PEC 1

Introduction to Software Engineering and Object Orientation

Alejandro Pérez Bueno

Oct 18, 2024

Table of Contents

Self-Responsibility Declaration

I certify that I have completed CAT1 completely individually and only with the help that the teaching staff of this subject considers appropriate, according to the instructions explained in the "Originality in the evaluation" section of the classroom. I understand that non-original work and/or the use of generative AI will mean that the submitted activity will not be marked and will automatically be assigned a grade of 0.

Module 1 Questions

Question 1

1. Model the main classes of the system following the defined architecture.

- The role is *architect* and the activity is *Modeling* because the architect is the one in charge of defining the outline of a system, which includes the main classes of the system.

2. Refine and validate the identified requirements, from the system domain's perspective.

- The role is *Functional Analyst* and the activity is *Requirements identification and management* because he is responsible for unifying the different views of the domain in a single model that is clear, concise, and consistent.

3. Plan and estimate project tasks.

- The role is *Project Manager* and the activity is *Project Management* because he is the one who has to organize the project to make sure every aspect of it goes as planned.

4. Develop the application code from the documents made by analysts.

- The role is *Programmer* and the activity is *Construction and testing* because a programmer must write the code for the application, following the designs appointed by analysts.

5. Compile and package the code of the new version of the application and distribute it to the corresponding servers.

- The role is *Deployment Manager* and the activity is *Maintenance and Re-engagning* because he is in charge of packaging a new version of a software coded by programmers into a server.

Question 2

Project 1

- **Type of Project:** Group 1. This project has a known solution and clear objective, as it involves updating a critical system with stable requirements and strict protocols.
- **Development Method:** Classical or Waterfall Life Cycle. This method is ideal for projects with clear objectives and known solutions, ensuring stable requirements and minimal changes.

Project 2

- **Type of Project:** Group 2. The project has a known solution but an unclear objective, as it involves developing an episodic game with potential changes based on user feedback.
- **Development Method:** Iterative and Incremental Life Cycle. This method supports periodic releases and adjustments, making it suitable for evolving solutions and refining features iteratively.

Project 3

- **Type of Project:** Group 4. This project has a little-known solution and unclear objective, as Júlia is exploring WordPress and WooCommerce with minimal investment.
- **Development Method:** Lean and Agile Development. This method is ideal for high uncertainty, allowing experimentation and quick adjustments, focusing on flexibility and responsiveness.

Question 3

A Type 4 project could be the development of a new wearable device aimed at improving mental well-being. The objective is unclear as it explores how wearable technology can impact mental health, a field with evolving research. The solution is also uncertain, involving experimentation with sensors and algorithms to determine effective monitoring methods. Extensive research is needed to identify beneficial metrics like stress levels or sleep patterns. This innovative project requires flexibility and adaptability, fitting the Type 4 classification due to its undefined objectives and solutions.

Module 2 Questions

Question 4

1. Employee (Abstract Class)

a. Subclasses: - Manager - Developer

b. Association: - Project: An Employee works on one or more Projects, and a Project has one or more Employees.

c. Attributes: - Employee: employeeID - Manager: department - Developer: programmingLanguage

d. Instances: - Employee: {employeeID: “E123”} - Manager: {employeeID: “M456”, department: “Sales”} - Developer: {employeeID: “D789”, programmingLanguage: “Java”}

2. Pet

a. Subclasses: - Dog - Cat

b. Association: - Owner: A Pet is owned by one and only one Owner, and an Owner can have any number of Pets.

c. Attributes: - Pet: name - Dog: breed - Cat: furColor

d. Instances: - Pet: {name: “Buddy”} - Dog: {name: “Max”, breed: “Labrador”} - Cat: {name: “Whiskers”, furColor: “Tabby”}

Question 5

1. Visibility of `title` in `Program`

- Visibility: Protected
  - Justification: Allows access by subclasses (`Movie`, `Series`) but not by unrelated classes (`Director`).

2. Visibility of `nationality` in `Director`

a. Visibility: - Private - Justification: Accessible only within the `Director` class.

b. Future Subclasses: - No Change - Justification: Remains private; change to protected if access by subclasses is needed.

3. Visibility of `numberSeasons` in `Series`

- Visibility: Public
  - Justification: Allows access by all classes (`Director`, `Movie`, `Series`, `Program`).

4. Association between `Director` and `Movie`

- Association: A Director directs one or more Movies, and a Movie is directed by one Director.
- Multiplicity: Director: 1 to many Movies, Movie: 1 Director
- Polymorphic Association: No
  - Justification: Not involving superclass-subclass relationship.

5. Relationships for `Chapter` and `AnimationMovie`

a. `Chapter` linked to `Series`: - Mechanism: Association - Justification: Represents a part-whole relationship.

b. `AnimationMovie` linked to `Movie`: - Mechanism: Inheritance - Justification: Represents a specialized "is-a" relationship.

Question 6

a) Classes and Attributes

1. **Officer (Abstract Class)**
   - Attributes:
     - identifierCode
     - firstName
     - lastName (one or two)
     - dateOfBirth
2. **Inspector (Concrete Class, Subclass of Officer)**
   - Attributes:

    - dateOfPromotion
3. **PoliceStation (Concrete Class)**
    - Attributes:
        - name
        - location
4. **Case (Concrete Class)**
    - Attributes:
        - identifierCode
        - codeName
        - startDate
        - closureDate (optional)

b) Associations

1. **WorksAt:**
    - Associates each instance of Officer with one and only one instance of PoliceStation and each instance of PoliceStation with one or more instances of Officer.
    - Justification: Not polymorphic.
2. **AssignedTo:**
    - Associates each instance of Case with one or more instances of Officer and each instance of Officer with any number of instances of Case.
    - Justification: Not polymorphic.
3. **ResponsibleFor:**
    - Associates each instance of Case with one and only one instance of Inspector and each instance of Inspector with any number of instances of Case.
    - Justification: Not polymorphic.
4. **RelatedCases:**
    - Associates each instance of Case with any number of instances of Case and vice versa.
    - Justification: Not polymorphic.