which you are welcome to include in the Appendices section.
* **Clarity:** 
  + Pseudocode is flexible; if you introduce something that is not obvious (i.e. syntax that is Java-like, Python-like, SQL-like) provide an explanation (i.e. "Note to reader "++" means increment variable by 1")
  + Use colors and/or indentations to distinguish various parts of your pseudocode.
* **Technical Soundness:** Review and understand loops, If/Then, and case statements, and know when to use one over another.
* **Depth and Coverage:** Practice both verbal and arithmetic expressions in your pseudocode.