МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»
Завідувач кафедри
_____ проф. Приходько С.Б.
«___» _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «магістр»

на тему: Математична модель оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів та програма для її реалізації

Виконав: студент групи 6151м
_____ Давидов Д.В.
(підпис, ПІБ)

Керівник роботи:
доцент кафедри ПЗАС, к.ф.-м.н.
(посада, науковий ступень вчене звання)
_____ Латанська Л.О.
(підпис, ПІБ)

Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____проф. С.Б.Приходько
(підпис)
« 27 » жовтня 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «магістр»

Студенту Давидову Денису Віталійовичу
(Прізвище, ім'я, по батькові)

1. Тема роботи: Математична модель оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів та програма для її реалізації
Керівник роботи к.ф.-м.н., доц. Латанська Л.О.
Затверджені наказом ректора № 1222-УЧ від « 15 » _____10_____ 2025 р.
2. Термін подання роботи: 08.12.2025 р.
3. Вихідні дані по роботі: _____

4. Перелік питань, що належать до розробки (найменування розділів) _____
Вступ (Актуальність теми. Зв'язок роботи з науковими програмами, планами, темами. Мета і завдання дослідження. Об'єкт дослідження. Предмет дослідження. Методи дослідження. Наукова новизна одержаних результатів. Практичне значення одержаних результатів. Особистий внесок здобувача. Апробація результатів досліджень**.** Публікації.); Огляд літератури та обґрунтування необхідності проведення досліджень за обраною темою; Викладення результатів власних досліджень з висвітленням того нового, що пропонується; Розділ з розробки програмного забезпечення; Організаційно-економічний розділ; Розділи з охорони праці; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма та методика випробувань програмного забезпечення)

5. Перелік презентаційних матеріалів Титульний слайд. Мета та завдання дослідження. Об'єкт та предмет дослідження. Наукова новизна одержаних результатів. Практичне значення одаржаних результатів. Особистий внесок здобувача. Апробація результатів дослідження та публікації. Аналіз існуючих моделей оцінювання складності об'єктно-орієнтованого проєктування. Рівняння еліпсоїда передбачення для нормалізованих метрик RFC, WMC та LCOM Spring Framework застосунків. Постановка задачі на розробку проєкту програмного забезпечення. Діаграма варіантів використання системи. Технічний проєкт програмного забезпечення. Вибір інструментів розробки. Графічне вікно з елементами інтерфейсу. Результати випробування програмного забезпечення. Висновки._____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 27.10.2025 р. _____

КАЛЕНДАРНИЙ ПЛАН

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу ВСТУП	28.10.2025	ДП*
2.	Підготовка розділу з огляду літератури та обґрунтування необхідності проведення досліджень за обраною темою	30.10.2025	ДП*
3.	Підготовка розділу (ів) за результатами власних досліджень	01.11.2025	ДП*
4.	Підготовка розділу з розробки програмного забезпечення	26.11.2025	
5.	Підготовка організаційно-економічного розділу	28.11.2025	
6.	Підготовка розділу з охорони праці	02.12.2025	
7.	Підготовка розділу з охорони навколишнього середовища	03.12.2025	
8.	Підготовка розділу ВИСНОВКИ	04.12.2025	
9.	Оформлення списку використаних джерел та додатків	05.12.2025	
10.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версії роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	08.12.2025	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку)
11.	Підготовка презентації та доповіді	12.12.2025	
12.	Попередній захист роботи на засіданні кафедри ПЗАС	15.12.2025	
13.	Подання на кафедру ПЗАС електронних версії наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді	19.12.2025	

* - за результатами (розділ звіту) дослідницької практики (ДП), яка була з 01.09.2025 по 26.10.2025 р.

Студент _____ Давидов Д.В.
(підпис) (ПІБ)

Керівник роботи _____ Латанська Л.О.
(підпис) (ПІБ)

РЕФЕРАТ

Давидов Денис Віталійович

«Математична модель оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів та програма для її реалізації»

Кваліфікаційна робота на здобуття освітнього рівня магістра зі спеціальності 121 – «Інженерія програмного забезпечення». Національний університет кораблебудування імені адмірала Макарова. Миколаїв, 2025 р.

Обсяг роботи: 101 стор., 9 табл., 17 рис., 37 використаних джерела, 5 додатків.

Актуальність теми роботи: побудова моделі, зазначеної у темі кваліфікаційної роботи, дозволить адаптувати підхід до оцінювання складності об’єктно-орієнтованого проєктування для систем, розроблених із використанням Spring Framework, розширивши існуючі методи аналізу за рахунок урахування специфіки семантики класів у середовищі Java, що сприятиме підвищенню точності, надійності та практичної цінності результатів оцінювання.

Мета та завдання дослідження: метою є підвищення достовірності оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів. Завдання дослідження: проаналізувати існуючі математичні моделі для оцінювання складності об'єктно-орієнтованого проєктування; побудувати математичну модель для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів з використанням нормалізуючого перетворення; розробити програмний інструмент для реалізації побудованої моделі.

Об’єктом дослідження є процес оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів.

Предметом дослідження є математична модель для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів.

Методи дослідження. У вирішенні поставлених завдань використано методи теорії ймовірності, математичної статистики, багатовимірного статистичного аналізу.

Наукова новизна одержаних результатів. Побудовано рівняння еліпсоїда передбачення для метрик RFC,WMC та LCOM на рівні застосунку, нормалізованих тривимірним перетворенням Бокса-Кокса із урахуванням викидів у даних, для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів. Побудоване рівняння, у порівнянні з існуючими математичними моделями, враховує кореляцію між метриками RFC, WMC та LCOM, що дозволяє підвищити достовірність оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів.

Практичне значення одержаних результатів. Розроблено програму для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів, використовуючи фреймворк PySide6.

Апробація результатів досліджень. Результати досліджень пройшли апробацію на VI Міжнародній науково-практичній інтернет конференції «Інформаційні технології: моделі, алгоритми, системи – 2025» (Миколаїв, 16-17 листопада 2025 р.).

Публікації. За результатами кваліфікаційної роботи зроблено одну публікацію, а саме матеріали Міжнародної науково-практичної інтернет конференції.

Ключові слова. еліпсоїд передбачення, складність ООП, семантика класів, Java застосунки, Spring Framework.

ABSTRACT

Davydov Denis Vitaliyovych

"A Mathematical model for estimating the complexity of object-oriented design of Spring Framework systems based on class semantics and software for its implementation"

Qualification work for obtaining a master's degree in specialty 121 - "Software Engineering". Admiral Makarov National University of Shipbuilding. Mykolaiv, 2025

Scope of work: 101 pages, 9 tables, 17 figures, 37 sources used, 5 appendices.

Relevance of the topic of the work: building the model specified in the topic of the qualification work will allow adapting the approach to assessing the complexity of object-oriented design for systems developed using the Spring Framework, expanding existing analysis methods by taking into account the specifics of class semantics in the Java environment, which will contribute to increasing the accuracy, reliability, and practical value of the assessment results.

Purpose and objectives of the study: the goal is to increase the reliability of assessing the complexity of object-oriented design of Spring Framework systems based on class semantics. Research objectives: to analyze existing mathematical models for assessing the complexity of object-oriented design; to build a mathematical model for assessing the complexity of object-oriented design of Spring Framework systems based on class semantics using a normalizing transformation; to develop a software tool for implementing the constructed model.

The object of the study is a process of assessing the complexity of object-oriented design of Spring Framework systems based on class semantics.

The subject of the study is a mathematical model for estimating the complexity of object-oriented design of Spring Framework systems based on class semantics.

Research methods. In solving the tasks set, methods of probability theory, mathematical statistics, and multivariate statistical analysis were used.

Scientific novelty of the results obtained. A prediction ellipsoid equation is constructed for application-level metrics RFC, WMC, and LCOM, normalized by a three-dimensional Box-Cox transformation taking into account outliers in the data, to assess the complexity of object-oriented design of Spring Framework systems based on class semantics. The constructed equation, in comparison with existing mathematical models, takes into account the correlation between the metrics RFC, WMC, and LCOM, which allows to increase the reliability of assessing the complexity of object-oriented design of Spring Framework systems based on class semantics.

Practical significance of the obtained results. A program has been developed to assess the complexity of object-oriented design of Spring Framework systems based on class semantics, using the PySide6 framework.

Testing of the research results. The research results were tested at the VI International Scientific and Practical Internet Conference " Інформаційні технології: моделі, алгоритми, системи – 2025" (Mykolaiv, November 16-17, 2025).

Publications. Based on the results of the qualification work, one publication was made, namely the materials of the International Scientific and Practical Internet Conference.

Keywords. prediction ellipsoid, OOP complexity, class semantics, Java applications, Spring Framework.

ЗМІСТ

ВСТУП 10
1 АНАЛІЗ ІСНУЮЧИХ МОДЕЛЕЙ ДЛЯ ОЦІНЮВАННЯ СКЛАДНОСТІ ОБ’ЄКТНО-ОРІЄНТОВАНОГО ПРОЄКТУВАННЯ 13
1.1 Аналіз існуючих моделей для оцінювання складності об’єктно-орієнтованого проєктування 13
1.2 Обґрунтування необхідності проведення дослідження за обраним напрямом 15
1.3 Висновки за розділом 1 16
2 МАТЕМАТИЧНА МОДЕЛЬ ДЛЯ ОЦІНЮВАННЯ СКЛАДНОСТІ ОБ’ЄКТНО-ОРІЄНТОВАНОГО ПРОЄКТУВАННЯ SPRING FRAMEWORK СИСТЕМ НА ОСНОВІ СЕМАНТИКИ КЛАСІВ 17
2.1 Методика проведення первинної обробки даних та побудови еліпсоїда передбачення 17
2.2 Передобробка набору даних для побудови математичної моделі оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів 19
2.3 Побудова рівняння еліпсоїда передбачення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів 21
2.4 Висновки за розділом 2 26
3 РОЗРОБКА ПРОЄКТУ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ СКЛАДНОСТІ ОБ’ЄКТНО-ОРІЄНТОВАНОГО ПРОЄКТУВАННЯ SPRING FRAMEWORK СИСТЕМ НА ОСНОВІ СЕМАНТИКИ КЛАСІВ 27
3.1 Постановка задачі на розробку проєкту програмного забезпечення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів 27

3.2 Розробка ескізного проєкту програмного забезпечення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів 29
3.3 Розробка технічного проєкту програмного забезпечення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів 36
3.4 Розробка робочого проєкту програмного забезпечення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів 40
3.5 Висновки за розділом 3 50
4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВПРОВАДЖЕННЯ ПРОГРАМИ 51
4.1 Розрахунок витрат на створення та впровадження програмного забезпечення 51
4.2 Розрахунок економічної ефективності розробки 54
5 ОХОРОНА ПРАЦІ 56
5.1 Структура управління питаннями охорони праці на підприємстві 56
5.2 Забезпечення техніки безпеки та охорона праці для конкретних посад або при виконанні конкретних завдань 57
5.3 Аналіз шкідливих і небезпечних факторів при роботі з комп’ютером 58
5.4 Розробка заходів по зменшенню дії шкідливих факторів при роботі за ПК . 60
6 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА 63
6.1 Аналіз існуючої проблеми необхідності охорони навколишнього середовища 63
6.2 Розробка заходів щодо зменшення забруднення 65
ВИСНОВКИ 67
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 69
ДОДАТОК А – Технічне завдання 73
ДОДАТОК Б – Опис програмного забезпечення 79
ДОДАТОК В – Програма та методика випробувань ПЗ 82
ДОДАТОК Г – Інструкція користувача 85

ДОДАТОК Д – Текст програми..88

ВСТУП

Актуальність теми. На сучасному етапі розвитку програмної інженерії актуальним завданням є оцінювання складності об’єктно-орієнтованого проєктування (ООП) [1] на основі зв’язків між класами, на основі ідентифікації класів та на основі семантики класів, оскільки від цього безпосередньо залежить якість, уразливість, масштабованість та витрати на підтримку програмних систем протягом усього життєвого циклу. Традиційно складність ООП оцінюється через окремі метрики, проте такий підхід не враховує їх взаємозалежності, що може призводити до хибних результатів.

В об'єктно-орієнтованому проєктуванні оцінка якості програмних систем стала надзвичайно важливою [2]. Набір метрик Чідамбера та Кемерера (CK Metrics) – це широко поширений набір метрик, призначений для вимірювання якості дизайну та коду в об'єктно-орієнтованих системах. Для аналізу складності на основі семантики класів можна використовувати такі метрики [3]: відповіді класу (RFC, the Response for Class), зважені методи класу (WMC, the Weighted Methods per Class) та ступінь зв’язності між методами класу (LCOM, the Lack of Cohesion in Methods). Аналіз зазначених метрик окремо дозволяє виявляти проблеми на рівні застосунку, не враховуючи кореляції між ними. Однак, використання більш комплексного підходу, який включає в себе врахування взаємозв’язку між метриками, дає змогу якісніше оцінити складність системи в цілому. Наприклад, перевищення допустимих значень RFC, WMC чи LCOM відповідно до [4, 5] може сигналізувати про надмірну складність класу, але лише їх комплексний аналіз на рівні всієї системи дозволяє своєчасно виявляти архітектурні дефекти та визначати потребу у рефакторингу.

Для реалізації комплексного підходу доцільним є застосування математичної моделі, яка базується на побудові еліпсоїда передбачення, що враховує взаємозв’язки між метриками RFC, WMC та LCOM. Такий підхід забезпечує можливість багатовимірного аналізу складності об’єктно-

орієнтованих застосунків, зокрема тих, що створюються із використанням Spring Framework, де високий рівень взаємодії між класами та компонентами ускладнює традиційні методи оцінювання.

Мета і завдання дослідження. Метою є підвищення достовірності оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів. Завдання дослідження: проаналізувати існуючі математичні моделі для оцінювання складності об'єктно-орієнтованого проєктування; побудувати математичну модель для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів з використанням нормалізуючого перетворення; розробити програмний інструмент для реалізації побудованої моделі.

Об’єктом дослідження є процес оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів.

Предметом дослідження є математична модель для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів.

Методи дослідження. У вирішенні поставлених завдань використано методи теорії ймовірності, математичної статистики, багатовимірного статистичного аналізу.

Наукова новизна одержаних результатів. Побудовано рівняння еліпсоїда передбачення для метрик RFC, WMC та LCOM на рівні застосунку, нормалізованих тривимірним перетворенням Бокса-Кокса із урахуванням викидів у даних, для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів. Побудоване рівняння, у порівнянні з існуючими математичними моделями, враховує кореляцію між метриками RFC, WMC та LCOM, що дозволяє підвищити достовірність оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів.

Практичне значення одержаних результатів. Розроблено програму для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів, використовуючи фреймворк PySide6.

Особистий внесок здобувача. Робота є самостійно виконаним дослідженням. У роботі [6], що опублікована у співавторстві, здобувачу належить рівняння еліпсоїду передбачення для метрик RFC, WMC та LCOM, нормалізованих тривимірним перетворенням Бокса-Кокса, для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів.

Апробація результатів дослідження. Результати досліджень пройшли апробацію на VI Міжнародній науково-практичній інтернет конференції «Інформаційні технології: моделі, алгоритми, системи – 2025» (Миколаїв, 16-17 листопада 2025 р.).

Публікації. За результатами кваліфікаційної роботи зроблено одну публікацію, а саме матеріали Міжнародної науково-практичної інтернет конференції.

1 АНАЛІЗ ІСНУЮЧИХ МОДЕЛЕЙ ДЛЯ ОЦІНЮВАННЯ СКЛАДНОСТІ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОЄКТУВАННЯ

У поточному розділі було проведено аналіз існуючих моделей для оцінювання складності об'єктно-орієнтованого проєктування. Аргументовано необхідність проведенння дослідження за обраним напрямом та обґрунтовано необхідність побудови нової моделі.

1.1 Аналіз існуючих моделей для оцінювання складності об'єктно-орієнтованого проєктування

На сучасному етапі розвитку програмної інженерії актуальним завданням є оцінювання складності об'єктно-орієнтованого проєктування (ООП) [1] на основі зв'язків між класами, на основі ідентифікації класів та на основі семантики класів, оскільки від цього безпосередньо залежить якість, уразливість, масштабованість та витрати на підтримку програмних систем протягом усього життєвого циклу. Традиційно складність ООП оцінюється через окремі метрики, проте такий підхід не враховує їх взаємозалежності, що може призводити до хибних результатів.

В об'єктно-орієнтованому проєктуванні оцінка якості програмних систем стала надзвичайно важливою [2]. Набір метрик Чідамбера та Кемерера (CK Metrics) – це широко поширений набір метрик, призначений для вимірювання якості дизайну та коду в об'єктно-орієнтованих системах. Для аналізу складності на основі семантики класів використовують наступні метрики [3]: відповіді класу (RFC, the Response for Class), зважені методи класу (WMC, the Weighted Methods per Class) та ступінь зв'язності між методами класу (LCOM, the Lack of Cohesion in Methods). Аналіз зазначених метрик окремо дозволяє виявляти проблеми на рівні застосунку, не враховуючи кореляції між ними. Однак, використання більш комплексного підходу, який включає в себе врахування взаємозв'язку між метриками, дає змогу якісніше оцінити

складність системи в цілому. Наприклад, перевищення допустимих значень RFC, WMC чи LCOM відповідно до [4, 5] може сигналізувати про надмірну складність класу, але лише їх комплексний аналіз на рівні всієї системи дозволяє своєчасно виявляти архітектурні дефекти та визначати потребу у рефакторингу.

Особливої актуальності така оцінка набуває для систем, розроблених із використанням Spring Framework, де об’єктно-орієнтована структура та інтенсивна взаємодія між компонентами створюють додаткові виклики для оцінювання складності [7]. Тому побудова математичної моделі для оцінювання складності об’єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів є доцільним напрямом дослідження, що сприятиме підвищенню достовірності та практичної цінності результатів аналізу.

Для реалізації комплексного підходу доцільним є застосування математичної моделі, яка базується на побудові еліпсоїда передбачення [5], що враховує взаємозв’язки між метриками RFC, WMC та LCOM. Такий підхід забезпечує можливість багатовимірного аналізу складності об’єктно-орієнтованих застосунків, зокрема тих, що створюються із використанням Spring Framework, де високий рівень взаємодії між класами та компонентами ускладнює традиційні методи оцінювання.

У роботі [8] ріняння еліпса передбачення будується на основі двовимірного нормалізуючого перетворення Бокса-Кокса. Для визначення об’єктно-орієнтованої складності додатків, розроблених на Java, автори застосовують вищезазначене рівняння на третьому кроці. Автори запропонували використовувати квадрат відстані Махаланобіса з рівняння еліпса передбачення для нормалізованих метрик RFC та CBO як показник об’єктно-орієнтованої складності додатків з відкритим кодом на Java з точки зору зв'язків між класами.

Математична модель, побудована у [9], базується на аналізі метрик WMC, DIT та NOC для Web застосунків, написаних мовою PHP. В цій роботі описується визначення складності з допомогою ідентифікації класів на рівні

застосунку. Для нормалізації даних використовується тривимірне перетворення Бокса-Кокса, що в свою чергу підвищує точність проведення подальших розрахунків.

На наш погляд, для побудови моделі на основі еліпсоїду передбачення, можна використати алгоритм, описаний у роботах [8, 10]. Спочатку за допомогою взаємо-зворотного нормалізуючого перетворення тривимірні негаусівські дані нормалізують таким чином, щоб розподіл нормалізованих даних був наближений до гаусівського. Далі проводиться ітераційний пошук та видалення даних, які визначено як тривимірні викиди. Далі для нормалізованих даних будують еліпсоїд передбачення. Вихід точки нормалізованих даних за межі еліпсоїда передбачення буде вказувати на наявність викидів з певною ймовірністю, для якої еліпсоїд був побудований [10]. Рекомендується використовувати довірчу ймовірність 0,995, яку зазвичай вибирають при визначенні викидів у багатовимірних даних. Крім того, вихід точки даних за межі еліпсоїда буде свідчити про неможливість проведення оцінювання складності об’єктно-орієнтованого проєктування даного застосунку. Якщо ж точка знаходиться між еліпсоїдом, побудованим для довірчої ймовірності 0,995 та еліпсоїдом, побудованим для довірчої ймовірності 0,95, то складність відповідного проєкту є висока. Розташування точки всередині обох еліпсоїдів вказує на низьку складність об’єктно-орієнтованого проєктування.

1.2 Обґрунтування необхідності проведення дослідження за обраним напрямом

Як було зазначено раніше, для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів може бути застосований еліпсоїд передбачення для нормалізованих даних метрик RFC, WMC та LCOM. Необхідність використання такого математичного методу обумовлена необхідністю багатовимірного аналізу складності об’єктно-орієнтованих застосунків.

Проаналізувавши існуючі моделі для оцінювання складності об’єктно-орієнтованого проєктування [8, 9, 10], можна дійти висновку, що дослідження

на тему оцінювання складності на основі семантики класів проведено не було. Тому для розширення існуючих методів аналізу складності ООП запропоновано створення моделі, що базується на аналізі семантики класів Spring Framework систем, що сприятиме підвищенню точності, надійності та практичної цінності результатів оцінювання. Модель буде побудовано на даних метрик RFC, WMC та LCOM на рівні застосунку, використовуючи тривимірне нормалізуюче перетворення Бокса-Кокса.

1.3 Висновки за розділом 1

У даному розділі було детально розглянуто існуючі математичні моделі для оцінювання складності об’єктно-орієнтованого проєктування, а також наведено обґрунтування актуальності та необхідності проведення дослідження за обраною тематикою. Особливу увагу приділено аналізу метрик об’єктно-орієнтованого проєктування та їхньому впливу на якість програмних систем.

Для підвищення достовірності та точності оцінювання складності об’єктно-орієнтованого проєктування на основі семантики класів доцільно використовувати еліпсоїд передбачення, сформований у просторі нормалізованих значень метрик RFC, WMC та LCOM. Такий підхід дозволяє враховувати багатовимірні залежності між метриками, зменшити вплив випадкових коливань у даних та забезпечити більш надійне та статистично обґрунтоване визначення складності проєкту. Використання еліпсоїда передбачення дає можливість сформувати формальний критерій для прийняття рішень щодо складності програмних систем та підвищити точність моделі загалом.

Таким чином, проведений огляд та аналіз підкреслюють необхідність розроблення удосконаленої математичної моделі, яка дозволить отримувати більш об’єктивні результати при оцінюванні складності об’єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів та сприятиме підвищенню ефективності процесів розробки програмного забезпечення.

2 МАТЕМАТИЧНА МОДЕЛЬ ДЛЯ ОЦІНЮВАННЯ СКЛАДНОСТІ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОЄКТУВАННЯ SPRING FRAMEWORK СИСТЕМ НА ОСНОВІ СЕМАНТИКИ КЛАСІВ

У даному розділі було виконано опис методики проведення первинної обробки даних та побудови еліпсоїда передбачення. Було виконано збір емпіричних даних метрик RFC, WMC та LCOM, проведено тест Мардіа на нормальність даних. Виконано нормалізацію даних тривимірним нормалізуючим перетворенням Бокса-Кокса. На основі отриманих нормалізованих даних метрик було побудовано рівняння еліспоїда передбачення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework застосунків на основі семантики класів.

2.1 Методика проведення первинної обробки даних та побудови еліпсоїда передбачення

На першому етапі необхідно провести первинну обробку даних. Перед цим потрібно відкинути, або прийняти гіпотезу про нормальний розподіл даних [11]. Для перевірки багатовимірних даних пропонується використати тест Мардіа, що базується на перевірці багатовимірної асиметрії та ексцесу [12]:

$$\beta_{1,k} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left[(X_i - \bar{X})^T S_N^{-1}\left(X_j - \bar{X}\right)\right]^3, \quad (2.1)$$

$$\beta_{2,k} = \frac{1}{N}\sum_{i=1}^{N}[(X_i - \bar{X})^T S_N^{-1}(X_i - \bar{X})]^2, \quad (2.2)$$

де X – k × 1 вектор випадкових величин, $X = (X_1, X_2, \dots, X_k)^T$, а S_N – зміщена коваріаційна матриця X, визначена як [12]:

$$S_N = \frac{1}{N}\sum_{i=1}^{N}(X_i - \bar{X})(X_i - \bar{X})^T, \quad (2.3)$$

де $\bar{X}$ – вектор вибіркових середніх, $\bar{X} = (\bar{X}_1, \bar{X}_2, \dots, \bar{X}_k)^T$, $\bar{X}_r = \frac{1}{N}\sum_{i=1}^{N} X_{r_i}$, де r = 1, 2, … , k; N – кількість точок даних.

Ми можемо записати матрицю (2.3) у вигляді:

$$S_N = \begin{pmatrix} S_{X_1X_1} & S_{X_1X_2} & S_{X_1X_3} \\ S_{X_2X_1} & S_{X_2X_2} & S_{X_2X_3} \\ S_{X_3X_1} & S_{X_3X_2} & S_{X_3X_3} \end{pmatrix},$$

де $S_{X_qX_r} = \frac{1}{N}\sum_{i=1}^{N}[X_{qi} - \bar{X}_q][X_{ri} - \bar{X}_i]$, q, r = 1, 2, …, k.

Для $\beta_{1,k}$ використовується тестова статистика

$$\frac{N}{6}\beta_{1,k}, \quad (2.4)$$

яка має наближений розподіл χ^2 з $k(k+1)(k+2)/6$ ступенями свободи та α – рівнем значущості.

Для $\beta_{2,k}$ використовується сам $\beta_{2,k}$, який має наближений нормальний закон розподілу з математичним сподіванням $k(k+2)$ та дисперсією $\frac{8k(k+2)}{N}$.

Щоб перевірити відхилення розподілу від нормального, потрібно порівняти значення статистики (2.4) з квантилем $\chi^2_{k(k+1)(k+2)/6}$, а значення $\beta_{2,k}$ з 1 – α квантилем $z_{2,k}$ нормального закону розподілу з математичним сподіванням $k(k+2)$ та дисперсією $\frac{8k(k+2)}{N}$.

Для проведення додаткової перевірки на нормальність розподілу даних можна використати квадрат відстані Махаланобіса, який дозволяє визначити відхилення спостережень від центру розподілу з урахуванням коваріаційної структури даних [13]. Цей метод особливо ефективний для багатовимірних даних, оскільки враховує не тільки розкид значень, але й взаємозв'язки між змінними:

$$d_i^2 = (Z - \bar{Z})^T S_N^{-1} (Z - \bar{Z}), \tag{2.5}$$

де $\bar{Z}$ – вектор вибіркових середніх, S_N – обернена коваріаційна матриця вибірки [13]:

$$S_N = \frac{1}{N} \sum_{i=1}^{N} (Z_i - \bar{Z})(Z_i - \bar{Z})^T, \tag{2.6}$$

де Z_i – гаусівський випадковий вектор, $Z = (Z_1, Z_2, \dots, Z_m)^T$ для кожної багатовимірної точки даних i, i = 1, 2, … , N; m – кількість компонентів вектору N.

Вектор Z – результат перетворення негаусівкого вектору $X = (X_1, X_2, \dots, X_m)^T$ за допомогою тривимірного нормалізуючого перетворення Бокса-Кокса [11]:

$$Z_j = \begin{cases} (X_j^{\lambda_j} - 1)/\lambda_j, & if \quad \lambda_j \neq 0; \\ \ln(X_j), & if \quad \lambda_j = 0. \end{cases} \tag{2.7}$$

Оцінки параметрів перетворення Бокса-Кокса знаходяться методом максимальної правдоподібності.

2.2 Передобробка набору даних для побудови математичної моделі оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів

Для проведення дослідження на ресурсі GitHub (https://github.com) знайдено 79 застосунків, що створені за допомогою Spring Framework. На першому етапі було проведено збір значень з метрик RFC, WMC та LCOM на рівні застосунку, застосовуючи програмне забезпечення для розрахунку числових характеристик класу CK Metrics [14]. У таблиці 2.1 наведено описову статистику цих даних.

Таблиця 2.1 – Описова статистика набору даних

Метрика	Min	Max	Mean	RMSD
Кількість класів	4	760	85,152	72,898
RFC	0,666	14,864	6,654	2,375
WMC	1,333	17,222	6,047	2,575
LCOM	0,083	420,481	22,268	23,02

Як зазначено в [15], для проведення подальших розрахунків, зібрані дані метрик проєктів нормуються:

$$\overline{X_j} = \frac{\sum_{j=1}^{CS} X_{j_i}}{CS},$$

де $\overline{X_j}$ – вектор метрик , X_{j_i} – значення відповідної метрики на рівні класу, CS – кількість класів проєкту, j = 1, 2, 3 (RFC, WMC, LCOM). Такий підхід із нормуванням значень метрик дозволяє привести їх до одного масштабу, що у свою чергу забезпечує коректну побудову коваріаційної матриці, а проведення оцінювання відстані Махаланобіса набуває змісту [15].

Після збору метрик було проведено тест Мардіа, що базується на перевірці багатовимірної асиметрії (2.1) та ексцесу (2.2) для перевірки гіпотези про нормальність розподілу даних. Згідно з цим тестом, набір наших даних не є гаусівським, оскільки статистика тесту для багатовимірної асиметрії цих даних більша за значення квантилю розподілу хі-квадрат, який становить 25,188 для 10 ступенів свободи та рівня значущості 0,005; статистика тесту для багатовимірного ексцесу також виявилась більшою за значення квантилю $z_{2,k}$, який становить 18,174. Тому для нормалізації даних запропоновано використати тривимірне нормалізуюче перетворення Бокса-Кокса (2.7).

Використовуючи тривимірне нормалізуюче перетворення Бокса-Кокса (2.7) було знайдено значення оцінок параметрів перетворення методом максимальної правдоподібності [12]. Значення оцінок параметрів становлять: $\hat{\lambda}_{RFC} = 0{,}51940127, \hat{\lambda}_{WMC} = 0{,}15501909, \hat{\lambda}_{LCOM} = 0{,}04530663.$

У нашому випадку у 79 точках даних не було виявлено жодного викиду. Для пошуку викидів було використано метод порівняння значення квадрату відстані Махаланобіса (2.5) із значенням статистики [16]

$$\frac{3(N^2-1)}{N(N-3)}F_{3,N-3,\alpha}, \tag{2.8}$$

що становить 14,434 для рівня значущості 0,005.

Для перевірки, виконаємо тест Мардіа для нормалізованих даних. За результатом перевірки можна зробити висновок, що після нормалізації тривимірних даних метрик RFC, WMC, LCOM їх розподіл не відхиляється від нормального: $\beta_{1,k} = 19{,}576$, що є меншим за значення квантилю розподілу хі-квадрат, $\beta_{2,k} = 16{,}931$, що також є меншим за значення квантилю $z_{2,k}$.

Отже, в подальшому для побудови еліпсоїда передбачення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів можна використовувати отриманий набір нормалізованих даних.

2.3 Побудова рівняння еліпсоїда передбачення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів

Для побудови рівняння для еліпсоїда передбачення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів було проведено нормалізацію даних за допомогою тривимірного перетворення Бокса-Кокса (2.7). Рівняння еліпсоїду передбачення [17] для нормалізованих даних метрик RFC, WMC та LCOM визначається як:

$$(Z-\bar{Z})^T S_Z^{-1}(Z-\bar{Z}) \leq \frac{3(N^2-1)}{N(N-3)}F_{3,N-3,\alpha}, \tag{2.9}$$

де Z – гаусівський вектор значень, Z = $\{Z_1, Z_2, Z_3\}^T$; $\bar{Z}$ – вектор середніх значень вибірки. $\bar{Z} = \{\bar{Z}_1, \bar{Z}_2, \bar{Z}_3\}^T$; N – кількість точок даних; $F_{3,N-3,\alpha}$ – квантиль F-розподілу з 3 та N-3 ступенями вільності, α – рівень значимості; S_z – коваріаційна матриця вибірки.

Коваріаційна матриця S_Z згідно з (2.6) має вигляд:

$$S_Z = \begin{pmatrix} 1{,}483884 & 0{,}514635 & 0{,}933355 \\ 0{,}514635 & 0{,}550495 & 1{,}098419 \\ 0{,}933355 & 1{,}098419 & 2{,}78562 \end{pmatrix}.$$

Обернена коваріаційна матриця у нашому випадку така:

$$S_Z^{-1} = \begin{pmatrix} 1{,}0121 & -1{,}2641 & 0{,}1594 \\ -1{,}2641 & 10{,}0991 & -3{,}559 \\ 0{,}1594 & -3{,}559 & 1{,}709 \end{pmatrix}.$$

Вектор вибіркових середніх $\bar{Z}$ має вигляд:

$$\bar{Z} = \{3{,}091; 1{,}904; 2{,}152\}$$

Рівень значущості α = 0,005.

За рівнянням (2.9) визначається межа еліпсоїду передбачення, вихід за яку буде сигналізувати про те, що точка являється викидом. Для визначення складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів нам потрібно використовувати в (2.9) статистику (2.8) для рівня значущості 0,05. Для даного рівня значущості статистика (2.8) має значення 8,496.

Для проведення оцінювання складності об’єктно-орієнтованого проєктування нам потрібно порівнювати значення квадрату відстані Махаланобіса для нормалізованих даних програмного застосунку із значенням 14,434. Якщо це значення більше за значення статистики – точка даних являється тривимірним викидом і в даному випадку ми не можемо визначити складність об’єктно-орієнтованого проєктування за (2.9). Якщо ж значення квадрату відстані Махаланобіса більше за 8,496 та менше за 14,434 це означає,

що програмний застосунок має високу складність за рахунок семантики класів. У випадку, коли значення квадрату відстані Махаланобіса менше за 8,496 означає, що складність об’єктно-орієнтованого проєктування системи на основі семантики класів не є високою [8].

У таблиці 2.2 наведено значення нормалізованих метрик RFC, WMC та LCOM, значення квадрату відстані Махаланобіса та складність програмного застосунку.

Таблиця 2.2 – Значення нормалізованих метрик RFC, WMC та LCOM, значення квадрату відстані Махаланобіса та складність програмного застосунку

№	RFC_BC	WMC_BC	LCOM_BC	d^2	Складність
1	2	3	4	5	6
1	3,9501	1,7254	1,6938	1,1068	Низька
2	1,0565	1,9681	2,4698	4,3845	Низька
3	2,3180	1,9450	2,4093	0,6776	Низька
4	3,5364	2,3617	2,8357	0,4692	Низька
5	1,9358	1,2096	1,5365	2,0254	Низька
6	3,7250	2,7252	3,6834	1,2671	Низька
7	2,6134	2,3259	4,0143	2,5902	Низька
8	3,9377	3,3032	4,3162	4,5364	Низька
9	2,7735	3,0255	5,0156	4,5709	Низька
10	4,0522	2,8901	3,4916	2,4334	Низька
11	3,2987	2,5619	3,1033	1,2242	Низька
12	2,0873	1,0638	0,2625	1,4236	Низька
13	1,9959	2,1675	3,2630	2,2842	Низька
14	2,1982	2,0896	2,8958	1,3261	Низька
15	4,4382	3,0751	4,5621	2,5699	Низька
16	2,3933	2,2611	3,5190	1,8266	Низька
17	3,7269	3,1278	4,4556	3,0369	Низька
18	2,7102	2,2713	2,8328	0,7930	Низька
19	4,0322	2,6487	3,4421	1,1177	Низька
20	4,0025	2,7895	3,7550	1,4724	Низька
21	2,8522	1,8210	2,2796	0,1712	Низька
22	3,9095	2,3678	3,1426	0,5553	Низька
23	3,6069	2,0412	1,5719	1,3258	Низька
24	3,1578	1,9955	2,3396	0,0155	Низька
25	2,6709	1,1110	1,0492	1,6893	Низька
26	4,2907	2,6100	3,0658	1,5322	Низька
27	4,9434	2,1736	1,9144	3,3541	Низька
28	2,6123	0,9207	0,5347	2,2047	Низька
29	4,8674	3,4226	6,0376	5,6649	Низька
30	4,1021	2,8371	3,9776	1,6010	Низька
31	2,1556	2,1569	3,6698	2,8835	Низька

Кінець таблиці 2.2

1	2	3	4	5	6
32	4,8245	2,0280	4,4112	10,6274	Висока
33	2,8147	0,9846	-0,4018	2,6297	Низька
34	1,1991	1,4305	0,6833	3,2454	Низька
35	1,3478	1,1977	0,6452	2,1447	Низька
36	3,1164	3,5776	6,9491	10,4034	Висока
37	1,1272	0,3454	-0,2220	5,4812	Низька
38	2,1681	2,1685	3,2711	1,8914	Низька
39	1,1735	0,6355	-2,3501	10,5644	Висока
40	2,6262	2,4344	2,8357	1,8005	Низька
41	2,7262	2,3142	3,1915	0,9039	Низька
42	2,4986	1,3421	0,0000	2,4158	Низька
43	5,8965	2,1233	3,4408	8,8725	Висока
44	3,0651	1,8498	1,8768	0,0523	Низька
45	3,0387	1,7945	2,7556	1,1927	Низька
46	3,7181	2,6941	3,5246	1,2245	Низька
47	1,9358	1,6287	1,7452	0,9487	Низька
48	1,0197	2,1370	1,6838	7,5727	Низька
49	-0,3656	0,2942	-1,7210	9,7230	Висока
50	4,0497	2,5590	3,7505	1,0775	Низька
51	1,4244	0,3584	-0,6250	4,5298	Низька
52	2,9575	0,9234	1,4977	5,5908	Низька
53	3,8989	1,0299	0,8858	4,6976	Низька
54	5,8168	2,1779	1,3341	8,4134	Низька
55	4,8981	2,4058	2,2627	3,2425	Низька
56	3,1938	1,5016	0,9054	0,7945	Низька
57	3,6283	2,5582	3,5519	0,7957	Низька
58	4,9306	2,0529	4,1878	9,0728	Висока
59	3,8084	1,3521	0,3482	2,6587	Низька
60	3,2839	1,6853	1,3578	0,4196	Низька
61	1,7379	0,9390	0,8943	2,5649	Низька
62	5,0930	1,9777	0,1217	10,5492	Висока
63	2,7150	1,3241	0,6758	0,7960	Низька
64	4,3477	2,6450	1,6658	7,5602	Низька
65	2,5818	1,4314	1,1512	0,4176	Низька
66	3,9828	1,7837	1,8773	1,0367	Низька
67	4,2232	1,5797	1,5819	2,3205	Низька
68	4,2431	2,2485	3,4836	1,7916	Низька
69	2,4986	1,3421	0,0000	2,4158	Низька
70	2,1057	1,8185	2,3593	0,9792	Низька
71	1,7186	0,6926	0,2699	3,1733	Низька
72	2,3996	1,7399	2,1036	0,4279	Низька
73	3,0338	1,0460	0,9544	2,4728	Низька
74	3,0352	1,9291	2,2205	0,0077	Низька
75	4,8688	2,4537	1,5955	6,1685	Низька
76	3,2322	1,1379	0,5829	1,8014	Низька
77	2,8571	0,9981	0,5047	1,9465	Низька
78	0,4869	0,7676	0,1034	4,7287	Низька
79	3,3168	1,9972	2,9585	0,7209	Низька

Графічним зображенням тривимірного еліпсоїда на двовимірну площину екрана ПК буде переріз цього еліпсоїда, через що буде неможливо визначити, де саме знаходиться точка даних. Тому прийнято рішення подавати результат оцінювання складності проєктів табличним способом.

Проведемо тестування побудованої моделі на 16 Spring Framework проєктах, які були взяті з відкритого джерела – репозиторію GitHub (https://github.com). Результати оцінювання складності ООП проєктів представлено в таблиці 2.3.

Таблиця 2.3 – Результати тестування побудованої моделі

№	RFC_BC	WMC_BC	LCOM_BC	d^2	Складність
1	2,9217	1,7467	1,9913	0,084677	Низька
2	3,0651	1,8498	1,8768	0,052307	Низька
3	3,9377	3,3032	4,3162	4,536443	Низька
4	1,6982	1,3060	2,6467	5,775271	Низька
5	3,7097	1,5096	1,1428	1,282818	Низька
6	2,8416	1,9692	3,1468	1,298228	Низька
7	2,9002	1,5360	2,6239	2,815018	Низька
8	3,4119	1,3562	1,3841	1,514269	Низька
9	3,5155	1,9922	1,5362	1,116584	Низька
10	1,6461	0,2942	-1,5975	5,195070	Низька
11	4,4477	2,5490	2,2736	3,369547	Низька
12	3,7171	2,1056	2,5330	0,264855	Низька
13	4,0152	2,1107	1,9025	1,211704	Низька
14	6,6054	3,6455	5,7473	9,204579	Висока
15	4,0090	4,0864	8,7007	17,368999	Складність неможливо оцінити
16	2,4595	3,8200	8,2549	19,733484	Складність неможливо оцінити

За результатами тестування можна зробити висновок, що 13 обраних програмних проєктів мають низьку складність об’єктно-орієнтованого проєктування на основі семантики класів, проєкт під номером 14 має високу складність, а проєкти під номерами 15 та 16 мають квадрат відстані Махаланобіса більший за граничне значення 14,434, тому це говорить про неможливість проведення оцінювання складності об’єктно-орієнтованого проєктування даних застосунків за побудованою моделею.

2.4 Висновки за розділом 2

У цьому розділі представлено методику первинної обробки даних та побудови еліпсоїда передбачення, для оцінювання складності об’єктно-орієнтованого проєктування програмних систем, розроблених із використанням Spring Framework, на основі семантики класів. Проведено аналіз вихідних даних на наявність викидів, побудовано рівняння еліпсоїда передбачення, яке забезпечує аналітичне відображення простору можливих значень метрик складності.

Перевірку вихідних даних на наявність викидів виконано із застосуванням тривимірного нормалізуючого перетворення Бокса-Кокса у поєднанні з квадратом відстані Махаланобіса (SMD), що дозволило виявити та усунути аномальні спостереження. Використання критерію Мардіа для оцінювання багатовимірної нормальності показало, що оброблений набір даних задовольняє умовам гаусівського розподілу, що підтверджує коректність подальших статистичних процедур.

На основі нормалізованих значень метрик RFC, WMC та LCOM було розраховано та побудовано рівняння еліпсоїда передбачення, яке описує багатовимірну область нормальних значень показників складності, що забезпечує підвищену точність оцінювання складності об’єктно-орієнтованого проєктування систем, реалізованих засобами Spring Framework.

3 РОЗРОБКА ПРОЄКТУ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ СКЛАДНОСТІ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОЄКТУВАННЯ SPRING FRAMEWORK СИСТЕМ НА ОСНОВІ СЕМАНТИКИ КЛАСІВ

На основі побудованої математичної моделі у розділі 2, у даному розділі було проведено постановку задачі на розробку проєкту ПЗ. Виконано ескізний, технічний та робочий проєкти програмного забезпечення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів.

3.1 Постановка задачі на розробку проєкту програмного забезпечення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів

Виходячи з вищевикладеного, сформулюємо постановку задачі на розробку проєкту програмного забезпечення. Отже, потрібно розробити програмний застосунок для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів, що буде задовольняти наступні вимоги:

Вимоги до складу виконуваних функцій:

− Вводити з файлу типу CSV емпіричні дані з метрик RFC, WMC та LCOM програмних проєктів.

− Вводити з інтерфейсу програми довірчу ймовірність, метрики RFC, WMC та LCOM, за якими потрібно оцінити складність ООП системи на основі семантики класів.

− Виконувати нормалізацію емпіричних даних тривимірним перетворенням Бокса-Кокса та ітераційно вилучати знайдені викиди шляхом обчислення квадрата відстані Махаланобіса та порівняння знайденого значення із значенням статистики на основі квантилю Фішера.

− Здійснювати побудову рівняння еліпсоїда передбачення для нормалізованих метрик RFC, WMC та LCOM.

− Обчислювати значення тестової статистики для квадрата відстані Махаланобіса.

− Здійснювати оцінювання складності ООП системи на основі семантики класів із заданою ймовірністю.

− Виводити на екран значення квадрата відстані Махаланобіса для нормалізованих метрик RFC, WMC та LCOM, значення тестової статистики для квадрата відстані Махаланобіса, результати оцінювання складності ООП системи на основі семантики класів із заданою ймовірністю.

− Виводити у текстовий файл значення квадрата відстані Махаланобіса для нормалізованих метрик RFC, WMC та LCOM, значення тестової статистики для квадрата відстані Махаланобіса, результати оцінювання складності ООП системи на основі семантики класів із заданою ймовірністю.

Вимоги до складу та параметрів технічних засобів:

Для роботи з програмою необхідний наступний склад технічних засобів з мінімальними параметрами:

− CPU: Intel® Celeron® − 2.16 МГц;

− RAM: 2 GB;

− HDD/SDD 10GB вільного місця;

− пристрої вводу-виводу: маніпулятор типу «миша», клавіатура, монітор з екраном роздільної здатності не менше ніж 1600x900 пікселів.

Вимоги до інформаційної та програмної сумісності

Програма повинна працювати під управлінням ОС Windows версії не нижче 7. Необхідна наявність інтерпретатора Python версії не нижче 3.8.

Деталізація постановки задачі на розробку програмного забезпечення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів представлена в технічному завданні, що наведено в додатку А.

3.2 Розробка ескізного проєкту програмного забезпечення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів

На даному етапі було виявлено акторів системи, проведено моделювання варіантів використання, побудовано діагарму діяльності варіантів використання програмного забезпечення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів.

В роботі використовується об’єктно-орієнтована методологія та уніфікована мова моделювання UML. В якості CASE-засобу обраний App DrawIo [18].

Розробка діаграми варіантів використання системи

Розробка Use Case – моделі відповідно до технічного завдання, представленого у додатку А, передбачає визначення та опис акторів системи, опис виявлених прецедентів, побудову діаграми варіантів використання (рис. 3.1) та розробку специфікацій варіантів використання [19].

Після проведення обстеження предметної галузі можна виділити 1 групу акторів – розробник.

Опис актора розробник: Користувач системи, якому доступний весь функціонал системи, може виконувати внесення інформації про метрики проєктів, виконувати запити на її обробку (побудова моделі, побудова еліпсоїда передбачення, визначення рівня складності ООП проєкту).

Виявлені варіанти використання системи представлено в таблиці 3.1.

Таблиця 3.1 – Виявлені варіанти використання програмного забезпечення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів

Код	Актори	Назва	Формулювання
1	2	3	4
Р1	Розробник	Ввести дані метрик RFC, WMC, LCOM	Варіант використання дозволяє користувачу вводити дані метрик RFC, WMC LCOM в систему

Кінець таблиці 3.1

1	2	3	4
Р2	Розробник	Ввести довірчу ймовірність	Варіант використання дозволяє користувачу вводити значення довірчої ймовірності в систему
Р3	Розробник	Ввести метрики RFC, WMC та LCOM застосунку	Варіант використання дозволяє користувачу вводити дані метрик RFC, WMC та LCOM застосунку для оцінки в систему
Р4	Розробник	Провести побудову моделі	Варіант використання дозволяє користувачу провести побудову моделі по введеним даним метрик
Р5	Розробник	Побудувати еліпсоїд передбачення	Варіант використання дозволяє користувачу побудувати рівняння еліпсоїда передбачення та сам еліпсоїд передбачення
Р6	Розробник	Визначити рівень складності ООП застосунку	Варіант використання дозволяє користувачу за введеними метриками оцінити рівень складності ООП застосунку
Р7	Розробник	Зберегти результати	Варіант використання дозволяє користувачу зберігти отримані результати оцінювання складності об'єктно-орієнтованого проєктування Spring Framework системи

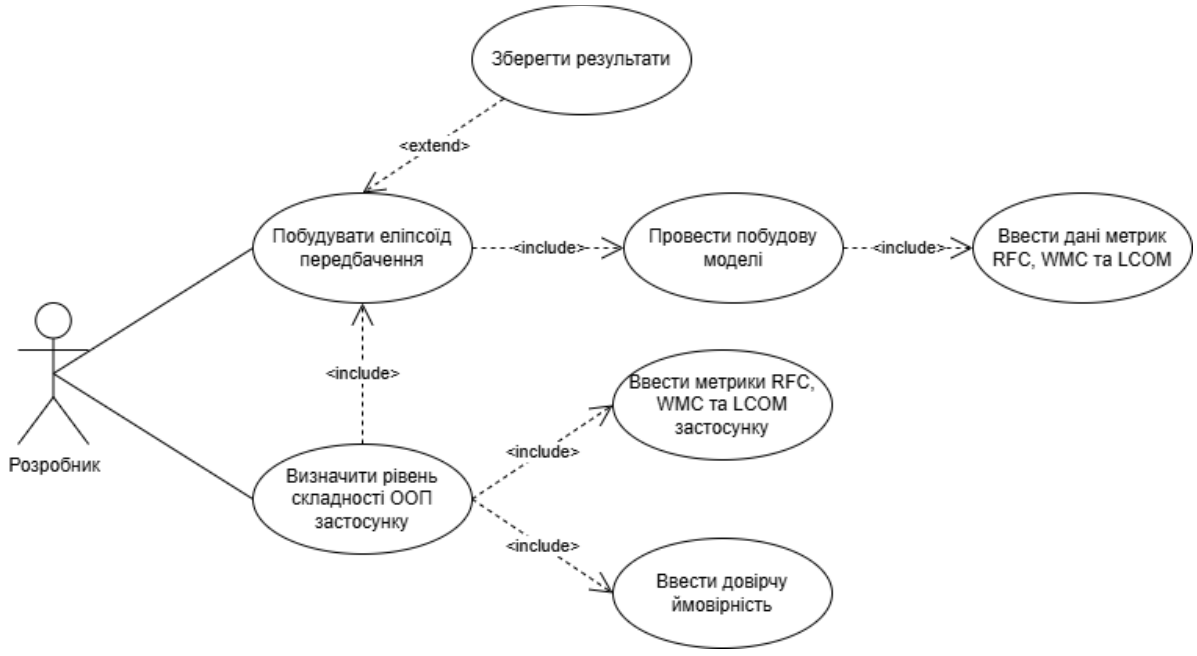


Рисунок 3.1 – Діаграма варіантів використання програмного забезпечення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем

Для систематизації представлення сценаріїв варіантів використання було побудовано діаграми діяльності для основних прецедентів системи, що дозволило детально простежити послідовність дій користувачів та внутрішніх процесів системи. На основі цих діаграм вдалося чіткіше визначити логіку

переходів між станами, виявити можливі альтернативні та виняткові потоки, а також уточнити взаємодію між окремими компонентами. Такий підхід забезпечив більш глибоке розуміння функціональних вимог і дав змогу сформувати цілісне бачення поведінки системи в різних ситуаціях. Відповідно до отриманих результатів було створено проєкт користувацького інтерфейсу, структура та елементи якого узгоджені з логікою сценаріїв. Це дозволило розробити інтерфейс, який не лише відповідає вимогам прецедентів, а й забезпечує інтуїтивність, послідовність і логічність взаємодії для майбутніх користувачів системи.

Розробка діаграми діяльності системи

Діаграма діяльності UML – візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій [20].

Діаграма діяльності відображає динаміку системи та представляє переходи потоків управління в системі від однієї дії до іншої, а також паралельні дії та альтернативні потоки. Основними елементами діаграм діяльності є: «Діяльність», «Перехід», «Елемент вибору», «Початок», «Кінець», «Лінія синхронізації».

На рисунку 3.2 предсталено діаграму діяльності системи.

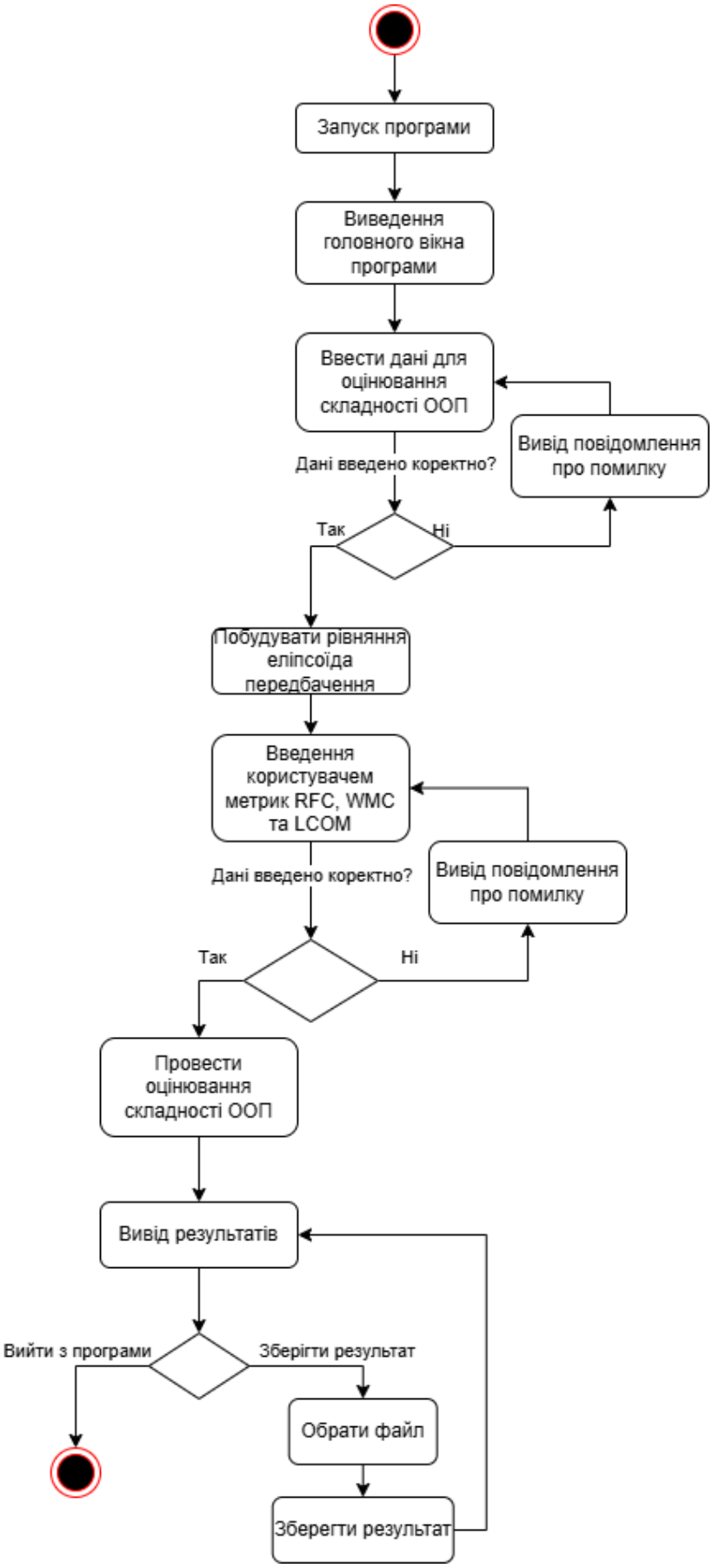


Рисунок 3.2 – Діаграма діяльності програмного забезпечення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем

Розробка зовнішньої специфікації програми

Назва програмного застосунку: ClassComplexityEvaluator.py.

Функції, які виконує програма:

− Вводити з файлу типу CSV емпіричні дані з метрик RFC, WMC та LCOM програмних проєктів.

− Вводити з інтерфейсу програми довірчу ймовірність, метрики RFC, WMC та LCOM, за якими потрібно оцінити складність ООП системи на основі семантики класів.

− Виконувати нормалізацію емпіричних даних тривимірним перетворенням Бокса-Кокса та ітераційно вилучати знайдені викиди шляхом обчислення квадрата відстані Махаланобіса та порівняння знайденого значення із значенням статистики на основі квантилю Фішера;

− Здійснювати побудову рівняння еліпсоїда передбачення для нормалізованих метрик RFC, WMC та LCOM.

− Обчислювати значення тестової статистики для квадрата відстані Махаланобіса.

− Здійснювати оцінювання складності ООП системи на основі семантики класів із заданою ймовірністю.

− Виводити на екран значення квадрата відстані Махаланобіса для нормалізованих метрик RFC, WMC та LCOM, значення тестової статистики для квадрата відстані Махаланобіса, результати оцінювання складності ООП системи на основі семантики класів із заданою ймовірністю.

− Виводити у текстовий файл значення квадрата відстані Махаланобіса для нормалізованих метрик RFC, WMC та LCOM, значення тестової статистики для квадрата відстані Махаланобіса, результати оцінювання складності ООП системи на основі семантики класів із заданою ймовірністю.

Вхідні дані:

− значення емпіричних даних метрик RFC, WMC та LCOM, значення довірчої ймовірності та значення метрик RFC, WMC та LCOM проєкту.

Вихідні дані:

− значення квадрату відстані Махаланобіса для нормалізованих метрик RFC, WMC та LCOM, значення тестової статистики для квадрата відстані Махаланобіса та результати оцінювання складності ООП системи на основі семантики класів із заданою ймовірністю.

Зовнішні ефекти: відображення вікна для введення даних, рендер таблиць та полів для виведення даних, вихід із програми.

Розробка ескізу користувацького інтерфесу програми

Одним із основних етапів проєктування програмного забезпечення є розробка інтерфейсу користувача програмного додатку.

На рисунках 3.4 – 3.6 представлено ескізи основних сцен та форм програмного додатку.

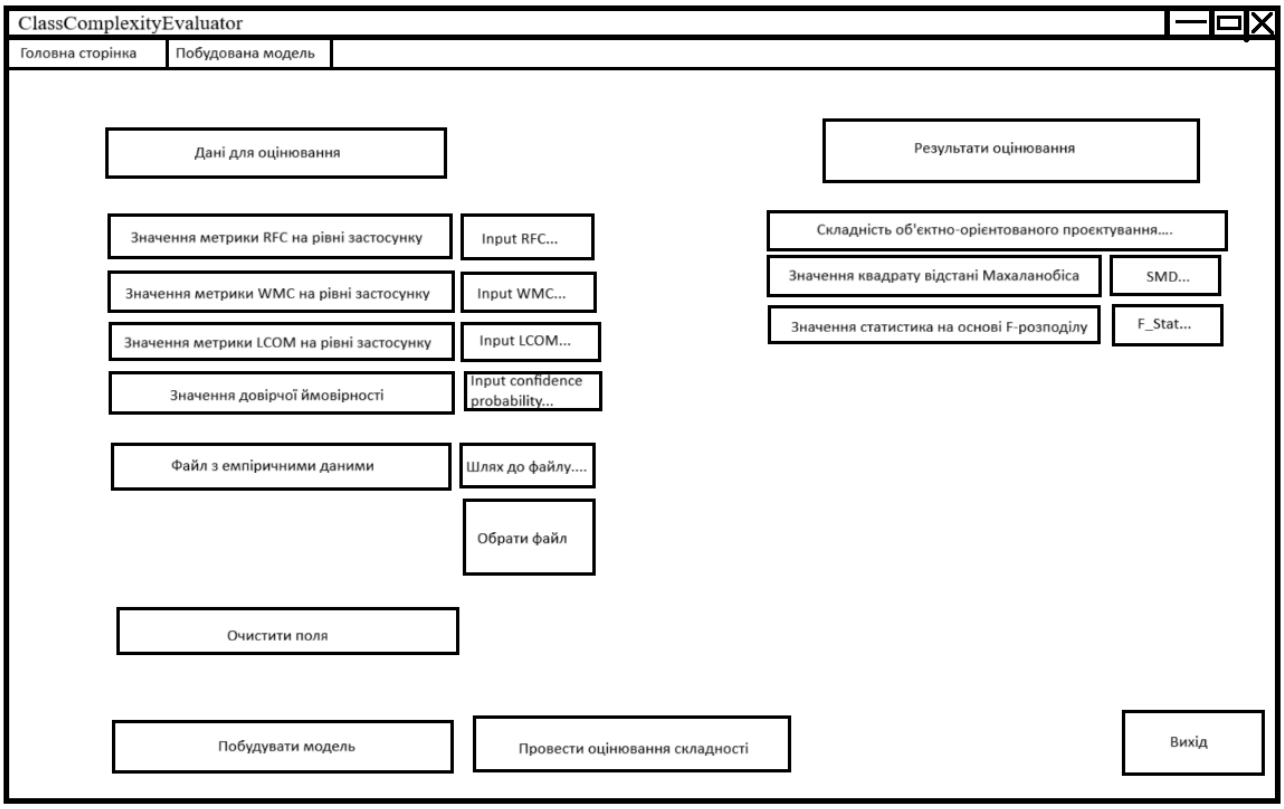


Рисунок 3.4 – Прототип головного вікна програми

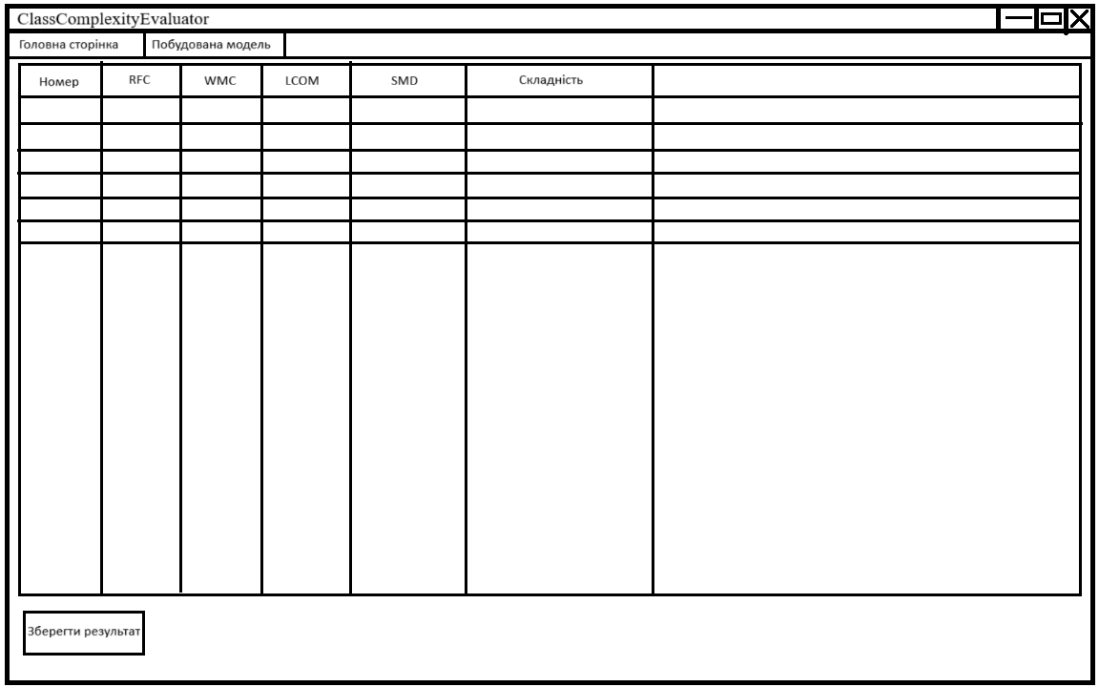


Рисунок 3.5 – Прототип вікна із побудованою моделею

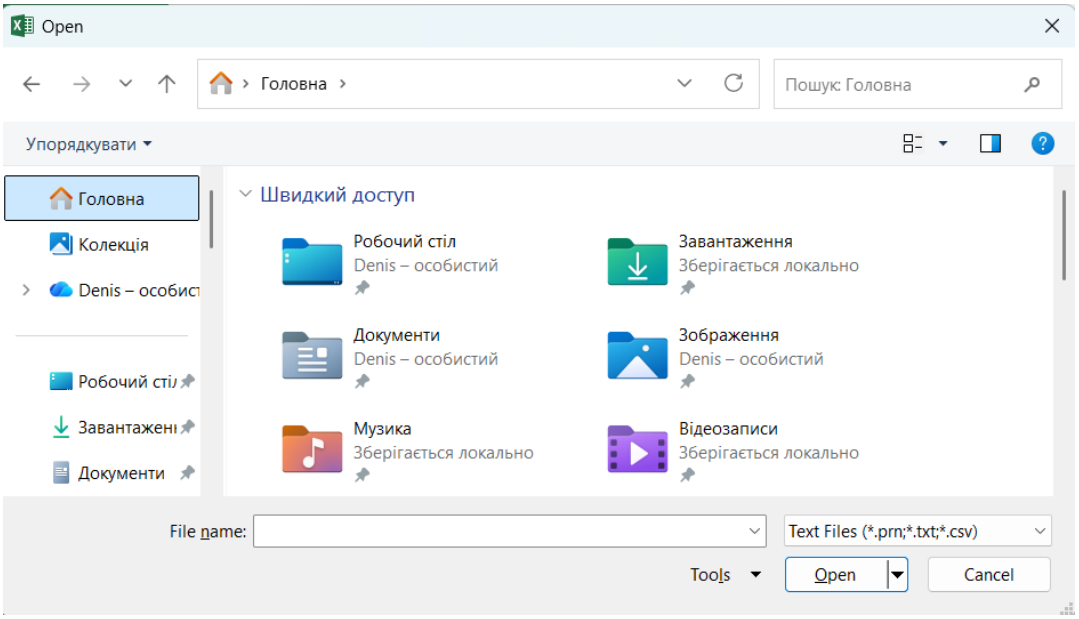


Рисунок 3.6 – Вікно вибору файла з емпіричними даними

У результаті ескізного проєктування було розроблено діаграму варіантів використання, діаграму діяльності, зовнішню специфікацію програмного забезпечення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів та ескіз користувацького інтерфейсу. Далі буде проведено розробку технічного проєкту програмного забезпечення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів.

3.3 Розробка технічного проєкту програмного забезпечення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів

Ґрунтуючись на розробленому ескізному проєкті, проведемо розробку технічного проєкту програмного забезпечення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів. На рисунку 3.7 представлено статичну модель системи у вигляді діаграми класів.

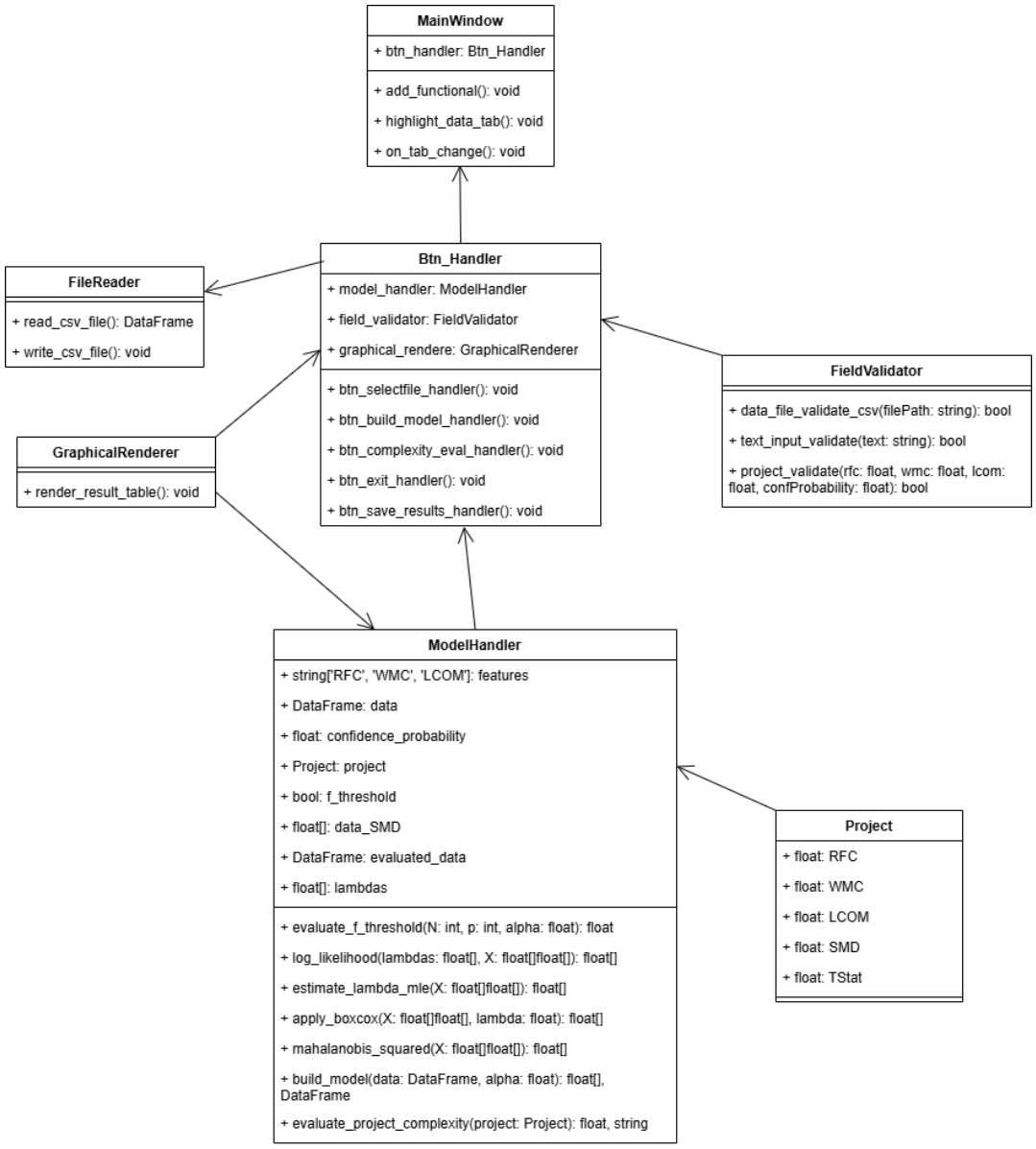


Рисунок 3.7 – Діаграма класів програмного забезпечення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів

Програма складється з наступних класів: MainWindow – головне вікно програми, у якому користувач буде вводити дані, проводити їх обробку та отримувати результат обробки; Btn_Handler – клас, який відповідає за прив’язку кнопок програми до їх відповідного функціоналу; FieldValidator – клас, який відповідає за валідацію полів для вводу та повідомлення користувачу про помилки у вхідних даних; FileReader – клас, який відповідає за зчитування .csv файлу з даними метрик RFC, WMC та LCOM для подальшої побудови моделі та збереження отриманих результатів; ModelHandler – клас, який відповідає за обробку даних, а саме – пошук викидів, нормалізацію даних, побудову моделі для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів, визначення складності досліджуваного програмного застосунку; GraphicalRenderer – клас, який відповідає за створення табличного виду представлення побудованої моделі. Project – клас, який відповідає за досліджуваний програмний проєкт.

Нижче представлено опис кожного класу.

Назва класу: MainWindow.

Функціонал класу:

1) Ініціалізація головного вікна програми;
2) Рендер полів для введення та виведення даних;
3) Завантаження даних метрик за допомогою модуля FileReader;
4) Рендер фенкціональних елементів (кнопок) для проведення розрахунків.

Вхідні дані:

– файл типу .csv для введення даних метрик RFC, WMC, LCOM для побудови моделі;

– дані метрик RFC, WMC, LCOM проєкту, складність якого буде оцінюватись;

– значення довірчої ймовірності.

Вихідні дані:

– побудована модель;

– результат оцінювання складності об'єктно-орієнтованого проєктування проєкту.

Зовнішні ефекти: при натисканні кнопки Обрати файл вивід вікна із можливістю вибору файлу із дамим метрик для побудови моделі.

Назва класу: Btn_Handler.

Функціонал класу: обробка подій натискання кнопок на головному екрані.

Вхідні дані:

– команди користувача (натискання функціолнальних елементів інтерфейсу – кнопок).

Вихідні дані:

– обробка відповідних подій, після натискання кнопок.

Зовнішні ефекти: виведення повідомлень про помилки у введенних даних.

Назва класу: FieldValidator.

Функціонал модуля: валідація данних, введених користувачем.

Вхідні дані:

– дані метрик RFC, WMC, LCOM та значення довірчої ймовірності;

– дані метрик RFC, WMC, LCOM для побудови моделі.

Вихідні дані:

– результат валідації введених даних.

Назва класу: FileReader.

Функціонал класу: зчитування та запис даних у файл

Вхідні дані:

– шлях до файлу для зчитування даних метрик;

– шлях до файлу для запису результатів оцінювання.

Вихідні дані:

– зчитані дайні метрик у масив;

– файл із записаною в нього інформацію про результати оцінювання складності об'єктно-орієнтованого проєктування системи.

Назва класу: ModelHandler.

Функціонал класу:

1) Ітераційний пошук та видалення викидів у даних;
2) Нормалізація даних тривимірним перетворенням Бокса-Кокса;
3) Розрахунок коваріаційної та оберненої коваріаційної матриці;
4) Розрахунок тестової статистики на основі F-розподілу;
5) Розрахунок значень квадратів відстаней Махаланобіса;
6) Оцінювання складності об’єктно-єрієнтованого проєктування системи.

Вхідні дані:

– дані метрик RFC, WMC, LCOM програмного застосунку, що досліджується;

– дані метрик RFC, WMC, LCOM із файлу для побудови моделі;

– значення довірчої ймовірності.

Вихідні дані:

– побудована модель у вигляді DataFrame (номер проєкту, значення квадрату відстані Махаланобіса для кожного проєкту, складність об’єктно-орієнтованого проєктування);

– результат оцінювання складності архітектури проєкту.

Назва класу: GraphicalRenderer.

Функціонал класу: вивід результатів побудованої моделі у табличному вигляді.

Вхідні дані:

– побудована модель у вигляді DataFrame (номер проєкту, значення квадрату відстані Махаланобіса для кожного проєкту, складність об’єктно-орієнтованого проєктування).

Вихідні дані:

– оновлена вкладка «Побудована модель» із табличним зображенням результатів.

Назва класу: Project.

Функціонал класу: збереження числових даних метрик досліджуваного проєкту.

Вхідні дані:

– значення метрик RFC, WMC, LCOM на рівні застосунку, значення тестової статистики.

Вихідні дані:

– значення квадрату відстані Махаланобіса для даного проєкту.

3.4 Розробка робочого проєкту програмного забезпечення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів

Для створення програмного забезпечення оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів мною були обрані такі засоби розробки:

− Мова програмування Python. Python є високорівневою мовою програмування, яка відзначається простою синтаксису, що сприяє зниженню порогу входу в розробку; має широкий спектр бібліотек для роботи з даними, побудови інтерфейсів користувача та розв'язання математичних задач; забезпечує можливість розподілу проєкту на окремі модулі та підтримує інтеграцію зі сторонніми бібліотеками. Завдяки великій спільноті розробників є легкий доступ до документації, прикладів та підтримки у разі виникнення проблем [21].

− Середовище розробки JetBrains PyCharm. Це інтегроване середовище розробки (IDE), спеціально створене для мови Python. PyCharm аналізує код у реальному часі та пропонує варіанти завершення, що дозволяє уникнути багатьох помилок. Вбудований аналізатор допомагає виявити синтаксичні помилки, неефективний код та потенційні проблеми ще до запуску програми.

PyCharm дозволяє покроково виконувати код, встановлювати точки зупинки та аналізувати значення змінних у реальному часі. IDE має вбудовану підтримку таких інструментів, як pip для управління бібліотеками, pytest для тестування, а також інтеграцію з віртуальними середовищами Python [22].

− Фреймворк PySide6. Це набір інструментів для створення сучасних графічних інтерфейсів користувача (GUI), який базується на бібліотеці Qt. Додатки, створені з PySide6, працюють на Windows, Linux і macOS без додаткових змін у коді. PySide6 надає доступ до понад 600 класів і модулів, що дозволяє створювати складні GUI, включаючи кнопки, таблиці, графіки, текстові поля та багато іншого [23].

Кожен з обраних інструментів − Python, PyCharm та PySide6 доповнюють один одного, утворюючи ефективне середовище розробки для програмного забезпечення, що відповідає поставленим завданням. Ці інструменти забезпечують зручність у роботі, високу продуктивність та дозволяють реалізувати всі необхідні функції для аналізу складності ООП.

В середовищі PyCharm було розроблено програмний застосунок з використанням фреймворку PySide6. Реалізовано зручний, простий та зрозумілий користувацький інтерфейс. Додаток являє собою програмне вікно з двома вкладкам: головна сторінка, сторінка з побудованою моделлю.

Для реалізації функціоналу розроблюваного ПЗ мною було використано такі бібліотеки мови Python:

− numpy для проведення математичних операцій над масивами емпіричних даних [24];

− scipy для статистики, оптимізації, інтеграції, лінійної алгебри, перетворень Фур’є, тощо [25];

− pandas це швидкий, потужний, гнучкий та простий у використанні інструмент для аналізу та маніпулювання даними з відкритим кодом [26];

− csv модуль реалізує класи для читання та запису табличних даних у форматі CSV. Він дозволяє програмістам сказати: «записати ці дані у форматі, який використовується Excel» або «зчитати дані з цього файлу, який був

згенерований Excel», не знаючи точних деталей формату CSV, що використовується Excel [27].

Як приклад нижче наведено текст одного із основних програмних модулів – ModelHandler, що відповідає за побудову моделі та визначення складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів.

```
class ModelHandler:

    def __init__(self, data, confidence_probability, project):
        self.features = ['RFC', 'WMC', 'LCOM']
        self.data = data
        self.confidence_probability = confidence_probability
        self.project = project
        self.f_threshold = self.evaluate_f_threshold(len(data),
len(self.features), self.confidence_probability)
        self.data_SMD, self.evaluated_data, self.lambdas =
self.build_model(self.data, self.confidence_probability)

    def evaluate_f_threshold(self, N, p, alpha):
        f_val = f.ppf(1 - alpha, p, N - p)
        threshold = (p * (N**2 - 1)) / (N * (N - p)) * f_val
        return threshold

    def PARAMS_box_cox_transform(self, X, lambdas):
        X = np.array(X)
        X_transformed = np.zeros_like(X)

        for j in range(X.shape[1]):
            if lambdas[j] == 0:
                X_transformed[:, j] = np.log(X[:, j])
            else:
                X_transformed[:, j] = (X[:, j] ** lambdas[j] - 1) / lambdas[j]
        return X_transformed

    def PARAMS_inverse_box_cox_transform(X_transformed, lambdas):
        X_original = np.zeros_like(X_transformed)
        for j in range(X_transformed.shape[1]):
            if lambdas[j] == 0:
                X_original[:, j] = np.exp(X_transformed[:, j])
            else:
                X_original[:, j] = (X_transformed[:, j] * lambdas[j] + 1) ** (1
/ lambdas[j])
        return X_original

    def PARAMS_log_likelihood(self, lambdas, X):
        N, k = X.shape
        X = np.maximum(X, 1e-6)

        transformed_X = self.PARAMS_box_cox_transform(X, lambdas)

        S_n = np.cov(transformed_X, rowvar=False)
```

```
        sign, log_det = np.linalg.slogdet(S_n)

        if sign <= 0:
            return np.inf

        term1 = np.sum((lambdas - 1) * np.sum(np.log(X), axis=0))
        log_likelihood_value = term1 - (N / 2) * log_det

        return -log_likelihood_value

    def PARAMS_estimate_lambda_mle(self, X):
        k = X.shape[1]
        initial_lambdas = np.ones(k)

        result = minimize(
            self.PARAMS_log_likelihood, initial_lambdas, args=(X,),
            method="L-BFGS-B", bounds=[(0.01, 5)] * k
        )

        return result.x

    def apply_boxcox(self, column, lam):
        return boxcox(column, lmbda=lam)

    def mahalanobis_squared(self, X):
        mean_vec = np.mean(X, axis=0)
        cov_matrix = np.cov(X, rowvar=False, ddof=0)
        cov_inv = inv(cov_matrix)
        diff = X - mean_vec
        d_squared = np.sum(diff @ cov_inv * diff, axis=1)
        return d_squared

    def build_model(self, data, alpha=0.005):
        X = data.copy().to_numpy()
        outliers = []
        iteration = 0

        while True:
            train_data = np.column_stack((X[:, 0], X[:, 1], X[:, 2]))

            train_lambdas = self.PARAMS_estimate_lambda_mle(train_data)
            print(train_lambdas)
            RFC_bc = self.apply_boxcox(X[:, 0], train_lambdas[0])
            WMC_bc = self.apply_boxcox(X[:, 1], train_lambdas[1])
            LCOM_bc = self.apply_boxcox(X[:, 2], train_lambdas[2])

            X_transformed = np.column_stack([RFC_bc, WMC_bc, LCOM_bc])

            d2 = self.mahalanobis_squared(X_transformed)

            threshold = self.evaluate_f_threshold(len(X), len(self.features),
alpha)
            print(threshold)
            mask = d2 > threshold

            if not np.any(mask):
                break
```

```
            deviations = d2 - threshold
            idx_to_remove = np.argmax(deviations)

            print(f"Ітерація {iteration+1}: Видалено викид з індексом
{idx_to_remove}, значення d² = {d2[idx_to_remove]:.4f}, поріг F_Stat =
{threshold:.4f}")

            outliers.append(X[idx_to_remove])
            X = np.delete(X, idx_to_remove, axis=0)
            iteration += 1

        print(f"\nЗагальна кількість викидів: {len(outliers)}")
        return d2, pd.DataFrame(X, columns=self.features), train_lambdas

    def evaluate_project_complexity(self):
        project_RFC_bc = self.apply_boxcox(self.project.RFC, self.lambdas[0])
        project_WMC_bc = self.apply_boxcox(self.project.WMC, self.lambdas[1])
        project_LCOM_bc = self.apply_boxcox(self.project.LCOM, self.lambdas[2])
        project_metrics_transformed = np.column_stack([project_RFC_bc,
project_WMC_bc, project_LCOM_bc])

        RFC_bc = self.apply_boxcox(self.evaluated_data['RFC'], self.lambdas[0])
        WMC_bc = self.apply_boxcox(self.evaluated_data['WMC'], self.lambdas[1])
        LCOM_bc = self.apply_boxcox(self.evaluated_data['LCOM'],
self.lambdas[2])
        model_metrics_transformed = np.column_stack([RFC_bc, WMC_bc, LCOM_bc])

        mean_vec = np.mean(model_metrics_transformed, axis=0)
        cov_matrix = np.cov(model_metrics_transformed, rowvar=False, ddof=0)
        cov_inv = inv(cov_matrix)
        diff = project_metrics_transformed - mean_vec
        project_SMD = np.sum(diff @ cov_inv * diff, axis=1)[0]

        self.project.SMD = project_SMD
        self.project.FStat = self.f_threshold

        complexity = ''

        if project_SMD < self.evaluate_f_threshold(len(self.data),
len(self.features), 0.05):
            complexity = "Складність ООП проєкту низька"
        elif project_SMD < self.f_threshold:
            complexity = "Складніть ООП проєкту висока"
        else:
            complexity = "Складність ООП проєкту не можливо оцінити"

        return project_SMD, complexity
```

Скріншоти реалізованого програмного забезпечення представлено на рисунках 3.8 – 3.9.

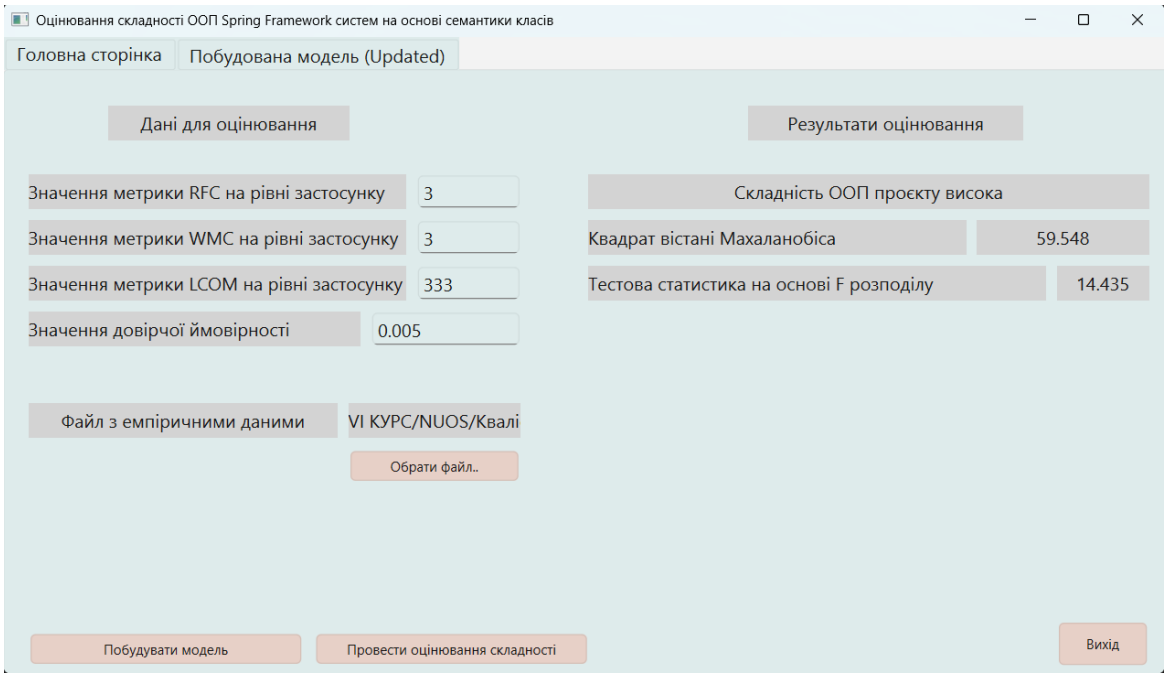


Рисунок 3.8 – Скріншот головного вікна застосунку

	RFC	WMC	LCOM	SMD	Складність ООП
1	8.5682	4.6136	5.1136	1.1068	Низька
2	2.3214	5.5714	10.3929	4.3845	Низька
3	4.5789	5.4737	9.8421	0.6776	Низька
4	7.4444	7.4815	14.4074	0.4692	Низька
5	3.8182	3.0303	4.4167	2.0254	Низька
6	7.9474	9.7105	30.1579	1.2670	Низька
7	5.2125	7.2875	39.9750	2.5902	Низька
8	8.5333	14.4000	51.5333	4.5365	Низька
9	5.5723	11.9518	91.7952	4.5709	Низька
10	8.8571	10.8929	25.5714	2.4334	Низька
11	6.8333	8.6481	18.2407	1.2241	Низька
12	4.1118	2.6770	1.2981	1.4236	Низька
13	3.9333	6.4800	20.9733	2.2842	Низька
14	4.3333	6.1111	15.1944	1.3261	Низька
15	9.9909	12.3617	63.2401	2.5699	Низька
16	4.7368	6.9474	26.1842	1.8267	Низька

Рисунок 3.9 – Скріншот вікна застосунку із побудованою моделлю

Після завершення розробки програмного забезпечення було проведено його тестування. Як приклад нижче наведено результат тестування модулів ModelHandler та GraphicalRenderer, які будують модель та генерують табличку із результатами відповідно. На рисунку 3.10 предсталено фрагмент таблчного представлення результату побудованої моделі.

	RFC	WMC	LCOM	SMD	Складність ООП
33	5.6667	2.5000	0.6667	2.6295	Низька
34	2.5400	3.6400	1.9600	3.2454	Низька
35	2.7778	3.0000	1.8889	2.1446	Низька
36	6.3815	17.2222	420.4815	10.4033	Висока
37	2.4286	1.4000	0.8000	5.4812	Низька
38	4.2727	6.4848	21.1212	1.8914	Низька
39	2.5000	1.8333	0.0833	10.5665	Висока
40	5.2407	7.8889	14.4074	1.8005	Низька
41	5.4648	7.2254	19.7042	0.9039	Низька
42	4.9615	3.3846	1.0000	2.4157	Низька
43	14.8642	6.2685	24.4722	8.8725	Висока
44	6.2571	5.0857	6.0571	0.0523	Низька
45	6.1935	4.8710	13.4194	1.1926	Низька
46	7.9286	9.5000	26.3095	1.2245	Низька
47	3.8182	4.2727	5.3636	0.9487	Низька
48	2.2667	6.3333	5.0667	7.5725	Низька

Рисунок 3.10 – Фрагмент таблчного представлення результату побудованої моделі

Порівнюючи результати, отримані за допомогою створеного програмного забезпечення, та результати розрахунків, представлених у таблиці 2.2, можна зробити висновок про те, що програмне забезпечення повністю та цілком відтворило результати побудованої моделі для об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів із пункту 2 даної роботи. Це свідчить про його працездатність та коректність реалізації математичної моделі, а також підтверджує можливість практичного використання розробленого інструмента для автоматизованого оцінювання складності програмного забезпечення на основі обраних метрик.

Тестування розробленого програмного забезпечення

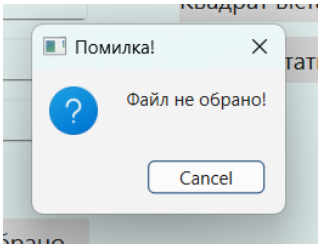
Тестування програмного забезпечення – це процес, під час якого проводиться ретельне дослідження та випробування програмного продукту з метою виявлення розбіжностей між фактичною поведінкою програми та її очікуваними результатами на основі заздалегідь визначеного набору тестових випадків. Цей процес включає в себе перевірку функціональності, продуктивності, безпеки та інших аспектів програми, щоб забезпечити її

відповідність вимогам та гарантувати стабільну роботу у реальних умовах використання.

Тестування програмного застосунку проведено за методикою випробувань, яку наведено у додатку В та представлено нижче.

Тестування програмного забезпечення проводилось методом «чорної скриньки». На рисунках 3.11 – 3.16 представлено результати тестування програмного забезпечення.

Якщо користувачем не обрано файл, програма виводить повідомлення про помилку.



Риснуок 3.11 – Повідомлення, що файл не обрано

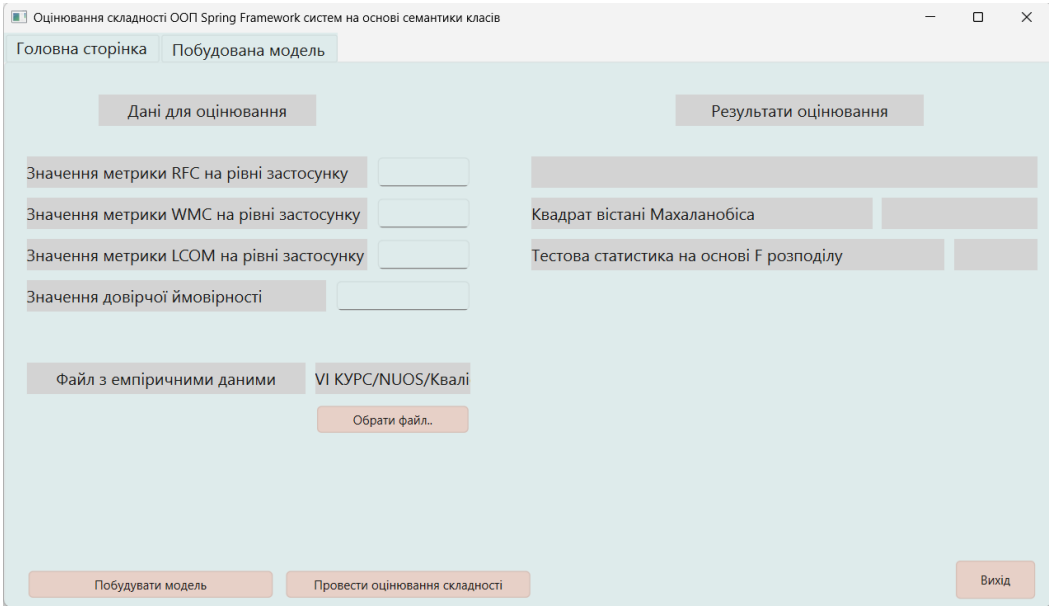


Рисунок 3.12 – Вікно програми із заповненим полем для вибору файла з даними

Якщо дані для оцінювання складності ООП проєкту не було введено, або вони не валідні відображається повідомлення про помилку.

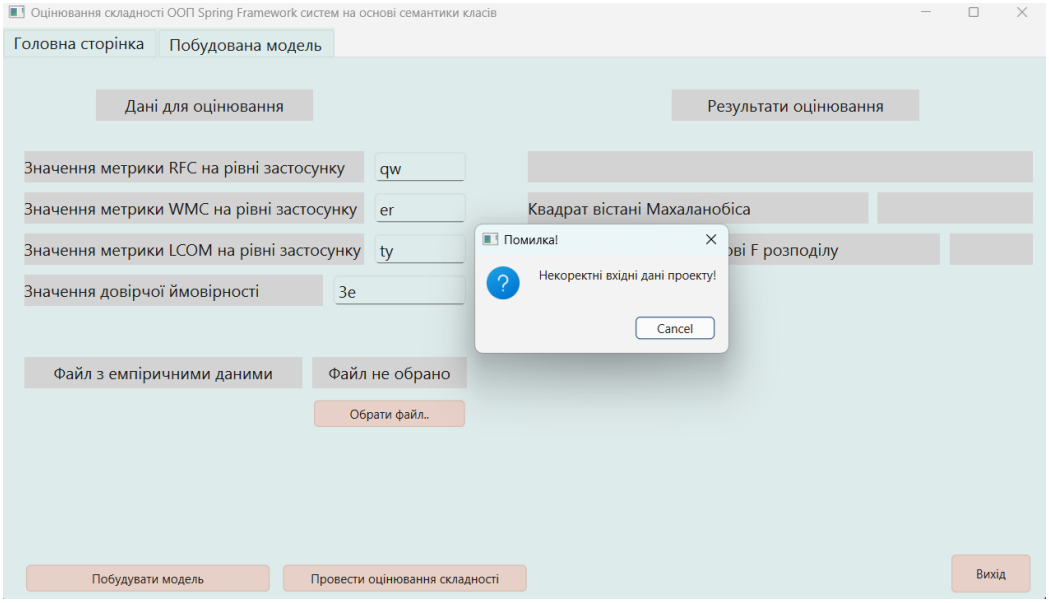


Рисунок 3.13 – Повідомлення про те, що введно некоректні дані для оцінювання складності ООП проєкту

Якщо дані не були завантажені у застосунок (не був обраний файл), побудова моделі не може бути здійснена. Через це з’являється відповідне повідомлення про помилку.

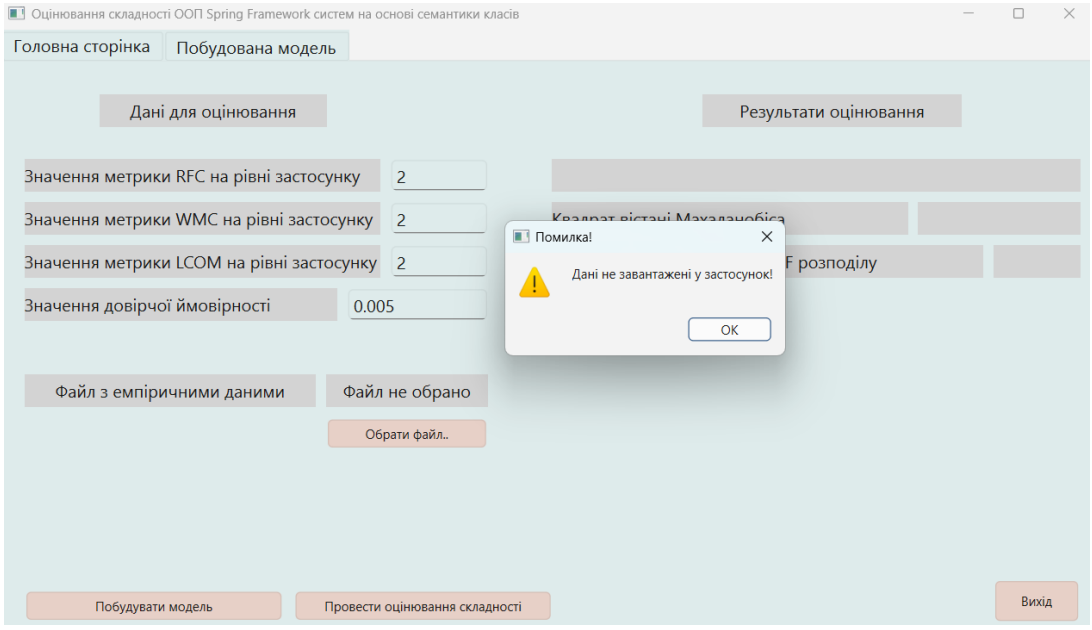


Рисунок 3.14 – Повідомлення про те, що дані для побудови моделі не завантажено у застосунок

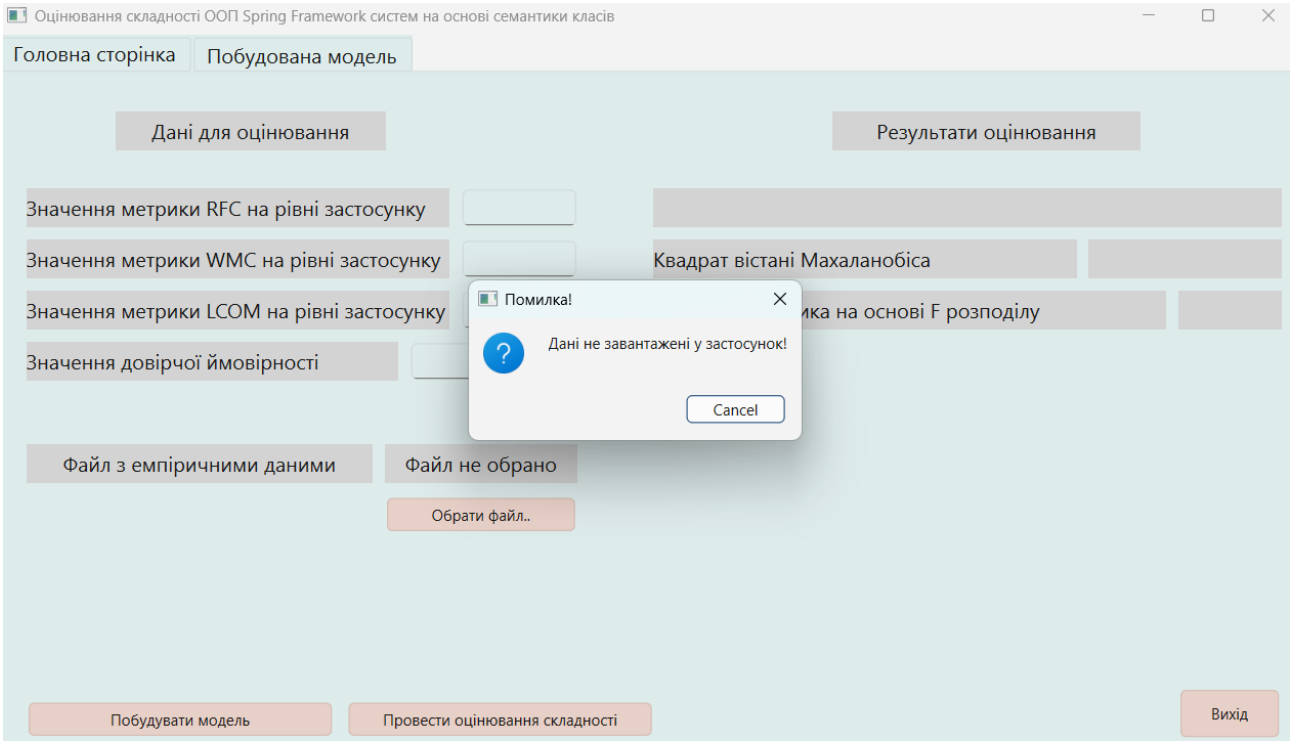


Рисунок 3.15 – Повідомлення про помилку при спробі оцінити складність ООП без побудованої моделі

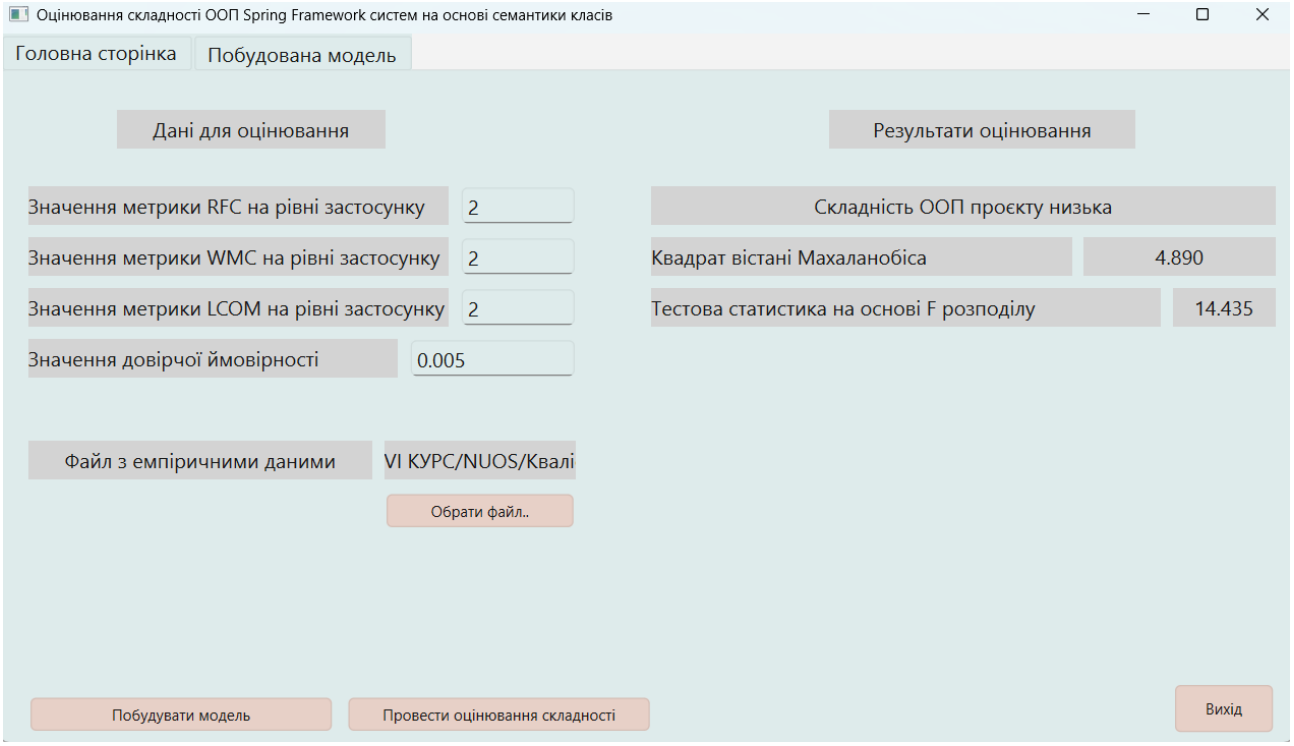


Рисунок 3.16 – Відображення результату оцінювання складності ООП, коли дані застосунку валідні та модель побудована

У результаті проведення тестування програмного забезпечення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework

систем на основі семантики класів за допомогю методу «чорної скриньки», усі випробування, описані в додатку В, пройдено успішно.

3.5 Висновки за розділом 3

В даному розділі було виконано постановку задачі на розробку проєкту програмного забезпечення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів, розроблено ескізний, технічний та робочий проєкти програмного забезпечення.

1) На етапі ескізного проєктування було виявлено акторів системи, проведено моделювання варіантів використання, побудовано діагарму діяльності варіантів використання програмного забезпечення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів.

2) На етапі створення технічного проєкту було виконано проєктування програми, виявлено модулі програмного забезпечення, визначено їх функціонал, вхідну та вихідну інформацію.

3) В робочому проєкті було аргументовано обрано набір програмних засобів для реалізації програмного забезпечення та розроблено програмне забезпечення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів. Під час випробування розробленого програмного забезпечення було доведено його працездатність та коректність розробленої моделі.

4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВПРОВАДЖЕННЯ ПРОГРАМИ

Одним із ключових етапів під час створення програмного забезпечення для розробника з економічної позиції є визначення його собівартості. Програмний продукт належить до особливої категорії товарів, які мають чимало специфічних властивостей.

Процес розробки ПЗ передбачає разові витрати на його створення, інвестиції в необхідні технічні ресурси, а також постійні витрати, пов'язані з подальшою підтримкою та експлуатацією системи. Економічна вигода від використання програми обчислюється з урахуванням усіх витрат на її функціонування. Співвідношення цієї вигоди до початкових інвестицій у розробку визначає рівень економічної результативності капітальних вкладень.

Усі економічні показники розраховують відповідно до актуальних на момент проведення розрахунків оптових цін, тарифних ставок та рівня заробітної плати.

4.1 Розрахунок витрат на створення та впровадження програмного забезпечення

Розрахунок витрат на створення та впровадження програмного забезпечення складається з витрат на заробітну плату розробникам, амортизацію та використання персонального комп'ютера, засоби розробки та матеріали і комплектуючі [28]. Розробка програмного забезпечення виконується спеціалістом із місячним окладом 20000 грн. Додаткова заробітня плата складає 10% від основної. Виходячи з цього, основна і додаткова заробітна плата спеціаліста складає 22000 грн, а вартість сучасного ноутбука складає 20000 грн. Ціна за кіловат-годину електроенергії становить 4,32 грн. Витрати на допоміжні матеріали наведено в таблиці 4.1

Таблиця 4.1 – Витрати на допоміжні матеріали

Пункт витрат	Сума, грн
Папір	190
Заправлення картриджа до принтера	250
Непередбачені витрати	300
Разом матеріали і комплектуючі	740

Вартість розробки програмної системи розраховується за формулою [29]:

$$C_{\text{пр}} = (З_{\text{зп}} + З_{\text{сз}} + З_{\text{зг}} + З_{\text{е}}) * T + З_{\text{М}'}, \qquad (4.1)$$

де $З_{\text{зп}}$ – основна і додаткова заробітна плата обслуговуючого персоналу, грню; $З_{\text{сз}}$ – відрахування на соціальні заходи (38% від основної та додаткової заробітної плати), грн.; $З_{\text{зг}}$ – загальногосподарські витрати (10% від заробітної плати), грн.; $З_{\text{е}}$ – витрати на електроенергію; $З_{\text{М}'}$ – витрати на основні та допоміжні матеріали; T – тривалість розробки, міс.

Розрахуємо $З_{\text{е}}$ при споживчій потужності 65 Вт, тривалості роботи в місяць: 20 * 8 = 160 годин, вартості кіловат-годин 4,32 грн, отримаємо:

$$З_{\text{е}} = 160 * 4{,}32 * 65 * 10^{-3} = 44{,}93 \text{ грн.}$$

Усі поточні витрати предсталено в таблиці 4.2

Таблиця 4.2 – Усі поточні витрати на розробку програмного забезпечення

Найменування витрат	Одиниця виміру	Кількість
Тривалість розробки	міс.	3
Основна і додаткова заробітні плати	грн.	22000
Відрахування та соціальні заходи	грн.	8360
Загальногосподарські витрати	грн.	2200
Витрати на допоміжні матеріали	грн.	740
Витрати на електроенергію	грн.	44,93

Згідно з формулою (4.1) провдемо числовий розрахунок вартості розробки програмного забезпечення:

$$C_{\text{пр}} = (22000 + 8360 + 2200 + 44{,}93) * 3 + 740 = 98554{,}79 \text{ грн.}$$

Амортизаційні відрахунки на устаткування складають 60% балансової вартості в рік:

$$А_{об} = 20000 * 0,6 = 12000 \text{ грн.}$$

У маштабах підприємства річні витрати на основні і допоміжні матеріали визначаються у розмірі 5% вартості основного устаткування:

$$В_М = 20000 * 0,05 = 1000 \text{ грн.}$$

Річний обсяг робіт ноутбука у годинах визначається таким способом:

$$Ф_М = 253,3 * Т_з,$$

де $Т_з$ – середнє місячне завантаження устаткування (7 годин), 253,3 – середня кількість робочих днів на рік:

$$Ф_М = 253,3 * 7 = 1773,1 \text{ год.}$$

Витрати на електроенергію $З_е$ при 1773,1 годинах роботи устаткування в рік складуть:

$$З_е = 1773,1 * 65 * 10^{-3} * 4,32 = 497,89 \text{ грн.}$$

Експлуатаційні витрати на ноутбук за рік складуть:

$$З_{зр} = 12000 + 1000 + 497,89 = 13497,89 \text{ грн.}$$

Отже, за перший рік витрати на створення та експлуатацію програмного забезпечення складуть:

$$З_{се} = 98554{,}79 + 13497{,}89 = 112055{,}68 \text{ грн.}$$

4.2 Розрахунок економічної ефективності розробки

Головним показником економічної доцільності використання програмного забезпечення є підвищення ефективності роботи з інформацією — зменшення витрат на її обробку при одночасному зростанні швидкості та точності отримання результатів.

Ручне опрацювання даних спричиняє низку небажаних економічних наслідків, серед яких:

– витрати на зберігання паперової документації;

– підвищене споживання канцелярських матеріалів;

– додаткові витрати, пов’язані з виправленням помилок, що виникають через людський фактор.

Впровадження програмного забезпечення дозволяє збільшити продуктивність праці та зменшити затрати робочого часу. Окремим позитивним результатом, який складно прямо оцінити в грошах, є підвищення оперативності управління та покращення якості виконуваних процесів.

Для коректної економічної оцінки ефективності необхідно проводити розрахунки з урахуванням однакових умов: часу, актуальних цін та норм витрат.

Визначимо пряму економічну ефективність, ґрунтуючись на тому, що впровадження програмного забезпечення вивільняє одного працівника(за експертною оцінкою фахівців підприємства).

Заробітна плата одного працівника в рік складає:

$$20000 * 12 * 1 = 240000 \text{ грн.}$$

Річний економічний ефект розраховується по формулі:

$$Є_{\text{рік}} = \Delta C_n - E_n * k,$$

де ΔC_n – вивільнені кошти після впровадження системи (240000 грн.) мінус експлуатаційні витрати (13497,89 грн.); E_n – коефіцієнт ефективності (дорівнює коефіцієнту амортизації (0,6)); k – одноразові витрати на впровадження продукту (112055,68 грн.).

$$Є_{\text{рік}} = 240000 - 13497{,}89 - 112055{,}68 * 0{,}6 = 159268{,}7 \text{ грн.}$$

Строк окупності системи розраховується по формулі:

$$T = \frac{k}{\Delta C} = \frac{112055{,}68}{240000 - 13497{,}89} = 0{,}49 \text{ року}$$

Отже строк окупності програмного забезпечення складає 5,88 місяців.

У цьому розділі були здійснені розрахунки на створення й експлуатацію програмного забезпечення, а також розрахунки економічної ефективності та строку окупності програмного забезпечення. Виходячи з розрахунків, впровадження такого програмного забезпечення окупить себе протягом першого року експлуатації за 5,88 місяців. Отже, можна зробити висновок, що дане програмне забезпечення економічно вигідне та обґрунтоване.

5 ОХОРОНА ПРАЦІ

Охорона праці охоплює комплекс організаційних, технічних, правових та профілактичних заходів, спрямованих на мінімізацію ризиків виробничих травм, професійних захворювань і негативного впливу виробничих факторів на працівників. Сучасний підхід до безпеки праці передбачає інтеграцію систем менеджменту ризиками, використання міжнародних стандартів та постійне удосконалення умов праці.

Нормативно-правова база України у сфері охорони праці включає Закон України «Про охорону праці», Закон України «Про пожежну безпеку», Закон України «Про основи законодавства України про охорону здоров'я», а також підзаконні акти, ДСТУ, санітарні норми та галузеві інструкції. Закон «Про охорону праці» визначає систему регулювання взаємовідносин між роботодавцем і працівником у питаннях створення безпечного виробничого середовища відповідно до сучасних принципів управління охороною праці. [30].

5.1 Структура управління питаннями охорони праці на підприємстві

Організація системи охорони праці на підприємстві здійснюється відповідно до чинного законодавства та побудована за принципом розподілу відповідальності між різними рівнями управління. Структура включає: керівництво, службу охорони праці, комісію (комітет) з питань охорони праці та працівників підприємства.

Керівництво – виконавчі органи підприємства відповідають за формування політики безпеки, забезпечення ресурсів, створення системи управління охороною праці та контроль за її функціонуванням.

Служба охорони праці – спеціалізований підрозділ, що здійснює контроль за виконанням нормативів, проводить інструктажі, оцінює та аналізує ризики, розробляє внутрішні положення, забезпечує ведення документації.

Комітет з охорони праці – колегіальний орган, до якого входять представники адміністрації та трудового колективу. Його завдання – обговорення проблемних питань, розробка рекомендацій та участь у плануванні профілактичних заходів.

Працівники – кожен працівник зобов'язаний дотримуватися вимог безпеки, виконувати інструкції, проходити навчання та повідомляти про виявлені небезпеки.

5.2 Забезпечення техніки безпеки та охорона праці для конкретних посад або при виконанні конкретних завдань

Для працівників ІТ-сфери характерні специфічні ризики, пов’язані з тривалою роботою за комп’ютером, високою концентрацією уваги та значним емоційним навантаженням. Забезпечення безпечних умов праці вимагає дотримання ергономічних, санітарно-гігієнічних та інформаційних норм.

– Правильне ергономічне облаштування робочого місця – регулювання висоти стільця та столу, розміщення монітора на рівні очей, оптимальна організація кабельного простору.

– Профілактика зорової втоми – використання якісних моніторів із сучасними антифлікер-технологіями, дотримання правил 20-20-20, оптимальний рівень освітлення.

– Безпечна робота з обладнанням – уникнення перевантаження зап’ясть, використання ергономічних аксесуарів.

– Забезпечення кібербезпеки – навчання персоналу, застосування систем резервного копіювання, багатофакторної автентифікації та засобів захисту інформації.

– Профілактика стресу – дотримання режиму праці і відпочинку, облаштування зон релаксації, впровадження корпоративних програм підтримки ментального здоров’я.

Загальною метою забезпечення техніки безпеки та охорони праці для ІТ компаній є створення безпечного та здорового робочого середовища, яке сприяє продуктивності та добробуту працівників. Це може досягатися шляхом

постійного вдосконалення процедур та поліпшення умов праці на основі оцінки ризиків та відповідних заходів безпеки.

5.3 Аналіз шкідливих і небезпечних факторів при роботі з комп'ютером

Робота з комп'ютерною технікою супроводжується впливом низки фізичних, хімічних та психофізіологічних чинників, представлених в таблиці 5.1 [31].

Таблиця 5.1 – Фактори впливу негативних чинників при роботі за комп'ютером

Категорія	Чинники
Фізичні	Електромагнітне випромінювання, статична електрика, підвищена яскравість або мерехтіння екрана, недостатня вологість повітря, неправильне освітлення.
Хімічні	Виділення озону, фенолу, формальдегіду та летких органічних сполук з нагрітих елементів техніки; підвищена концентрація пилу.
Психофізичні	Зорове перенапруження, стрес, монотонність роботи, перевантаження опорно-рухового апарату.

Розглянемо основні шкідливі чинники, які здатні негативно позначатися на стані здоров'я та загальному самопочутті працівників під час роботи з комп'ютерною технікою. До таких факторів належать: тривале перебування у сидячій позі, вплив електромагнітних полів, підвищене навантаження на органи зору, перенапруження м'язів кистей рук, ризики розвитку захворювань дихальної системи, алергічні реакції, а також можливі ускладнення у вагітних жінок.

Тривале сидіння за робочим місцем сприяє виникненню напруження у м'язах шиї, спини, плечового пояса та верхніх кінцівок. Подібне статичне навантаження може спричиняти широкий спектр проблем – від хронічних болей у спині до таких поширених захворювань, як остеохондроз, порушення кровообігу в органах малого таза, простатит чи геморой. Недостатній рівень рухової активності нерідко є одним із чинників розвитку ожиріння.

Остеохондроз виникає внаслідок дегенеративних змін міжхребцевих дисків, які з часом можуть зміщуватися та формувати грижу. Такі зміни нерідко

тиснуть на нервові корінці або спинний мозок, що проявляється болями у спині, кінцівках, порушенням роботи внутрішніх органів, а в тяжких випадках – навіть ризиком паралічу. Однією з ключових передумов розвитку цього захворювання є недостатній тонус м’язів спини, характерний для людей, які тривалий час працюють у сидячому положенні.

Геморой також часто пов’язаний з малорухливим способом життя, надмірною вагою, нераціональним харчуванням (вживання гострих чи копчених продуктів), запальними процесами в органах малого таза тощо. Ожиріння, у свою чергу, може бути наслідком порушеного режиму харчування, низької фізичної активності, хронічного стресу, нестачі сну, гормональних змін або надмірного вживання жирної їжі. Надлишкова маса тіла підвищує навантаження на серцево-судинну систему, сприяє розвитку атеросклерозу, підвищує ризики гіпоксії через порушення роботи органів дихання.

Навантаження на органи зору формуються під впливом множинних чинників. Око чутливе до мерехтіння монітора та мікрорухів зображення, що змушує м’язи, відповідальні за акомодацію, перебувати у постійній напрузі. Це поступово знижує чіткість зору та може спричиняти короткозорість чи інші порушення. Для зменшення навантаження необхідно правильно налаштовувати монітор (контраст, яскравість, частоту оновлення), застосовувати зручні шрифти, коректно компонувати робочі вікна у програмному забезпеченні.

Тривала робота за дисплеєм змушує очі адаптуватися до зображення, яке складається з безлічі світних точок, що відрізняється від природного сприйняття друкованого тексту. Це призводить до втоми зорового аналізатора, появи головних болів, двоїння, надмірної сльозотечі, сухості очей, спотворення зображення.

Перенапруження суглобів кистей рук часто спричиняє розвиток синдрому зап’ястного каналу – захворювання, пов’язаного зі здавленням серединного нерва в області кисті через повторювані одноманітні рухи при роботі з клавіатурою та мишею.

Робота за комп’ютером також супроводжується емоційними навантаженнями. Стресові реакції виникають у випадках втрати даних,

аварійного вимкнення техніки, пошкодження файлів, дії вірусів, а також через необхідність виконання великого обсягу інформаційних завдань. Стрес може мати короткочасний чи тривалий характер, бути позитивним або негативним, але за значної інтенсивності здатен провокувати серйозні проблеми зі здоров’ям, зокрема розлади серцево-судинної системи.

Ще одним важливим фактором ризику є можливий негативний вплив на органи дихання. Під час тривалої роботи комп’ютер нагрівається, і його елементи можуть виділяти в повітря певні хімічні речовини. Окрім того, електростатичне поле притягує пил, який піднімається у повітря і може потрапляти в легені. Знижена вологість у приміщенні та деіонізація повітря також негативно впливають на дихальні шляхи.

Пил і хімічні речовини здатні спричиняти алергічні реакції різного ступеня складності – від легкого подразнення слизових оболонок до серйозних проявів, зокрема анафілактичного шоку. Нагріті компоненти комп’ютера можуть виділяти органічні речовини (наприклад, трифенілфосфат), а пил, що накопичується всередині корпусу, стає сприятливим середовищем для розвитку мікроорганізмів і грибків. Симптоми алергії можуть включати висипання на шкірі, риніт, сльозотечу, порушення дихання, зниження імунітету [32].

5.4 Розробка заходів по зменшенню дії шкідливих факторів при роботі за ПК

Для зниження впливу шкідливих чинників під час роботи за ПК необхідно дотримуватися ряду профілактичних заходів. Вони спрямовані на попередження порушень опорно-рухового апарату, зорової втоми, емоційного вигорання та ризиків, пов'язаних з електробезпекою.

Під час роботи на ПК:

− Регулярно очищати обладнання від пилу спеціальними засобами, вимикаючи техніку перед обробкою.

− Розміщувати клавіатуру та мишу на стабільній поверхні з можливістю регулювання положення.

− Не допускати розміщення рідин або металевих предметів на системному блоці чи ноутбуці.

− Дотримуватися режиму праці − не більш ніж 2 години безперервної роботи з наступною перервою тривалістю 15 хвилин.

− Під час перерв виконувати розминку для очей, кистей рук та м'язів спини.

− Не здійснювати самостійний ремонт техніки, уникати контакту з внутрішніми компонентами пристроїв.

При роботі з ПК заборонено:

− самостійно відкривати, розбирати або намагатися ремонтувати системний блок, ноутбук, монітор, клавіатуру, комп’ютерну мишу та будь-які інші елементи техніки, оскільки такі дії можуть призвести до ураження електричним струмом або пошкодження обладнання;

− вставляти будь-які сторонні предмети у вентиляційні отвори комп’ютера чи монітора, що може спричинити коротке замикання або вихід пристрою з ладу;

− розміщувати на системному блоці, ноутбуці чи периферійних пристроях металеві речі, відкриті ємності з рідиною (склянки, вази, горщики з квітами), адже потрапляння вологи всередину корпусу здатне викликати пожежу або стати причиною електротравми.

Тривалість безперервної роботи за комп’ютером не повинна перевищувати двох годин. Після такого відрізку часу необхідно зробити перерву тривалістю не менше 15 хвилин, щоб зменшити навантаження на зір та м’язи.

Якщо під час роботи виникає дискомфорт у очах, відчуття печіння, погіршується чіткість зображення або з’являються інші неприємні симптоми, користувачу слід негайно зробити коротку паузу, відвести погляд від монітора та розслабити очні м’язи.

Для зниження емоційного напруження, покращення мозкового кровообігу, зменшення втоми зорового апарату та негативних наслідків малорухливого способу життя рекомендується кілька разів протягом робочого

дня виконувати спеціальні вправи. Такі комплекси можуть включати гімнастику для очей, розтягування м’язів спини, легкі рухові вправи для рук та плечового поясу.

6 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Охорона навколишнього середовища є одним із ключових напрямів сталого розвитку сучасного суспільства, особливо в умовах стрімкого розвитку електротехніки, інформаційних та комп'ютерних технологій. Масове впровадження електронних пристроїв, комп'ютерних систем, телекомунікаційного обладнання та програмно-апаратних комплексів значно підвищує рівень техногенного навантаження на довкілля. Це проявляється у зростанні обсягів споживання електроенергії, утворенні електронних відходів, електромагнітному забрудненні, а також у непрямих викидах парникових газів, пов'язаних з роботою дата-центрів та обчислювальної інфраструктури.

В Україні питання охорони навколишнього природного середовища регулюється Законом України «Про охорону навколишнього природного середовища», який визначає правові, економічні та соціальні основи екологічної безпеки. Відповідно до цього закону, охорона довкілля є обов'язком держави, підприємств, організацій та громадян і передбачає раціональне використання природних ресурсів, запобігання негативному впливу господарської діяльності та впровадження екологічно безпечних технологій. Особливої актуальності ці положення набувають у галузі електротехніки та інформаційних технологій, де технічний прогрес повинен поєднуватися з дотриманням принципів екологічної відповідальності.

6.1 Аналіз існуючої проблеми необхідності охорони навколишнього середовища

Розвиток електротехнічних та комп'ютерних систем є невід'ємною складовою цифровізації економіки та суспільства. Водночас функціонування електронних пристроїв, комп'ютерної техніки, серверного обладнання та мережевої інфраструктури супроводжується низкою негативних екологічних чинників. Однією з основних проблем є зростання споживання електроенергії.

За даними міжнародних досліджень, інформаційно-комунікаційні технології споживають значну частку світової електроенергії, що призводить до збільшення викидів діоксиду вуглецю при використанні традиційних джерел енергії [33].

Суттєвою екологічною проблемою є також утворення електронних відходів (e-waste). До них належать застарілі комп'ютери, мобільні пристрої, електронні плати, блоки живлення, акумулятори та інші компоненти. Такі відходи містять небезпечні речовини, зокрема свинець, ртуть, кадмій та бромовані сполуки, які можуть потрапляти в ґрунт і воду, завдаючи шкоди екосистемам і здоров'ю людини. Відповідно до екологічного законодавства України, неправильне поводження з небезпечними відходами заборонене та підлягає адміністративній і кримінальній відповідальності [34].

Окрему увагу слід приділити впливу електромагнітних полів, що створюються електротехнічним обладнанням, комп'ютерними системами та засобами зв'язку. Хоча рівні електромагнітного випромінювання зазвичай не перевищують встановлених нормативів, тривала дія таких факторів потребує постійного контролю та дотримання санітарних норм. Згідно з державними стандартами та рекомендаціями Всесвітньої організації охорони здоров'я, необхідно забезпечувати безпечні умови експлуатації електронних пристроїв, особливо в робочих і навчальних приміщеннях [35].

Крім того, розробка програмного забезпечення та експлуатація інформаційних систем також мають непрямий вплив на довкілля. Великі обчислювальні системи, хмарні сервіси та дата-центри потребують значних енергетичних ресурсів і систем охолодження, що збільшує екологічне навантаження. У зв'язку з цим виникає необхідність переходу до концепції «зелених» інформаційних технологій (Green IT), яка передбачає мінімізацію негативного впливу IT-інфраструктури на навколишнє середовище.

Таким чином, аналіз існуючих проблем свідчить про нагальну потребу комплексного підходу до охорони навколишнього середовища у сфері електротехніки та комп'ютерних технологій, що включає як технічні, так і організаційні та правові заходи.

6.2 Розробка заходів щодо зменшення забруднення

З метою зменшення негативного впливу електротехнічних та інформаційних технологій на довкілля доцільно впроваджувати комплекс заходів екологічного спрямування. Одним із пріоритетних напрямів є підвищення енергоефективності електронних пристроїв і комп'ютерних систем. Це досягається шляхом використання енергоощадних компонентів, оптимізації алгоритмів роботи програмного забезпечення, а також застосування сучасних стандартів енергоспоживання, таких як Energy Star [36].

Важливим заходом є організація раціонального поводження з електронними відходами. Передбачається впровадження системи роздільного збору, повторного використання та переробки електронних компонентів. У межах підприємств і навчальних закладів доцільно створювати пункти збору застарілої техніки з подальшою передачею її спеціалізованим організаціям для утилізації. Такий підхід відповідає принципам циркулярної економіки та вимогам чинного екологічного законодавства України [34].

Зниження рівня електромагнітного забруднення досягається шляхом дотримання нормативних відстаней між робочими місцями, використання сертифікованого обладнання, екранування джерел випромінювання та регулярного контролю параметрів електромагнітних полів. Особливу увагу слід приділяти проєктуванню електротехнічних систем з урахуванням вимог електромагнітної сумісності.

У сфері програмної інженерії доцільним є впровадження принципів «зеленого програмування», які передбачають оптимізацію коду, зменшення обчислювальної складності алгоритмів і раціональне використання ресурсів. Це дозволяє знизити навантаження на апаратні засоби та, відповідно, скоротити енергоспоживання. Також перспективним є використання віртуалізації та хмарних технологій з розміщенням серверів у дата-центрах, що використовують відновлювані джерела енергії [37].

Реалізація зазначених заходів сприяє підвищенню рівня екологічної безпеки, зменшенню техногенного впливу на довкілля та відповідає положенням Закону України «Про охорону навколишнього природного

середовища». У підсумку це забезпечує гармонійне поєднання розвитку електротехніки та інформаційних технологій із принципами сталого розвитку та екологічної відповідальності.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було проведено комплексне дослідження у сфері теорії ймовірності, математичної статистики та багатовимірного статистичного аналізу, спрямоване на розроблення математичної моделі оцінювання складності об’єктно-орієнтованого проєктування систем, створених із використанням Spring Framework, на основі семантики класів. У процесі виконання роботи було опрацьовано, проаналізовано та систематизовано наукову літературу за обраним напрямом, що дозволило визначити недоліки існуючих підходів та обґрунтувати доцільність проведення дослідження.

Проведено аналіз існуючих математичних моделей оцінювання складності об’єктно-орієнтованого проєктування, на основі якого запропоновано удосконалений підхід, що ґрунтується на використанні еліпсоїда передбачення для нормалізованих значень метрик RFC, WMC та LCOM.

Для забезпечення достовірності та статистичної коректності результатів було застосовано тривимірне нормалізуюче перетворення Бокса-Кокса та оцінку відстані Махаланобіса, що дало змогу виключити викиди та підтвердити гаусівський характер розподілу даних за критерієм Мардіа.

У результаті дослідження побудовано та удосконалено рівняння еліпсоїда передбачення, яке відображає просторову взаємозалежність між основними метриками складності. Це дозволило створити математичну модель оцінювання складності об’єктно-орієнтованого проєктування Spring Framework систем, що забезпечує високу точність і стабільність оцінювання.

Запропонована модель була побудована на основі аналізу 79 відкритих застосунків із платформи GitHub, що забезпечує репрезентативність вибірки та практичну значущість отриманих результатів.

Розроблена математична модель може бути використана для виявлення архітектурних дефектів програмних систем на ранніх етапах їх життєвого

циклу, що сприятиме підвищенню якості коду, зниженню вартості супроводу та своєчасному проведенню рефакторингу.

На основі розробленої математичної моделі було створено програмне забезпечення для оцінювання складності об’єктно-орієнтованого проєктування систем, створених із використанням Spring Framework, на основі семантики класів. До розробленої програмної документації входять: технічне завдання, опис програмного забезпечення, текст програми, інструкція користувача, програма та методика випробувань. Дані документи представлено у додатках А, Б, В, Г та Д.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Khan S. A., Khan R. A. Object oriented design complexity quantification model. *Procedia Technology*. 2012. Vol. 4. P. 548–554. DOI: 10.1016/j.protcy.2012.05.087.

2. Chidamber S. R., Kemerer C. F. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*. 1994. Vol. 20, No. 6. P. 476–493.

3. Chidamber S. R., Kemerer C. F. Towards a metrics suite for object oriented design. *ACM SIGPLAN Notices*. 1991. Vol. 26, Issue 11. P. 197–211. DOI: 10.1145/118014.117970.

4. Benkaddour R. Analyzing CK Metrics Results and Quality Assessment. 2024. URL: https://medium.com/@benkaddourmed54/analyzing-ck-metrics-results-and-quality-assessment-a70ba56534f0 (дата звернення: 15.09.2025).

5. Application of transformed prediction ellipsoids for outliers detection in multivariate non-gaussian data / [S. Prykhodko, L. Makarova, K. Prykhodko et al.] // *Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)* : the 15th International Conference, IEEE, Lviv-Slavske, 2020 : proceedings. – P. 359–362. DOI: 10.1109/TCSET49122.2020.235454

6. Латанська Л., Давидов Д. Оцінювання складності об’єктно-орієнтованого проєктування Java-застосунків, створених за допомогою фреймворка Spring, з використанням метрик RFC, WMC та LCOM. *Інформаційні технології: моделі, алгоритми, системи (ITMAS – 2025)*: матеріали VI Міжнародної наук.-практ. інтернет-конф., Миколаїв, 16–17 листопада 2025 р. Миколаїв, 2025. С. 58–59.

7. K-Java: A Complete Semantics of Java. URL: https://fsl.cs.illinois.edu/publications/bogdanas-rosu-2015-popl.pdf. (дата звернення: 15.09.2025).

8. Prykhodko S., Prykhodko N., Smykodub T. A joint statistical estimation of the RFC and CBO metrics for open-source applications developed in Java. *Computer Sciences and Information Technologies: 2022 IEEE 17th International Conference*

(CSIT), Lviv, 10–12 November 2022. P. 442–445. DOI: 10.1109/CSIT56902.2022.10000457.

9. Prykhodko A., Malakhov E. Determining object-oriented design complexity due to the identifications of classes of open-source web applications created using PHP frameworks. *Radio Electronics, Computer Science, Control*. 2024. No. 2. P. 160–167.

10. Prykhodko S. B., Prykhodko N. V., Kudin O. O., Smykodub T. G. Constructing the transformed prediction ellipses on the basis of normalizing transformations for bivariate non-Gaussian data. *Problems of Information Technology*. 2017. No. 1 (021). P. 134–138. ISSN 1998-7005. URL: https://journals.kntu.kherson.ua/index.php/ppmm/issue/view/73/72 (дата звернення: 15.09.2025).

11. Johnson R. A., Wichern D. W. *Applied multivariate statistical analysis.* Pearson Prentice Hall, 2007. 800 p.

12. Mardia K. V. Measures of multivariate skewness and kurtosis with applications. *Biometrika*. 1970. Vol. 57. P. 519–530. DOI: 10.1093/biomet/57.3.519.

13. Prykhodko S. B., Shutko I. S., Prykhodko A. S. A nonlinear regression model to estimate the size of web apps created using the CakePHP framework. *Radio Electronics, Computer Science, Control*. 2021. Vol. 59, No. 4. P. 129–139. DOI: 10.15588/1607-3274-2021-4-12.

14. Aniche M. Java code metrics calculator (CK). 2015. URL: https://github.com/mauricioaniche/ck (дата звернення: 15.09.2025).

15. Johnston J., DiNardo J. *Econometric methods*. 4th ed. McGraw-Hill, 1997.

16. Afifi A. A., Azen S. P. *Statistical analysis: a computer-oriented approach*. New York; London: Academic Press, 1979. 442 p.

17. Chew V. Confidence, prediction and tolerance regions for the multivariate normal distribution. *Journal of the American Statistical Association*. 1966. Vol. 61, Issue 315. P. 605–617. DOI: https://doi.org/10.2307/2282774.

18. Diagrams App. URL: https://app.diagrams.net/ (дата звернення: 15.09.2025).

19. Booch, G., Rumbaugh, J., Jacobson, I. (2005). The Unified Modeling Language User Guide (2nd ed.). Addison-Wesley.

20. Діаграма діяльності UML. URL: https://dou.ua/forums/topic/40575/ (дата звернення: 15.09.2025)

21. Python language main page. URL: https://www.python.org (дата звернення: 15.09.2025).

21. PyCharm: the Python IDE for data science and web development. URL: https://www.jetbrains.com/pycharm/ (дата звернення: 15.09.2025).

23. PySide6. URL: https://pypi.org/project/PySide6/ (дата звернення: 15.09.2025).

24. NumPy. URL: https://numpy.org/ (дата звернення: 15.09.2025)

25. SciPy. URL: https://scipy.org/ (дата звернення: 15.09.2025)

26. Pandas. URL https://pandas.pydata.org/ (дата звернення: 15.09.2025)

27. CSV File Reading and Writing. URL: https://docs.python.org/3/library/csv.html (дата звернення: 15.09.2025)

28. Визначення економічної ефективності програмного виробу. URL: https://studwood.net/1325190/ekonomika/viznachennya_ekonomichnoyi_efektivnosti_programnogo_virobu? (дата звернення 15.09.2025)

29. Лозовська Л.І., Дудник В.В. Сучасні підходи до вартісної оцінки програмних продуктів. URL: https://eurodev.duan.edu.ua/images/PDF/2014/2/16.pdf (дата звернення 07.11.2024)

30. Про охорону праці: Закон України від 21.11.2002 р. No 229-IV. URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text (дата звернення 15.09.2025)

31. Арламов О. Ю. Конспект лекції Безпека життєдіяльності та цивільний захист. Київ: НТУ України КПІ ім. Ігоря Сікорського, 2018. С 54 – 76.

32. Шкідливі та небезпечні виробничі чинники при роботі з компютерною технікою. URL: https://studfile.net/preview/5211197/page:3/ (дата звернення 15.09.2025).

33. International Energy Agency. Data centres and data transmission networks. URL: https://www.iea.org/energy-system/buildings/data-centres-and-data-transmission-networks (дата звернення 15.09.2025).

34. Закон України «Про охорону навколишнього природного середовища». URL: https://zakon.rada.gov.ua/laws/show/1264-12 (дата звернення 15.09.2025).

35. World Health Organization. Electromagnetic fields and public health. URL: https://www.who.int/teams/environment-climate-change-and-health/radiation-and-health/non-ionizing/emf (дата звернення 15.09.2025).

36. Energy Star Program Requirements for Computers. URL: https://www.energystar.gov/sites/default/files/specs//private/Computers_Program_Requirements.pdf (дата звернення 15.09.2025).

37. Murugesan S. Green IT: Principles and Practices. IEEE IT Professional. URL: https://ieeexplore.ieee.org/document/4446673 (дата звернення 15.09.2025).

ДОДАТОК А – Технічне завдання

Вступ

Повна назва розроблюваного проєкту: Програмне забезпечення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів.

Галузь застосування: фірми, які займаються розробкою ПЗ та використовують оцінку метрик для визначення складності проєктування застосунків.

1 Підстави для розробки

Підставою для розробки програмного забезпечення є завдання на кваліфікаційну роботу на здобуття ступеня вищої освіти «магістр» зі спеціальності 121 «Інженерія програмного забезпечення» в НУК ім. адмірала Макарова.

2 Призначення розробки

2.1 Функціональне призначення

Функціональним призначенням програмного забезпечення є оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів. Це включає в себе введення емпіричних даних для аналізу, їх нормалізацію, побудову рівняння еліпсоїда передбачення для нормалізованих метрик RFC, WMC та LCOM, обчислення значення тестової статистики для квадрату відстані Махаланобіса, проведення оцінювання складності ООП системи на основі семантики класів, вивід на екран результатів обчислень.

2.2 Експлуатаційне призначення

Експлуатаційним призначенням програмного забезпечення є автоматизований аналіз об’єктно-орієнтованих проєктів, розроблених з

використанням фреймворку Spring, з метою оцінювання їх архітектурної та проєктної складності, для допомоги розробникам, менеджерам та аналітикам у прийнятті обґрунтованих рішень щодо якості, підтримуваності, складності та ефективності проєктних рішень в об’єктно-орієнтованих системах.

3 Вимоги до програмного забезпечення

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу виконуваних функцій

Програмне забезпечення повинно виконувати такі функції:

− Вводити з файлу типу CSV емпіричні дані з метрик RFC, WMC та LCOM програмних проєктів.

− Вводити з інтерфейсу програми довірчу ймовірність, метрики RFC, WMC та LCOM, за якими потрібно оцінити складність ООП системи на основі семантики класів.

− Виконувати нормалізацію емпіричних даних тривимірним перетворенням Бокса-Кокса та ітераційно вилучати знайдені викиди шляхом обчислення квадрата відстані Махаланобіса та порівняння знайденого значення із значенням статистики на основі квантилю Фішера;

− Здійснювати побудову рівняння еліпсоїда передбачення для нормалізованих метрик RFC, WMC та LCOM.

− Обчислювати значення тестової статистики для квадрата відстані Махаланобіса.

− Здійснювати оцінювання складності ООП системи на основі семантики класів із заданою ймовірністю.

− Виводити на екран значення квадрата відстані Махаланобіса для нормалізованих метрик RFC, WMC та LCOM, значення тестової статистики для квадрата відстані Махаланобіса, результати оцінювання складності ООП системи на основі семантики класів із заданою ймовірністю.

− Виводити у текстовий файл значення квадрата відстані Махаланобіса для нормалізованих метрик RFC, WMC та LCOM, значення тестової статистики

для квадрата відстані Махаланобіса, результати оцінювання складності ООП системи на основі семантики класів із заданою ймовірністю.

3.1.2 Вимоги до організації вхідних та вихідних даних

Вхідними даними є значення емпіричних даних метрик RFC, WMC та LCOM, значення довірчої ймовірності та значення метрик RFC, WMC та LCOM проєкту.

Вихідними даними є значення квадрату відстані Махаланобіса для нормалізованих метрик RFC, WMC та LCOM, значення тестової статистики для квадрата відстані Махаланобіса та результати оцінювання складності ООП системи на основі семантики класів із заданою ймовірністю.

3.2 Вимоги до надійності

3.2.1 Вимоги до забезпечення надійності функціонування програмного забезпечення

Програмне забезпечення має функціонувати безперервно на персональних обчислювальних системах. У випадку виникнення помилок у роботі комп'ютера, відновлення нормального функціонування повинно відбуватися після перезавантаження програмного продукту.

3.2.2 Час відновлення після відмови

Час відновлення після відмови, спричиненої проблемами з електроживленням або каналом зв’язку, або не фатальним збоєм операційної системи ПК, не повинен перевищувати 10 хвилин за умови відповідності експлуатаційним показникам технічних і програмних засобів. Щодо відновлення після відмови, що виникла через несправність технічних засобів або фатальний збій операційної системи, час не повинен перевищувати час, необхідний для усунення несправностей технічних засобів та переустановлення програмних засобів.

3.2.3 Вимоги до контролю вхідної та вихідної інформації

У випадку введення користувачем неправильних даних, програмне забезпечення повинно повідомити про помилку і надати можливість її виправити.

3.3 Вимоги до умов експлуатації

3.3.1 Кліматичні умови експлуатації

Умови експлуатації програми збігаються з умовами експлуатації технічних засобів на яких буде завантажено та використовуватись програма для подальшого використання.

3.3.2 Вимоги до чисельності та кваліфікації користувачів

Програма передбачена для експлуатації одним користувачем.

Необхідний рівень підготовки користувачів: мінімальні навички в користуванні комп’ютером.

3.4 Вимоги до складу і параметрів технічних засобів

Для роботи з програмою необхідний наступний склад технічних засобів з мінімальними параметрами:

− CPU: Intel® Celeron® − 2.16 МГц;

− RAM: 2 GB;

− HDD/SDD 10GB вільного місця;

− пристрої вводу-виводу: маніпулятор типу «миша», клавіатура, монітор з екраном роздільної здатності не менше ніж 1600x900 пікселів.

3.5 Вимоги до інформаційної та програмної сумісності

Програма повинна працювати під управлінням ОС Windows версії не нижче 7. Необхідна наявність інтерпретатора Python версії не нижче 3.8.

3.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

3.7 Вимоги до транспортування та зберігання

Можливе переміщення та зберігання резервних копій програмного забезпечення на будь-яких носіях інформації, при дотриманні умов експлуатації технічних засобів на яких буде завантажено та використовуватись програма.

4 Вимоги до програмної документації

Програмна документація повинна містити:

− технічне завдання;

− опис прогами;

− програму та методику випробувань;

− інструкцію користувача;

− тексти програмних модулів.

5 Техніко-економічні показники

Розрахунок техніко-економічних показників наведено у розділі 4 даної роботи. Виходячи з розрахунків, впровадження такого програмного забезпечення окупить себе протягом першого року експлуатації за 5,88 місяців.

6 Стадії та етапи розробки

Стадії та етапи розробки програмного забезпечення для оцінювання складності ООП через зв’язки між класами застосунків, що створюються за допомогою Spring Framework представлені в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки програмного забезпечення

Стадії розробки	Етапи розробки	Дата початку	Дата завершення
1	2	3	4
1 Технічне завдання	1.1 Обґрунтування необхідності розробки програми	01.09.2025	07.09.2025
	1.2 Розробка технічного завдання	08.09.2025	20.09.2025
	1.3 Затвердження технічного завдання	21.09.2025	25.09.2025
2 Ескізний проєкт	2.1 Розробка ескізного проєкту	26.09.2025	10.10.2025
	2.2 Затвердження ескізного проєкту	11.10.2025	15.10.2025

Кінець таблиці А.1

1	2	3	4
3 Технічний проєкт	3.1 Розробка технічного проєкту	16.10.2025	05.11.2025
	3.2 Затвердження технічного проєкту	06.11.2025	10.11.2025
4 Робочий проєкт	4.1 Розробка програми	11.11.2025	30.11.2025
	4.2 Розробка програмної документації	01.12.2025	06.12.2025
	4.3 Випробування програми	07.12.2025	10.12.2025

7 Порядок контролю та приймання

Контроль за аналізом та проєктуванням кожної окремої складової програмного виробництва відбувається на кожному етапі враховуючи вимоги, що визначені в технічному завданні.

У випадку виявлення помилок під час процедури приймання програмного продукту, складається акт щодо виявлених недоліків. Цей акт підписують представники замовника і розробника і затверджують керівники обох організацій – замовника та розробника. Розробник зобов'язаний усунути виявлені помилки протягом не більше ніж за 2 тижні і повідомити замовника про проведення повторної перевірки.

ДОДАТОК Б – Опис програмного забезпечення

1 Загальні відомості

Найменування: Програмне забезпечення для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів.

Позначення: ClassComplexityEvaluator.

Для роботи програмного забезпечення необхідні такі програмні продукти:

− операційна система Microsoft Windows 7 та вище;

− Python версії не нижче 3.8.

Розроблюване програмне забезпечення було написано з використанням мови програмування Python.

2 Функціональне призначення розробки

Функціональним призначенням програмного забезпечення є оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів. Це включає в себе введення емпіричних даних для аналізу, їх нормалізацію, побудову рівняння еліпсоїда передбачення для нормалізованих метрик RFC, WMC та LCOM, обчислення значення тестової статистики для квадрату відстані Махаланобіса, проведення оцінювання складності ООП системи на основі семантики класів, вивід на екран результатів обчислень.

Програмне забезпечення повинно виконувати такі функції:

− Вводити з файлу типу CSV емпіричні дані з метрик RFC, WMC та LCOM програмних проєктів.

− Вводити з інтерфейсу програми довірчу ймовірність, метрики RFC, WMC та LCOM, за якими потрібно оцінити складність ООП системи на основі семантики класів.

− Виконувати нормалізацію емпіричних даних тривимірним перетворенням Бокса-Кокса та ітераційно вилучати знайдені викиди шляхом

обчислення квадрата відстані Махаланобіса та порівняння знайденого значення із значенням статистики на основі квантилю Фішера;

− Здійснювати побудову рівняння еліпсоїда передбачення для нормалізованих метрик RFC, WMC та LCOM.

− Обчислювати значення тестової статистики для квадрата відстані Махаланобіса.

− Здійснювати оцінювання складності ООП системи на основі семантики класів із заданою ймовірністю.

− Виводити на екран значення квадрата відстані Махаланобіса для нормалізованих метрик RFC, WMC та LCOM, значення тестової статистики для квадрата відстані Махаланобіса, результати оцінювання складності ООП системи на основі семантики класів із заданою ймовірністю.

− Виводити у текстовий файл значення квадрата відстані Махаланобіса для нормалізованих метрик RFC, WMC та LCOM, значення тестової статистики для квадрата відстані Махаланобіса, результати оцінювання складності ООП системи на основі семантики класів із заданою ймовірністю.

3 Опис логічної структури

Програмне забезпечення розроблено за шаблоном MVC.

Кожен файл відповідає за конкретну частину логіки:

Btn_Handler.py – обробка подій кнопок.

FieldValidator.py – перевірка правильності введення даних.

ModelHandler.py – побудова моделі.

MainWindow.py – головне вікно програми.

FileReader.py – зчитування та запис файлів.

4 Технічні засоби

Для роботи з програмою необхідний наступний склад технічних засобів з мінімальними параметрами:

− CPU: Intel® Celeron® − 2.16 МГц;

− RAM: 2 GB;

− HDD/SDD 10GB вільного місця;

− пристрої вводу-виводу: маніпулятор типу «миша», клавіатура, монітор з екраном роздільної здатності не менше ніж 1600x900 пікселів.

5 Виклик та завантаження

Запуск програмного додатку відбуватиметься у 2 етапи:

− встановлення усіх необхідних додаткових бібліотек, для коректної роботи додатку (у директорії з проєктом відкрити термінал та ввести команду python -m pip install -r requirements.txt);

− після встановлення усіх необхідних додаткових бібліотек запустити програмний додаток командою python main.py.

6 Вхідні дані

Вхідними даними є значення емпіричних даних метрик RFC, WMC та LCOM, значення довірчої ймовірності та значення метрик RFC, WMC та LCOM проєкту.

7 Вихідні дані

Вихідними даними є значення квадрату відстані Махаланобіса для нормалізованих метрик RFC, WMC та LCOM, значення тестової статистики для квадрата відстані Махаланобіса та результати оцінювання складності ООП системи на основі семантики класів із заданою ймовірністю.

ДОДАТОК В – Програма та методика випробувань ПЗ

1 Об’єкт випробовування

Об’єктом для випробувань є програмне забезпечення для оцінювання складності ООП через зв’язки між класами застосунків, що створюються за допомогою Spring Framework.

Сфера використання: фірми, які займаються розробкою ПЗ та використовують оцінку метрик для визначення складності проєктування застосунків.

Позначення випробуваної програми: ClassComplexityEvaluator.

2 Мета випробовування

Метою випробувань є необхідність перевірки відповідності програмного забезпечення вимогам, які викладені у технічному завданні, що наводиться у додатку А.

3 Вимоги до програми

Під час випробування програмного забезпечення потрібно перевірити, чи відповідають її функціональні характеристики вимогам, які визначені та описані у розділі «Вимоги до функціональних характеристик» технічного завдання.

4 Вимоги до програмної документації

Програмна документація повинна містити:

− технічне завдання;

− опис програми;

− програму та методику випробувань;

− інструкцію користувача;

− тексти програмних модулів.

5 Засоби та порядок випробувань

Програмні засоби випробувань

До програмних засобів, необхідних для випробувань належать: ОС Windows версії не нижче 7; необхідна наявність інтерпретатора Python версії не нижче 3.8.

Апаратні засоби випробувань

До апаратних засобів, необхідних для випробувань належать компоненти персонального комп’ютера, комплектації вище за:

− CPU: Intel® Celeron® − 2.16 МГц;

− RAM: 2 GB;

− HDD/SDD 10GB вільного місця;

− пристрої вводу-виводу: маніпулятор типу «миша», клавіатура, монітор з екраном роздільної здатності не менше ніж 1600x900 пікселів.

Порядок проведення випробувань

Проведення випробування програмного забезпечення повинно відбуватися в наступному порядку:

− виконуються інструкції до кожної функції;

− робиться аналіз отриманих результатів і встановлюється відповідність програмного продукту вимогам і системним документам.

При виявлені помилок у роботі системи складається їх перелік і обговорюється термін їх виправлення розробником. Після цього замовник проводить повторне тестування в повному обсязі (можливе використання нових додатків чи тестів).

6 Методика випробувань

Перед проведенням випробувань, запускаємо програмний застосунок ClassComplexityEvaluator. Відповідно до функціональних вимог до побудованого програмного забезпечення виконується його тестування.

Виконується перевірка функціоналу, відповідності реалізованого інтерфейсу, всі можливі операції із віконними формами.

Випробування проводиться за стратегією «чорної скриньки». У таблиці В.1 наведено програму випробувань.

Таблиця В.1 – Програма випробувань

№	Дія	Очікуваний результат
1	Вибір користувачем файла з даними	Програма виводить поідомлення про успішність або неуспішність завантаження даних у застосунок
2	Внесення користувачем даних	Програма виводить повідомлення про результат внесення користувачем даних
3	Побудова моделі еліпсоїда передбачення	Програма виводить повідомлення про результат побудови моделі
4	Визначення рівня складності ООП програмного застосунку	Програма виводить результати оцінювання у текстові поля, призначені для результатів

ДОДАТОК Г – Інструкція користувача

Після завантаження архіву з програмним продуктом, його необхідно розархівувати. Далі необхідно встановити усі необхідні додаткові бібліотеки, для коректної роботи додатку (у директорії з проєктом відкрити термінал та ввести команду python -m pip install -r requirements.txt). Після встановлення усіх необхідних додаткових бібліотек запустити програмний додаток командою python main.py.

Після запуску з’явиться головна сторінка програми, зображена на рисунку Г.1.

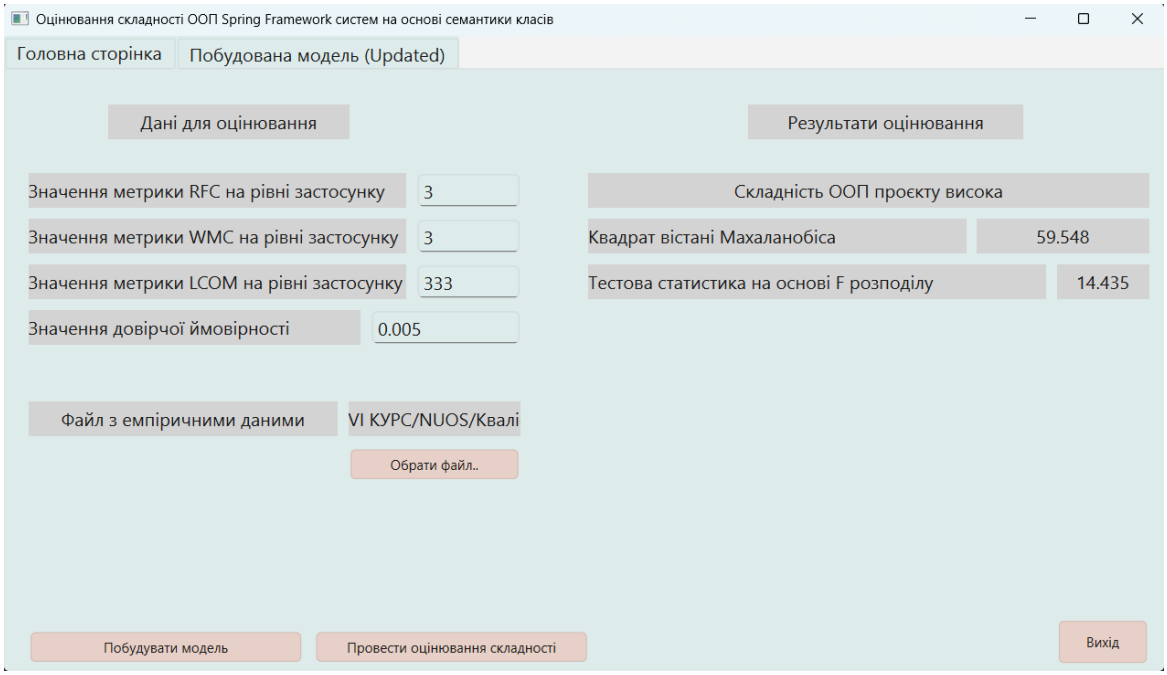


Рисунок Г.1 – Скріншот головної сторінки програмного застосунку

Далі, на початку роботи з програмою необхідно заповнити всі відповідні текстові поля: «Значення метрики RFC на рівні застосунку» − числове значення з плаваючою точкою, «Значення метрики WMC на рівні застосунку» − числове значення з плаваючою точкою, «Значення метрики LCOM на рівні застосунку» − числове значення з плаваючою точкою, «Значення довірчої ймовірності» − числове значення з плаваючою точкою.

Функціональна кнопка «Обрати файл..» відповідає за виклик вікна для вибору файлу з емпіричними даними для побудови моделі. Кнопка «Очистити поля» відповідає за безпосереднє очищення всіх текстових полів застосунку та видалення існуючої моделі для змоги побудови іншої.

Далі необхідно натиснути кнопку «Побудувати модель». Якщо всі вхідні дані занесено коректно, ми побачимо зміну назви сторінки з «Побудована» на «Побудована модель (Updated)». Вікно програми з побудованою моделлю зображено на рисунку Г.2.

	RFC	WMC	LCOM	SMD	Складність ООП
1	8.5682	4.6136	5.1136	1.1068	Низька
2	2.3214	5.5714	10.3929	4.3845	Низька
3	4.5789	5.4737	9.8421	0.6776	Низька
4	7.4444	7.4815	14.4074	0.4692	Низька
5	3.8182	3.0303	4.4167	2.0254	Низька
6	7.9474	9.7105	30.1579	1.2670	Низька
7	5.2125	7.2875	39.9750	2.5902	Низька
8	8.5333	14.4000	51.5333	4.5365	Низька
9	5.5723	11.9518	91.7952	4.5709	Низька
10	8.8571	10.8929	25.5714	2.4334	Низька
11	6.8333	8.6481	18.2407	1.2241	Низька
12	4.1118	2.6770	1.2981	1.4236	Низька
13	3.9333	6.4800	20.9733	2.2842	Низька
14	4.3333	6.1111	15.1944	1.3261	Низька
15	9.9909	12.3617	63.2401	2.5699	Низька
16	4.7368	6.9474	26.1842	1.8267	Низька

Рисунок Г.2 – Вікно програми з побудованою моделлю

Для отримання результатів оцінювання складності ООП нашого застосунку необхідно натиснути на кнопку «Провести оцінювання складності». Після цього результати з’являться у текстових полях блоку «Результати оцінювання». Результати Виведення результатів оцінювання складності ООП застосунку зображено на рисунку Г.3.

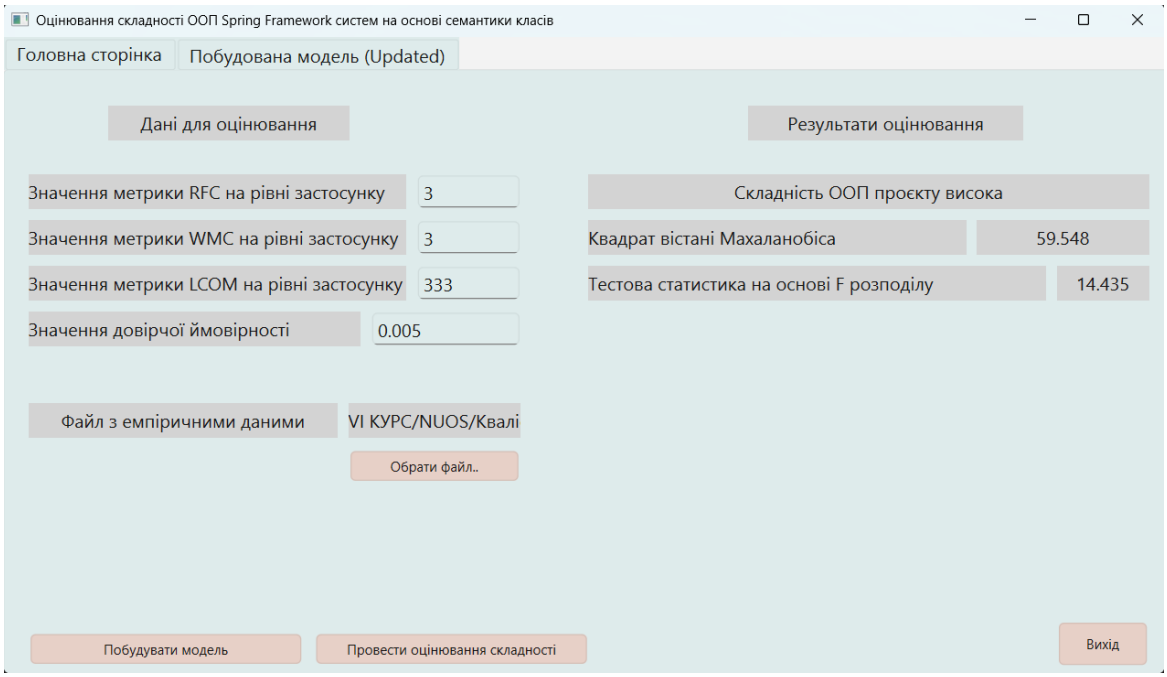


Рисунок Г.3 – Виведення результатів оцінювання складності ООП застосунку

Для виходу із програмного застосунку можна скористатись кнопкою «Вихід», яка розташована внизу справа. Це приведе до закриття всіх графічних вікон та завершення програми.

ДОДАТОК Д – Текст програми

Нижче наведено текст програми для оцінювання складності об'єктно-орієнтованого проєктування Spring Framework систем на основі семантики класів.

main.py

```
from MainWindow import configure_app

def application():
    configure_app()

if __name__ == '__main__':
    application()
```

Project.py

```
class Project:

    def __init__(self, RFC, WMC, LCOM):
        self.RFC = RFC
        self.WMC = WMC
        self.LCOM = LCOM
        self.SMD = None
        self.FStat = None
```

FieldValidator.py

```
import csv

class FieldValidator:

    def data_file_validate_csv(self, file_path, errors=None):
        errors = []
        try:
            with open(file_path, 'r', encoding='utf-8') as file:
                reader = csv.DictReader(file)
                required_fields = ['RFC', 'WMC', 'LCOM']
                for field in required_fields:
                    if field not in reader.fieldnames:
                        errors.append(f"Відсутнє поле у заголовку:
{field}")
                if errors:
```

```
                        return False, errors

                    for line_number, row in enumerate(reader, start=2):
                        try:
                            float(row['RFC'])
                            float(row['WMC'])
                            float(row['LCOM'])
                        except ValueError:
                            errors.append(f"Строка {line_number}: не всі
елементи являються числами.")
                if errors:
                    return False, errors
                return True, None
            except FileNotFoundError:
                return False, ["Файл не знайдено."]
            except Exception as e:
                return False, [f"Виникла помилка: {str(e)}"]

        def text_input_validate(self, text):
            if not text.strip():
                return False
            try:
                float(text)
                return True
            except ValueError:
                return False
```

FileReader.py

```
import csv
import pandas as pd

def read_csv_file(file_path):
    rfc = []
    wmc = []
    lcom = []
    try:
        with open(file_path, 'r', encoding='utf-8') as file:
            reader = csv.DictReader(file)
            for row in reader:
                try:
                    rfc.append(float(row['RFC']))
                    wmc.append(float(row['WMC']))
                    lcom.append(float(row['LCOM']))
                except ValueError:
                    continue
```

```
        except Exception as e:
            print(f"Помилка при читанні CSV: {e}")
        df = pd.read_csv(file_path)
        features = ['RFC', 'WMC', 'LCOM']
        data = df[features].copy()
        return data
```

ModelHandler.py

```
import pandas as pd
import numpy as np
from scipy.stats import f, boxcox
from scipy.linalg import inv
from scipy.optimize import minimize

class ModelHandler:

    def __init__(self, data, confidence_probability, project):
        self.features = ['RFC', 'WMC', 'LCOM']
        self.data = data
        self.confidence_probability = confidence_probability
        self.project = project
        self.f_threshold = self.evaluate_f_threshold(len(data),
len(self.features), self.confidence_probability)
        self.data_SMD, self.evaluated_data, self.lambdas =
self.build_model(self.data, self.confidence_probability)

    def evaluate_f_threshold(self, N, p, alpha):
        f_val = f.ppf(1 - alpha, p, N - p)
        threshold = (p * (N**2 - 1)) / (N * (N - p)) * f_val
        return threshold

    def PARAMS_box_cox_transform(self, X, lambdas):
        X = np.array(X)
        X_transformed = np.zeros_like(X)

        for j in range(X.shape[1]):
            if lambdas[j] == 0:
                X_transformed[:, j] = np.log(X[:, j])
            else:
                X_transformed[:, j] = (X[:, j] ** lambdas[j] - 1) /
lambdas[j]
        return X_transformed

    def PARAMS_inverse_box_cox_transform(X_transformed, lambdas):
        X_original = np.zeros_like(X_transformed)
        for j in range(X_transformed.shape[1]):
            if lambdas[j] == 0:
                X_original[:, j] = np.exp(X_transformed[:, j])
            else:
                X_original[:, j] = (X_transformed[:, j] * lambdas[j] + 1)
** (1 / lambdas[j])
        return X_original

    def PARAMS_log_likelihood(self, lambdas, X):
        N, k = X.shape
```

```
            X = np.maximum(X, 1e-6)

            transformed_X = self.PARAMS_box_cox_transform(X, lambdas)

            S_n = np.cov(transformed_X, rowvar=False)
            sign, log_det = np.linalg.slogdet(S_n)

            if sign <= 0:
                return np.inf

            term1 = np.sum((lambdas - 1) * np.sum(np.log(X), axis=0))
            log_likelihood_value = term1 - (N / 2) * log_det

            return -log_likelihood_value

        def PARAMS_estimate_lambda_mle(self, X):
            k = X.shape[1]
            initial_lambdas = np.ones(k)

            result = minimize(
                self.PARAMS_log_likelihood, initial_lambdas, args=(X,),
                method="L-BFGS-B", bounds=[(0.01, 5)] * k
            )

            return result.x

        def apply_boxcox(self, column, lam):
            return boxcox(column, lmbda=lam)

        def mahalanobis_squared(self, X):
            mean_vec = np.mean(X, axis=0)
            cov_matrix = np.cov(X, rowvar=False, ddof=0)
            cov_inv = inv(cov_matrix)
            diff = X - mean_vec
            d_squared = np.sum(diff @ cov_inv * diff, axis=1)
            return d_squared

        def build_model(self, data, alpha=0.005):
            X = data.copy().to_numpy()
            outliers = []
            iteration = 0

            while True:
                train_data = np.column_stack((X[:, 0], X[:, 1], X[:, 2]))

                train_lambdas = self.PARAMS_estimate_lambda_mle(train_data)
                print(train_lambdas)
                RFC_bc = self.apply_boxcox(X[:, 0], train_lambdas[0])
                WMC_bc = self.apply_boxcox(X[:, 1], train_lambdas[1])
                LCOM_bc = self.apply_boxcox(X[:, 2], train_lambdas[2])

                X_transformed = np.column_stack([RFC_bc, WMC_bc, LCOM_bc])

                d2 = self.mahalanobis_squared(X_transformed)

                threshold = self.evaluate_f_threshold(len(X),
len(self.features), alpha)
                print(threshold)
```

```
                mask = d2 > threshold

                if not np.any(mask):
                    break

                deviations = d2 - threshold
                idx_to_remove = np.argmax(deviations)

                print(f"Ітерація {iteration+1}: Видалено викид з індексом
{idx_to_remove}, значення d² = {d2[idx_to_remove]:.4f}, поріг F_Stat =
{threshold:.4f}")

                outliers.append(X[idx_to_remove])
                X = np.delete(X, idx_to_remove, axis=0)
                iteration += 1

            print(f"\nЗагальна кількість викидів: {len(outliers)}")
            return d2, pd.DataFrame(X, columns=self.features), train_lambdas

        def evaluate_project_complexity(self):
            project_RFC_bc        =        self.apply_boxcox(self.project.RFC,
self.lambdas[0])
            project_WMC_bc        =        self.apply_boxcox(self.project.WMC,
self.lambdas[1])
            project_LCOM_bc       =       self.apply_boxcox(self.project.LCOM,
self.lambdas[2])
            project_metrics_transformed   =   np.column_stack([project_RFC_bc,
project_WMC_bc, project_LCOM_bc])

            RFC_bc       =       self.apply_boxcox(self.evaluated_data['RFC'],
self.lambdas[0])
            WMC_bc       =       self.apply_boxcox(self.evaluated_data['WMC'],
self.lambdas[1])
            LCOM_bc      =      self.apply_boxcox(self.evaluated_data['LCOM'],
self.lambdas[2])
            model_metrics_transformed   =   np.column_stack([RFC_bc,   WMC_bc,
LCOM_bc])

            mean_vec = np.mean(model_metrics_transformed, axis=0)
            cov_matrix   =   np.cov(model_metrics_transformed,   rowvar=False,
ddof=0)
            cov_inv = inv(cov_matrix)
            diff = project_metrics_transformed - mean_vec
            project_SMD = np.sum(diff @ cov_inv * diff, axis=1)[0]

            self.project.SMD = project_SMD
            self.project.FStat = self.f_threshold

            complexity = ''

            if    project_SMD    <    self.evaluate_f_threshold(len(self.data),
len(self.features), 0.05):
                complexity = "Складність ООП проєкту низька"
            elif project_SMD < self.f_threshold:
                complexity = "Складніть ООП проєкту середня"
            else:
                complexity = "Складність ООП проєкту висока"
```

```
            return project_SMD, complexity
```

BtnHandler.py

```
from PySide6 import QtCore
from PySide6.QtWidgets import QMessageBox, QApplication, QFileDialog

from FieldValidator import FieldValidator
from Project import Project
from File_Reader import read_csv_file
from ModelHandler import ModelHandler
from GraphicalRenderer import GraphicalRenderer

class Btn_Handler:

    file_name = ""

    def __init__(self, main_window):
        self.model_handler = None
        self.main_window = main_window
        self.field_validator = FieldValidator()
        self.graphical_renderer = GraphicalRenderer()

    def btn_selectfile_handler(self):
        file_path, _ = QFileDialog.getOpenFileName(
            self.main_window,
            'Вибір файлу',
            QtCore.QDir.rootPath(),
            'CSV файли (*.csv)'
        )
        if file_path:
            is_valid, errors =
self.field_validator.data_file_validate_csv(file_path)
            if is_valid:
                self.file_name = file_path
                self.main_window.ui.label_filepath.setText(str(file_path))
            else:
                for error in errors:
                    print(f"- {error}")
        else:
            self.main_window.ui.label_filepath.setText("File not
checked!")

    def btn_build_model_handler(self):
        project_rfc = float(self.main_window.ui.input_project_rfc.text())
        project_wmc = float(self.main_window.ui.input_project_wmc.text())
        project_lcom =
float(self.main_window.ui.input_project_lcom.text())
        project_confidence_probability =
float(self.main_window.ui.input_confidence.text())

        project = Project(project_rfc, project_wmc, project_lcom)

        data = read_csv_file(self.file_name)
        self.model_handler = ModelHandler(data,
project_confidence_probability, project)
```

```
            self.graphical_renderer.render_result_table(self.model_handler,
self.main_window)

        def btn_complexity_eval_handler(self):
            if(self.model_handler==None):
                reply = QMessageBox.question(
                self.main_window,
                "Помилка!",
                "Дані не завантажені у застосунок!",
                QMessageBox.Cancel
                )

            else:
                project_SMD,                    complexity                    =
self.model_handler.evaluate_project_complexity()

self.main_window.ui.label_output_smd.setText(f"{self.model_handler.project.SMD:.
3f}")

self.main_window.ui.label_output_FTest.setText(f"{self.model_handler.project.FSt
at:.3f}")
                self.main_window.ui.label_output_result.setText(complexity)

        def btn_exit_handler(self):
            reply = QMessageBox.question(
                self.main_window,
                "Вихід із застосунку",
                "Ви впевнені, що хочете вийти?",
                QMessageBox.Yes | QMessageBox.No,
                QMessageBox.No
            )

            if reply == QMessageBox.Yes:
                QApplication.instance().quit()
```

GraphicalRenderer.py

```
from PySide6.QtWidgets import QTableWidget, QTableWidgetItem, QVBoxLayout
from PySide6.QtGui import QColor, QBrush


class GraphicalRenderer:


    def render_result_table(self, metrics_handler, main_window):
        table = QTableWidget()
        table.setColumnCount(5)
        table.setHorizontalHeaderLabels(["RFC",   "WMC",   "LCOM",   "SMD",
"Складність ООП"])

        table.setRowCount(len(metrics_handler.data_SMD))

        for i, smd in enumerate(metrics_handler.data_SMD):
            rfc = metrics_handler.evaluated_data.iloc[i]["RFC"]
            wmc = metrics_handler.evaluated_data.iloc[i]["WMC"]
```

```
                lcom = metrics_handler.evaluated_data.iloc[i]["LCOM"]

                table.setItem(i, 0, QTableWidgetItem(f"{rfc:.4f}"))
                table.setItem(i, 1, QTableWidgetItem(f"{wmc:.4f}"))
                table.setItem(i, 2, QTableWidgetItem(f"{lcom:.4f}"))

                item_smd = QTableWidgetItem(f"{smd:.4f}")
                table.setItem(i, 3, item_smd)

                if                              smd                             <
metrics_handler.evaluate_f_threshold(len(metrics_handler.data),
len(metrics_handler.features), 0.05):
                    level = "Низька"
                    color = QColor(144, 238, 144)
                elif smd < metrics_handler.f_threshold:
                    level = "Висока"
                    color = QColor(255, 200, 100)
                else:
                    level = "Не можливо оцінити складність"
                    color = QColor(255, 100, 100)

                item_level = QTableWidgetItem(level)
                item_level.setBackground(QBrush(color))
                table.setItem(i, 4, item_level)

            layout = QVBoxLayout()
            layout.addWidget(table)
            main_window.ui.tab_builded_model.setLayout(layout)
            main_window.ui.highlight_data_tab()
```

MainWindow.py

```
import sys

from PySide6 import QtCore
from PySide6.QtCore import QMetaObject, QRect, Qt
from PySide6.QtGui import QFont
from    PySide6.QtWidgets    import    QApplication,    QLabel,    QLineEdit,
QMainWindow, QPushButton, QTabWidget, QWidget

from Btn_Handler import Btn_Handler

class MainWindow(object):
    def setupUi(self, MainWindow):
        if not MainWindow.objectName():
```

```
            MainWindow.setObjectName(u"MainWindow")
        MainWindow.resize(1017, 557)

        MainWindow.ui = self
        self.btn_handler = Btn_Handler(MainWindow)

        self.centralwidget = QWidget(MainWindow)
        self.centralwidget.setObjectName(u"centralwidget")
        self.tab_widget = QTabWidget(self.centralwidget)
        self.tab_widget.setObjectName(u"tab_widget")
        self.tab_widget.setGeometry(QRect(0, 0, 1021, 561))
        font = QFont()
        font.setPointSize(12)
        font.setItalic(False)
        font.setUnderline(False)
        font.setStrikeOut(False)
        self.tab_widget.setFont(font)
        self.tab_widget.setStyleSheet(u"background-color:   rgb(222,   235,
235);")
        self.tab_main_page = QWidget()
        self.tab_main_page.setObjectName(u"tab_main_page")
        self.label_project_data = QLabel(self.tab_main_page)
        self.label_project_data.setObjectName(u"label_project_data")
        self.label_project_data.setGeometry(QRect(90, 30, 211, 30))
        font1 = QFont()
        font1.setPointSize(12)
        self.label_project_data.setFont(font1)
        self.label_project_data.setStyleSheet(u"background-color: rgb(211,
211, 211);")
        self.label_project_data.setAlignment(Qt.AlignmentFlag.AlignCenter)
        self.label_rfc = QLabel(self.tab_main_page)
        self.label_rfc.setObjectName(u"label_rfc")
        self.label_rfc.setGeometry(QRect(20, 90, 331, 30))
        self.label_rfc.setFont(font1)
        self.label_rfc.setStyleSheet(u"background-color:   rgb(211,   211,
211);")

self.label_rfc.setAlignment(Qt.AlignmentFlag.AlignLeading|Qt.AlignmentFlag.Align
Left|Qt.AlignmentFlag.AlignVCenter)
        self.label_wmc = QLabel(self.tab_main_page)
        self.label_wmc.setObjectName(u"label_wmc")
        self.label_wmc.setGeometry(QRect(20, 130, 331, 30))
        self.label_wmc.setFont(font1)
        self.label_wmc.setStyleSheet(u"background-color:   rgb(211,   211,
211);")
```

```
self.label_wmc.setAlignment(Qt.AlignmentFlag.AlignLeading|Qt.AlignmentFlag.Align
Left|Qt.AlignmentFlag.AlignVCenter)
              self.label_lcom = QLabel(self.tab_main_page)
              self.label_lcom.setObjectName(u"label_lcom")
              self.label_lcom.setGeometry(QRect(20, 170, 331, 30))
              self.label_lcom.setFont(font1)
              self.label_lcom.setStyleSheet(u"background-color:   rgb(211,   211,
211);")

self.label_lcom.setAlignment(Qt.AlignmentFlag.AlignLeading|Qt.AlignmentFlag.Alig
nLeft|Qt.AlignmentFlag.AlignVCenter)
              self.label_confidence = QLabel(self.tab_main_page)
              self.label_confidence.setObjectName(u"label_confidence")
              self.label_confidence.setGeometry(QRect(20, 210, 291, 30))
              self.label_confidence.setFont(font1)
              self.label_confidence.setStyleSheet(u"background-color:    rgb(211,
211, 211);")

self.label_confidence.setAlignment(Qt.AlignmentFlag.AlignLeading|Qt.AlignmentFla
g.AlignLeft|Qt.AlignmentFlag.AlignVCenter)
              self.input_project_rfc = QLineEdit(self.tab_main_page)
              self.input_project_rfc.setObjectName(u"input_project_rfc")
              self.input_project_rfc.setGeometry(QRect(360, 90, 91, 30))
              self.input_project_rfc.setFont(font1)
              self.input_project_wmc = QLineEdit(self.tab_main_page)
              self.input_project_wmc.setObjectName(u"input_project_wmc")
              self.input_project_wmc.setGeometry(QRect(360, 130, 91, 30))
              self.input_project_wmc.setFont(font1)
              self.input_project_lcom = QLineEdit(self.tab_main_page)
              self.input_project_lcom.setObjectName(u"input_project_lcom")
              self.input_project_lcom.setGeometry(QRect(360, 170, 91, 30))
              self.input_project_lcom.setFont(font1)
              self.input_confidence = QLineEdit(self.tab_main_page)
              self.input_confidence.setObjectName(u"input_confidence")
              self.input_confidence.setGeometry(QRect(320, 210, 131, 30))
              self.input_confidence.setFont(font1)
              self.label_file_data = QLabel(self.tab_main_page)
              self.label_file_data.setObjectName(u"label_file_data")
              self.label_file_data.setGeometry(QRect(20, 290, 271, 30))
              self.label_file_data.setFont(font1)
              self.label_file_data.setStyleSheet(u"background-color:    rgb(211,
211, 211);")
              self.label_file_data.setAlignment(Qt.AlignmentFlag.AlignCenter)
              self.btn_setfilepath = QPushButton(self.tab_main_page)
```

```
            self.btn_setfilepath.setObjectName(u"btn_setfilepath")
            self.btn_setfilepath.setGeometry(QRect(300, 330, 151, 30))
            font2 = QFont()
            font2.setPointSize(9)
            self.btn_setfilepath.setFont(font2)
            self.btn_setfilepath.setStyleSheet(u"background-color:    rgb(231,
208, 199);")
            self.label_filepath = QLabel(self.tab_main_page)
            self.label_filepath.setObjectName(u"label_filepath")
            self.label_filepath.setGeometry(QRect(300, 290, 151, 30))
            self.label_filepath.setFont(font1)
            self.label_filepath.setStyleSheet(u"background-color:     rgb(211,
211, 211);")
            self.label_filepath.setAlignment(Qt.AlignmentFlag.AlignCenter)
            self.label_results = QLabel(self.tab_main_page)
            self.label_results.setObjectName(u"label_results")
            self.label_results.setGeometry(QRect(650, 30, 241, 30))
            self.label_results.setFont(font1)
            self.label_results.setStyleSheet(u"background-color: rgb(211, 211,
211);")
            self.label_results.setAlignment(Qt.AlignmentFlag.AlignCenter)
            self.label_output_result = QLabel(self.tab_main_page)
            self.label_output_result.setObjectName(u"label_output_result")
            self.label_output_result.setGeometry(QRect(510, 90, 491, 30))
            self.label_output_result.setFont(font1)
            self.label_output_result.setStyleSheet(u"background-color:
rgb(211, 211, 211);")

self.label_output_result.setAlignment(Qt.AlignmentFlag.AlignCenter)
            self.label_results_2 = QLabel(self.tab_main_page)
            self.label_results_2.setObjectName(u"label_results_2")
            self.label_results_2.setGeometry(QRect(510, 130, 331, 30))
            self.label_results_2.setFont(font1)
            self.label_results_2.setStyleSheet(u"background-color:    rgb(211,
211, 211);")

self.label_results_2.setAlignment(Qt.AlignmentFlag.AlignLeading|Qt.AlignmentFlag
.AlignLeft|Qt.AlignmentFlag.AlignVCenter)
            self.label_results_3 = QLabel(self.tab_main_page)
            self.label_results_3.setObjectName(u"label_results_3")
            self.label_results_3.setGeometry(QRect(510, 170, 401, 30))
            self.label_results_3.setFont(font1)
            self.label_results_3.setStyleSheet(u"background-color:    rgb(211,
211, 211);")
```

```
self.label_results_3.setAlignment(Qt.AlignmentFlag.AlignLeading|Qt.AlignmentFlag
.AlignLeft|Qt.AlignmentFlag.AlignVCenter)
                self.label_output_smd = QLabel(self.tab_main_page)
                self.label_output_smd.setObjectName(u"label_output_smd")
                self.label_output_smd.setGeometry(QRect(850, 130, 151, 30))
                self.label_output_smd.setFont(font1)
                self.label_output_smd.setStyleSheet(u"background-color:    rgb(211,
211, 211);")
                self.label_output_smd.setAlignment(Qt.AlignmentFlag.AlignCenter)
                self.label_output_FTest = QLabel(self.tab_main_page)
                self.label_output_FTest.setObjectName(u"label_output_FTest")
                self.label_output_FTest.setGeometry(QRect(920, 170, 81, 30))
                self.label_output_FTest.setFont(font1)
                self.label_output_FTest.setStyleSheet(u"background-color: rgb(211,
211, 211);")
                self.label_output_FTest.setAlignment(Qt.AlignmentFlag.AlignCenter)
                self.btn_complexity_eval = QPushButton(self.tab_main_page)
                self.btn_complexity_eval.setObjectName(u"btn_complexity_eval")
                self.btn_complexity_eval.setGeometry(QRect(270, 490, 241, 30))
                self.btn_complexity_eval.setFont(font2)
                self.btn_complexity_eval.setStyleSheet(u"background-color:
rgb(231, 208, 199);")
                self.btn_build_model = QPushButton(self.tab_main_page)
                self.btn_build_model.setObjectName(u"btn_build_model")
                self.btn_build_model.setGeometry(QRect(20, 490, 241, 30))
                self.btn_build_model.setFont(font2)
                self.btn_build_model.setStyleSheet(u"background-color:    rgb(231,
208, 199);")
                self.btn_exit = QPushButton(self.tab_main_page)
                self.btn_exit.setObjectName(u"btn_exit")
                self.btn_exit.setGeometry(QRect(920, 480, 81, 41))
                self.btn_exit.setFont(font2)
                self.btn_exit.setStyleSheet(u"background-color:    rgb(231,    208,
199);")
                self.tab_widget.addTab(self.tab_main_page, "")
                self.tab_builded_model = QWidget()
                self.tab_builded_model.setObjectName(u"tab_builded_model")
                self.tab_widget.addTab(self.tab_builded_model, "")
                MainWindow.setCentralWidget(self.centralwidget)

                self.tab_widget.currentChanged.connect(self.on_tab_changed)

                self.retranslateUi(MainWindow)
                self.tab_widget.setCurrentIndex(0)
```

```
            QMetaObject.connectSlotsByName(MainWindow)

            self.add_functional()

        def retranslateUi(self, MainWindow):
            _translate = QtCore.QCoreApplication.translate
            MainWindow.setWindowTitle(_translate("MainWindow",     "Оцінювання
складності ООП Spring Framework систем на основі семантики класів"))
            self.label_project_data.setText(_translate("MainWindow", "Дані для
оцінювання"))
            self.label_rfc.setText(_translate("MainWindow",  "Значення метрики
RFC на рівні застосунку"))
            self.label_wmc.setText(_translate("MainWindow",  "Значення метрики
WMC на рівні застосунку"))
            self.label_lcom.setText(_translate("MainWindow", "Значення метрики
LCOM на рівні застосунку"))
            self.label_confidence.setText(_translate("MainWindow",   "Значення
довірчої ймовірності"))
            self.label_file_data.setText(_translate("MainWindow",    "Файл    з
емпіричними даними"))
            self.btn_setfilepath.setText(_translate("MainWindow",      "Обрати
файл.."))
            self.label_filepath.setText(_translate("MainWindow",    "Файл    не
обрано"))
            self.label_results.setText(_translate("MainWindow",    "Результати
оцінювання"))
            self.label_results_2.setText(_translate("MainWindow",     "Квадрат
вістані Махаланобіса"))
            self.label_results_3.setText(_translate("MainWindow",     "Тестова
статистика на основі F розподілу"))
            self.btn_complexity_eval.setText(_translate("MainWindow",
"Провести оцінювання складності"))
            self.btn_build_model.setText(_translate("MainWindow",  "Побудувати
модель"))
            self.btn_exit.setText(_translate("MainWindow", "Вихід"))

self.tab_widget.setTabText(self.tab_widget.indexOf(self.tab_main_page),
_translate("MainWindow", "Головна сторінка"))

self.tab_widget.setTabText(self.tab_widget.indexOf(self.tab_builded_model),
_translate("MainWindow", "Побудована модель"))

        def add_functional(self):
            self.btn_setfilepath.clicked.connect(lambda:
self.btn_handler.btn_selectfile_handler())
```

```
            self.btn_build_model.clicked.connect(lambda:
self.btn_handler.btn_build_model_handler())
            self.btn_complexity_eval.clicked.connect(lambda:
self.btn_handler.btn_complexity_eval_handler())
            self.btn_exit.clicked.connect(lambda:
self.btn_handler.btn_exit_handler())

        def highlight_data_tab(self):

self.tab_widget.setTabText(self.tab_widget.indexOf(self.tab_builded_model),
"Побудована модель (Updated)")

        def on_tab_changed(self, index):
            if index != self.tab_widget.indexOf(self.tab_main_page):

self.tab_widget.setTabText(self.tab_widget.indexOf(self.tab_builded_model),
"Побудована модель")

    def configure_app():
        app = QApplication(sys.argv)
        qMainWindow = QMainWindow()
        ui = MainWindow()
        ui.setupUi(qMainWindow)
        qMainWindow.show()
        sys.exit(app.exec())
```