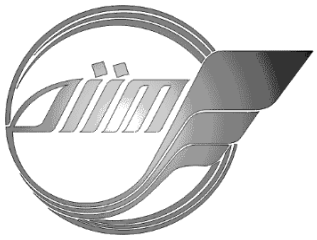
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



Дніпровський національний університет
залізничного транспорту імені академіка В. Лазаряна

Кафедра «Комп'ютерні інформаційні технології»

Лабораторна робота №2

з дисципліни «Архітектура та проектування програмних засобів»

на тему: «Основи структурного моделювання. Діаграма класів.»

Виконав:
студент гр.ПЗ1911
Сафонов Д. Є.
Прийняла:
Куропятник О. С.

Дніпро, 2020

Тема. Основи структурного моделювання. Діаграма класів.

Мета. Вивчити: способи подання класів і базових зав’язків між ними; прийоми моделювання сутностей і зав’язків між ними. Отримати практичні навички з моделювання предметної області за допомогою діаграми класів.

Постановка задачі згідно загального та індивідуального завдання.

Виконати структурне моделювання ПЗ відповідно завданню лабораторної роботи №1(Автопілот для легкового автомобіля на дорогах загального користування).

Визначити сутності предметної області, їх атрибути, операції. Специфікувати класи сутностей у вигляді CRC-карток. Визначити та обґрунтувати зв’язки між класами. Побудувати діаграму класів.

Виконати аналіз застосованих проектних рішень.

Опис сутностей з визначенням їх атрибутів та операцій.

Ui:

Поєднує усі об’єкти інтерфейсів користувача, а саме:

Settings_main

Trip_chooser

Steering_wheel

Cam

Pedals

Controller:

Дає змогу зв’язуватися об’єктам наступних класів:

Driver

Database

Ui

Database:

Зберігає налаштування авто

User:

Клас, який представляє режим користувача.

Специфікація класів сутностей у вигляді CRC-карток.

Ui

Базовий клас	Похідні класи (нащадки)
-	-
Обов’язки	Зв’язки
Об’єднати усі класи інтерфейсу користувача	Settings_main, Trip_chooser, Cam, Steering_wheel, Pedals, Controller

Gui_Scene

Базовий клас	Похідні класи (нащадки)
-	Menu, Trip_Chooser
Обов’язки	Зв’язки
Базовий клас графічного інтерфейсу	Menu, Trip_Chooser

Menu

Базовий клас	Похідні класи (нащадки)
Gui_scene	Settings_main
Обов’язки	Зв’язки
Повертати обраний пункт меню	Gui_Scene, Settings_main

Trip_chooser

Базовий клас	Похідні класи (нащадки)
Gui_scene	-
Обов’язки	Зв’язки
Вибір подорожі	Gui_scene, Ui

Settings_main

Базовий клас	Похідні класи (нащадки)
Menu	-
Обов’язки	Зв’язки
Перехід за обраним пунктом меню	Menu, Ui

Analog_ui

Базовий клас	Похідні класи (нащадки)
-	Steering_wheel, Cam, Pedals
Обов’язки	Зв’язки
Повертати релевантне значення фізичного інтерфейсу	Steering_wheel, Cam, Pedals

Steering_wheel

Базовий клас	Похідні класи (нащадки)
Analog_ui	-
Обов’язки	Зв’язки
Повертати релевантне значення повороту руля	Ui, Analog_ui

Cam

Базовий клас	Похідні класи (нащадки)
Analog_ui	-
Обов’язки	Зв’язки
Повертати релевантну картинку з камер	Ui, Analog_ui

Pedals

Базовий клас	Похідні класи (нащадки)
Analog_ui	-
Обов’язки	Зв’язки
Повертати релевантну силу натиску на педалі	Ui, Analog_ui

Controller

Базовий клас	Похідні класи (нащадки)
-	-
Обов’язки	Зв’язки
Зв’язок між User, Ui, Database	Database, Driver, Ui

Driver

Базовий клас	Похідні класи (нащадки)
-	Controller, Low_assistance
Обов’язки	Зв’язки
Дозволити водію повністю контролювати авто	Controller, Low_assistance, Controller, Wheel_interface, Engine_interface

Low_assistance

Базовий клас	Похідні класи (нащадки)
Driver	High_assistance
Обов’язки	Зв’язки
Дозволити водію контролювати авто, та допомагати у простих ситуаціях	Driver, High_assistance

High_assistance

Базовий клас	Похідні класи (нащадки)
Low_assistance	Developer
Обов’язки	Зв’язки
Дозволити водію контролювати авто, та допомагати у простих ситуаціях, та забирати контроль у критичних	Low_assistance, Developer

Passenger

Базовий клас	Похідні класи (нащадки)
Driver	Developer
Обов’язки	Зв’язки
Дозволити пасажиру обрати кінцеву зупинку	Driver, Developer

Developer

Базовий клас	Похідні класи (нащадки)
High_assistance, Driver, Passenger	-
Обов’язки	Зв’язки
Дозволити розробнику модифікувати та тестувати систему	High_assistance, Passenger

Database

Базовий клас	Похідні класи (нащадки)
-	-
Обов’язки	Зв’язки
Зберігати налаштування	Controller

Engine_interface

Базовий клас	Похідні класи (нащадки)
-	-
Обов’язки	Зв’язки
Інтерфейс двигуна	Driver

Wheel_interface

Базовий клас	Похідні класи (нащадки)
-	-
Обов’язки	Зв’язки
Інтерфейс колісної бази	Driver

Опис та обґрунтування зв’язків між класами.

Gui_scene та Menu — паттерн “Компонувальник”

Gui_scene та Trip_chooser — наслідування через те, що існують спільні операції із класом Menu.

Settings_menu та Menu — Наслідування через те, що існують додаються окремі операції до існуючих у Menu.

Analog_ui та Steering_wheel, Cam, Pedals — наслідування через те, що три класи мають спільні атрибути та операції.

Ui та Steering_wheel, Cam, Pedals, Settings_main, Trip_chooser — об’єкти класу об’єднуються в один через спільне призначення.

Controller та Ui — залежність через те, що Ui оновлює налаштування у Controller.db

Controller та Driver — агрегація через те, що користувач може змінити режим, тим самим змінюючи тип, тим самим змінюючи об’єкт.

Driver та Low_assistance, Passenger — наслідування через те, що додаються нові атрибути. Також усі класи наслідувані від Driver мають керівничий метод update, який може переписуватися у наслідуваних класах, але вона завжди відповідає за контроль двигуна та колесної бази.

High_assistance та Low_assistance — наслідування через те, що додаються нові атрибути.

Developer та Passenger, High_assistance — наслідування через те, що Developer має мати доступ до методів обох класів(за потреби можна зробити обидва класи атрибутами розробника, і прибрати таким чином подвійне наслідування, для мов програмування, які його не підтримують)

Controller та Wheel_interface, Engine_interface — Керування обома класами через клас Driver та його нащадків.

Діаграма класів.

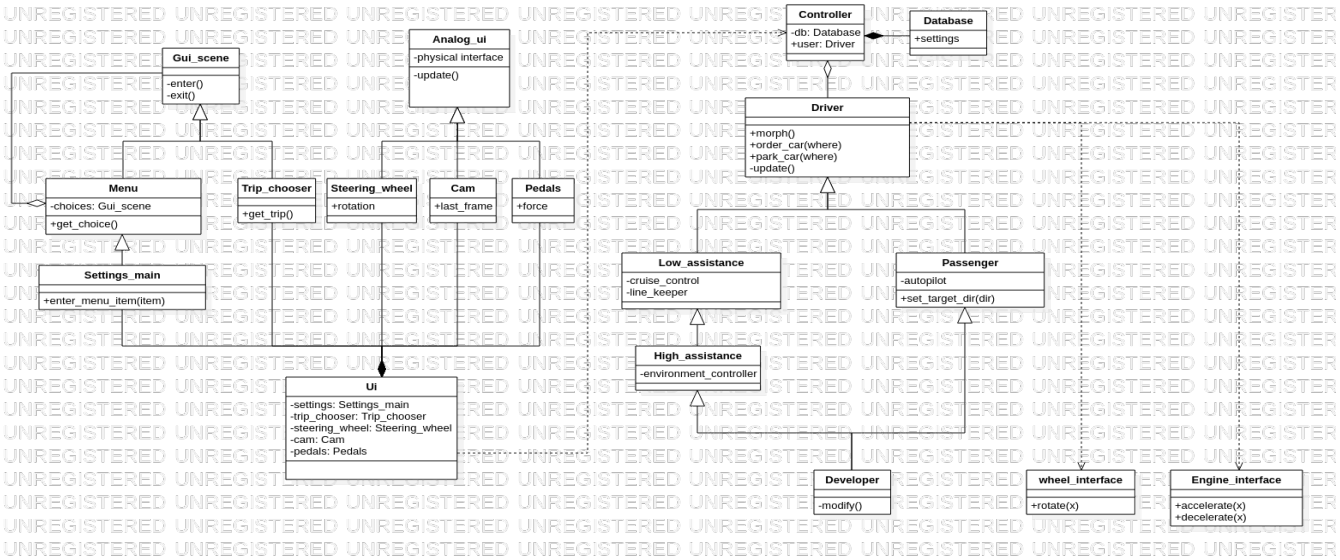


Рисунок 1

Аналіз результатів та висновки щодо переваг та недоліків у використанні засобів UML для специфікації ПЗ, а також відносно особливостей прийнятих проектних рішень.

Одна з найбільших переваг у використанні UML — дуже докладний розбір архітектури ПЗ. Найбільший недолік — час розробки. Але як і завжди це компроміс. Також до недоліків можна віднести те, що потрібно вчити нові правила. Щодо особливостей прийнятих рішень: я розділив програму на три частини: Інтерфейс користувача, контролер та маленька база даних для збереження налаштувань користувача. Усі події у інтерфейсі оброблюються через користувача контролера. Найбільша перевага — можна розробити з нуля модель досить компактну та зрозумілу модуль ПЗ, для реалізації найпростішими інструментами.