МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
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: Нелінійна регресійна модель для оцінювання розміру програмного забезпечення, що створюється мовою C#,та програма для її реалізації

Виконав: студент групи 6151мз
_____Доценко А.С.
(підпис, ПІБ)
Керівник роботи:
доцент кафедри ПЗАС, к.т.н, доцент.
(посада, науковий ступень вчене звання)
_____ Фаріонова Т.А.
(підпис, ПІБ)

Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
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 Доценко Андрію Сергійовичу _____
(Прізвище, ім'я, по батькові)

1. Тема роботи: Нелінійна регресійна модель для оцінювання розміру програмного забезпечення, що створюється мовою C#,та програма для її реалізації _____

Керівник роботи доцент кафедри ПЗАС, к.т.н, доцент Фаріонова Т.А. _____

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 та охорони навколишнього середовища; 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 та Практичне значення одержаних результатів, Аналіз існуючих моделей та методів, що застосовуються для оцінювання розміру програмного забезпечення, Обґрунтування актуальності дослідження, Побудова нелінійної регресійної моделі для оцінювання розміру програмного забезпечення, що створюється мовою C#, Архітектура програми, Діаграма варіантів використання, Діаграма діяльності, Діаграма класів, Діаграма послідовності, Результати розробки, 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.

К А Л Е Н Д А Р Н И Й П Л А Н

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 Доценко А.С.

(підпис) (ПІБ)

Керівник роботи Фаріонова Т.А.

(підпис) (ПІБ)

РЕФЕРАТ

Доценко Андрій Сергійович

Нелінійна регресійна модель для оцінювання розміру програмного забезпечення, що створюється мовою C#,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: 113 стор., 16 табл., 8 рис., 38 використаних джерел, 5 додатків.

Актуальність теми: C# – сучасна об’єктно-орієнтована мова програмування загального призначення, яка застосовується для розробки різноманітних типів ПЗ. Існує чимало моделей для визначення розміру ПЗ, створених під конкретні мови програмування та фреймворки. Проте через те, що ці методики враховують специфіку певної мови програмування, їх використання для інших мов може призвести до неточних або некоректних результатів. Отже, побудова нелінійної регресійної моделі для оцінювання розміру ПЗ, що створюється мовою C#, є актуальним завданням інженерії ПЗ, що сприяє підвищенню достовірності оцінювання розміру відповідного ПЗ, та має практичне значення.

Мета та завдання дослідження: підвищення достовірності оцінювання розміру ПЗ, що створюється мовою C#, та програма для її реалізації. Завдання: визначити метрики, необхідні для оцінювання розміру ПЗ, що створюється мовою C#; проаналізувати вже існуючі моделі та методи оцінювання розміру ПЗ, що створюється мовою C#; відібрати проекти ПЗ з відкритим вихідним кодом, що створюються мовою C#, які будуть використовуватися для побудови нелінійної регресійної моделі; побудувати нелінійну регресійну модель для оцінювання розміру ПЗ, що створюється мовою C#; розробити програму для

оцінювання розміру ПЗ, створеного мовою C#, яка буде реалізовувати побудовану модель.

Об'єкт дослідження: процес оцінювання розміру ПЗ, створеного мовою C#.

Предмет дослідження: нелінійна регресійна модель для оцінювання розміру ПЗ, створеного мовою C#.

Методи дослідження: методи теорії ймовірностей, математичної статистики, математичного моделювання, регресійного аналізу, об'єктно-орієнтованого програмування.

Наукова новизна: удосконалено нелінійну регресійну модель для оцінювання розміру ПЗ, створеного мовою C#, за рахунок застосування нормалізуючого перетворення на основі десяткового логарифму і викидів регресії, що дозволило підвищити достовірність оцінювання розміру ПЗ, створеного мовою C#, в порівнянні з існуючими моделями.

Практичне значення одержаних результатів: розробка програми для оцінювання розміру ПЗ, створеного мовою C#, на основі побудованої моделі.

Апробація результатів дослідження: основні положення і результати досліджень, викладені у кваліфікаційній роботі, пройшли апробацію на VIII Всеукраїнській науково-практичній інтернет-конференції студентів, аспірантів та молодих вчених за тематикою «Сучасні комп’ютерні системи та мережі в управлінні» (24 листопада 2025 р., м. Херсон, м. Хмельницький).

Публікації: основні результати кваліфікаційної роботи викладено у 1 науковій праці – матеріалах конференції.

Ключові слова: програмне забезпечення, оцінювання розміру, нелінійна регресійна модель, нормалізуюче перетворення, C#.

ABSTRACT

Dotsenko Andrii Serhiiovych

A nonlinear regression model for software size estimation created in C# and a software for its implementation

Qualification work for obtaining a master's degree in the speciality 121 «Software Engineering». Admiral Makarov National University of Shipbuilding. Mykolaiv, 2025.

Volume of work: 113 pages, 16 tables, 8 figures, 38 references, 5 appendices.

Relevance of the topic: C# is a modern, object-oriented, general-purpose programming language used for developing various types of software. Numerous models exist for estimating software size, each designed for specific programming languages and frameworks. However, since these methodologies take into account the specifics of a particular programming language, applying them to other languages may lead to inaccurate or unreliable results. Therefore, developing a nonlinear regression model for estimating the size of software written in C# is an important task in software engineering, as it improves the reliability of size estimation and has practical significance.

Purpose and objectives of the study: to improve the reliability of estimating the size of software developed in C#, and to create a program that implements this estimation. Task: determining the metrics required for estimating the size of software developed in C#; analyzing existing models and methods for estimating the size of C# software; selecting open-source C# software projects to be used for constructing the nonlinear regression model; developing the nonlinear regression model for estimating the size of C# software; creating a software tool that implements the developed model.

The object of the study: the process of size estimating of software developed in C#.

Subject of the study: a nonlinear regression model to size estimating of software developed in C#.

Methods of the study: methods of probability theory, mathematical statistics, mathematical modeling, regression analysis, object-oriented programming.

Scientific novelty: the nonlinear regression model for size estimating of C# software by using normalizing transformation based on decimal logarithm and regression outliers, which will increase the reliability of size estimating of C# software in comparison with existing models.

Practical significance of the obtained results: development of a software for size estimating of C# software, based on the built model.

Testing of the research results: the main provisions and research results set forth in the qualification work were tested at the VIII All-Ukrainian Scientific and Practical Internet Conference of Students, Postgraduates, and Young Scientists “Modern Computer Systems and Networks in Management” (November 24, 2025, Kherson, Khmelnytskyi).

Publications: the main results of qualification work are set forth in 1 scientific work – conference materials.

Keywords: software, size estimation, nonlinear regression model, normalizing transformation, C#.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 9

ВСТУП 10

1 ДОСЛІДЖЕННЯ МЕТОДІВ ТА МОДЕЛЕЙ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЩО СТВОРЮЄТЬСЯ МОВОЮ C# 14

1.1 Особливості та переваги використання мови програмування C# 14

1.2 Моделі та методи, що застосовуються для оцінювання розміру програмного забезпечення, створеного мовою C# 17

1.3 Обгрунтування актуальності дослідження по обраній темі 19

2 НЕЛІНІЙНА РЕГРЕСІЙНА МОДЕЛЬ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЩО СТВОРЮЄТЬСЯ МОВОЮ C# 24

2.1 Методи побудови нелінійних регресійних моделей для оцінювання розміру програмного забезпечення 24

2.2 Методи для попередньої обробки даних 28

2.3 Побудова нелінійної регресійної моделі для оцінювання розміру програмного забезпечення, створеного мовою C# 31

3 ПРОЕКТ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЩО СТВОРЮЄТЬСЯ МОВОЮ C# 36

3.1 Постановка задачі на розробку програми для оцінювання розміру програмного забезпечення, створеного мовою C# 36

3.2 Архітектура програми для оцінювання розміру програмного забезпечення, створеного мовою C# 38

3.3 Ескізний проект програми для оцінювання розміру програмного забезпечення, створеного мовою C# 42

3.4 Технічний проект програми для оцінювання розміру програмного забезпечення, створеного мовою C# 50

3.5 Робочий проект програми для оцінювання розміру програмного забезпечення, створеного мовою C# 55

4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВІД РОЗРОБКИ І ВПРОВАДЖЕННЯ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЩО СТВОРЮЄТЬСЯ МОВОЮ C# 62

4.1 Розрахунок витрат на створення й експлуатацію програми для оцінювання розміру програмного забезпечення, створеного мовою C# 62
4.2 Економічна ефективність розробки та впровадження програми для оцінюваня розміру програмного забезпечення, створеного мовою C# 66
5 ОХОРОНА ПРАЦІ 68
5.1 Загальні вимоги з охорони праці при роботі на комп'ютері 68
5.2 Розробка заходів щодо зменшення впливу шкідливих та небезпечних факторів при роботі на комп'ютері 71
6 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА 75
6.1 Вплив людини на навколишнє середовище 75
6.2 Розробка заходів щодо зменшення негативного впливу людини на навколишнє середовище 78
ВИСНОВКИ 82
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 84
ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЩО СТВОРЮЄТЬСЯ МОВОЮ C# 89
ДОДАТОК Б – КОД ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЩО СТВОРЮЄТЬСЯ МОВОЮ C# 95
ДОДАТОК В – ОПИС ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЩО СТВОРЮЄТЬСЯ МОВОЮ C# 106
ДОДАТОК Г – ІНСТРУКЦІЯ КОРИСТУВАЧА ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЩО СТВОРЮЄТЬСЯ МОВОЮ C# 108
ДОДАТОК Д – ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАННЯ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЩО СТВОРЮЄТЬСЯ МОВОЮ C# 111

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних

МНК – метод найменших квадратів

ПЗ – програмне забезпечення

CSV – Comma-Separated Values

LOC – a number of lines of code

MMRE – a mean magnitude of relative error

MRE – a magnitude of relative error

PRED – percentage of prediction

SMD – squared Mahalanobis distance

UML – Unified Modeling Language

WPF – Windows Presentation Foundation

ВСТУП

Актуальність теми. Сучасні мови програмування можна умовно поділити на універсальні, що підходять для розв’язання широкого спектра завдань, наприклад, C#, Java, Python, та спеціалізовані, які використовуються для більш вузького кола задач, наприклад, HTML, JavaScript, SQL. Кожна з них має власні особливості, сильні сторони та певні недоліки.

C# – це сучасна мова програмування загального призначення, створена компанією Microsoft. Вона є об’єктно-орієнтованою та застосовується для розробки різноманітних типів ПЗ: від вебсервісів і ігор, зокрема, на рушії Unity, до настільних застосунків і мобільних застосунків (Xamarin/MAUI). C# є основною мовою платформи .NET та поєднує в собі можливості C++ і простоту Java, забезпечуючи типобезпечність, автоматизоване керування пам’яттю та доступ до об’ємної стандартної бібліотеки [1].

Згідно з даними компанії TIOBE, що займається аналізом якості ПЗ, та її індексом популярності мов програмування TIOBE, C# протягом останніх років стабільно утримує п’яту позицію у рейтингу. У 2023 році C# була визнана «мовою року», отримавши нагороду за найбільше зростання популярності за один рік [2, 3].

За результатами щорічного опитуванням DOU про мови програмування, C# знаходиться на 4 місці за популярністю серед комерційних розробок. Не зважаючи на деякий спад популярності цієї мови програмування – з 13,6% у 2019 році до 11,9% у 2024 році, вона стабільно залишається у першій п`ятірці найпопулярніших мов програмування на думку IT-фахівців з України. Якщо ж розглядати безпосередньо Software Engineers, то у цій сфері C# займає 3 місце по використанню у реальних проєктах із 14,7% та 2 місце як мова, яку б обрали розробники для наступного проєкту із 11,7% [4].

Розробка дієвої системи оцінювання розміру ПЗ, створеного мовою C#, є актуальним завданням і потребує вдосконалення існуючих підходів. Існує чимало моделей для визначення розміру ПЗ, створених під конкретні мови програмування та фреймворки, наприклад, для вебзастосунків на PHP [5], ПЗ, написаного на Java [6], та інші. Проте через те, що ці методики враховують специфіку певної мови програмування, їх використання для інших мов може призвести до неточних або некоректних результатів.

Для обраної мови програмування C# також існують математичні моделі для оцінювання розміру ПЗ [7-11], детально вони будуть розглянуті далі.

Отже, зважаючи на таку кількість публікацій та моделей з обраної теми, побудова нелінійної регресійної моделі для оцінювання розміру ПЗ, що створюється мовою C#, є актуальним завданням інженерії ПЗ, що сприяє підвищенню достовірності оцінювання розміру відповідного ПЗ, та має практичне значення.

Мета і завдання дослідження. Метою роботи є підвищення достовірності оцінювання розміру ПЗ, що створюється мовою C#, та програма для її реалізації.

Завдання, які потрібно вирішити, для досягнення мети:

– визначити метрики, необхідні для оцінювання розміру ПЗ, що створюється мовою C#;

– проаналізувати вже існуючі моделі та методи оцінювання розміру ПЗ, що створюється мовою C#;

– обгрунтувати актуальності дослідження по обраній темі;

– проаналізувати методи, які застосовуються для побудови нелінійних регресійних моделей для оцінювання розміру ПЗ;

– проаналізувати методи, які застосовуються для попередньої обробки даних;

– відібрати проекти ПЗ з відкритим вихідним кодом, що створюються мовою C#, які будуть використовуватися для побудови нелінійної регресійної моделі;

– побудувати нелінійну регресійну модель для оцінювання розміру ПЗ, що створюється мовою C#;

– перевірити якість побудованої нелінійної регресійної моделі;

– розробити програму для оцінювання розміру ПЗ, створеного мовою C#, яка буде реалізовувати побудовану модель.

Об’єктом досліджень є процес оцінювання розміру ПЗ, створеного мовою C#.

Предметом досліджень є нелінійна регресійна модель для оцінювання розміру ПЗ, створеного мовою C#.

Методи дослідження. Для вирішення поставлених завдань були використані методи теорії ймовірностей, математичної статистики, математичного моделювання, регресійного аналізу, об'єктно-орієнтованого програмування.

Наукова новизна одержаних результатів полягає в удосконаленні нелінійної регресійної моделі для оцінювання розміру ПЗ, створеного мовою C#, за рахунок застосування нормалізуючого перетворення на основі десяткового логарифму і викидів регресії, що дозволило підвищити достовірність оцінювання розміру ПЗ, створеного мовою C#, в порівнянні з існуючими моделями.

Практичне значення одержаних результатів полягає у розробці програми для оцінювання розміру ПЗ, створеного мовою C#, на основі побудованої моделі.

Особистий внесок здобувача. Кваліфікаційна робота є самостійно виконаною працею. Усі результати, викладені у роботі, отримані автором особисто. У праці, яка опублікована у співавторстві [12], здобувачеві належить

побудова нелінійної регресійної моделі для оцінювання розміру ПЗ, створеного мовою C#.

Апробація результатів роботи. Основні положення і результати досліджень, викладені у кваліфікаційній роботі, пройшли апробацію на VIII Всеукраїнській науково-практичній інтернет-конференції студентів, аспірантів та молодих вчених за тематикою «Сучасні комп’ютерні системи та мережі в управлінні» (24 листопада 2025 р., м. Херсон, м. Хмельницький).

Публікації. Основні результати кваліфікаційної роботи викладено у 1 науковій праці – матеріалах конференції.

1 ДОСЛІДЖЕННЯ МЕТОДІВ ТА МОДЕЛЕЙ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЩО СТВОРЮЄТЬСЯ МОВОЮ C#

1.1 Особливості та переваги використання мови програмування C#

Як було сказано раніше, мова програмування C# – є сучасною, об’єктно-орієнтованою мовою, що була розроблена компанією Microsoft як частина платформи .NET. Вона поєднує в собі потужні можливості для розробки ПЗ з простотою та зручністю використання, що робить її привабливою для широкого кола програмістів – від початківців до професіоналів [1].

Однією з ключових особливостей C# є її об’єктно-орієнтована природа. Підтримка принципів ООП (інкапсуляція, успадкування, поліморфізм) сприяє створенню модульного, легко масштабованого та повторно використовуваного коду. Це особливо важливо при розробці великих і складних проєктів, де організація коду впливає на його підтримку та розвиток [13]. Об'єктно-орієнтована парадигма підтримується повністю, включаючи класичні інкапсуляцію, успадкування та поліморфізм. Успадкування має деякі особливості: множинне успадкування класів не підтримується, але можна використовувати множинне успадкування інтерфейсів [14].

Крім цього, C# ще підтримує такі парадигми програмування, як функціональне програмування, подійно-орієнтоване програмування (event-driven programming), узагальнене програмування (generic programming) та рефлексивне програмування (reflective programming) [15].

Мова C# має строгу типізацію, що дозволяє виявляти помилки ще на етапі компіляції, підвищуючи надійність і безпеку програм. Сильна типізація зменшує кількість помилок під час виконання та полегшує підтримку великого коду [16].

Код, написаний на C#, компілюється в проміжний байт-код, зрозумілий виконуючому середовищу CLR (або аналогічному). Код є керованим (managed) [17], що означає відсутність прямої роботи з вказівниками та пам'яттю. Виконуюче середовище надає автоматичне керування (розподіл і очищення) пам'яті за допомогою сервісу під назвою «прибиральник сміття» (garbage collector). Автоматичне керування пам’яттю через механізм збирача сміття (garbage collection) звільняє розробника від необхідності вручну розподіляти й звільняти пам’ять, що зменшує ризик витоків пам’яті та підвищує стабільність програм [18].

Завдяки розвиненому та гнучкому синтаксису, C# дозволяє писати код, який легко читається. Такі можливості синтаксису, як оператори null-об'єднання (null-coalescing assignment), null-умовні оператори (null-conditional operators), ініціалізатори автоматичних властивостей (auto-implemented properties), інтерполяція рядків (string interpolation), неявна типізація (implicitly typed local) variables), анонімні типи та багато іншого, у великій кількості присутні в цій мові, але не є обов'язковими. Така гнучкість дозволяє вибрати необхідний стиль програмування, більш класичний, там, де це продиктовано, наприклад, оптимізацією коду, або віддати перевагу лаконічному, сучасному стилю.

Опрацювання виняткових ситуацій уніфіковано за допомогою зручних мовних конструкцій try-catch-finally, а функціонал роботи з ними знаходиться в системі класів успадкованими від System.Exception [19], що впорядковує роботу як із системними, так і користувацькими винятками.

Використання узагальнених типів (generics) дозволяє створювати класи та методи з підтримкою різних типів, що робить код гнучким і дозволяє позбавитися пакування – розпакування (boxing-unboxing) на рівні середовища виконання.

У C# інтегрований Language Integrated Query (LINQ) – компонент .NET, який розширює мовні конструкції, дозволяючи писати код, на кшталт запитів

SQL. Цю можливість можна використовувати для зручної обробки та отримання даних з колекцій, масивів, XML документів, реляційних БД та інших джерел даних [20]. Підтримуються лямбда-вирази, що надає можливість створення анонімних функцій, та їх застосування у місці використання.

Що стосується мови C#, то в ній існує абстракція над асинхронним кодом – модель асинхронного програмування завдань (Task asynchronous programming model). Ця модель дає можливість писати максимально лаконічний код для створення асинхронних методів за допомогою класу Tasks та ключових слів async та await, повністю приховуючи низькорівневі механізми (і багато високорівневих механізмів) керування потоками та завданнями з бібліотеки TPL. У той же час, і в цій моделі є тонкі налаштування роботи з паралельними завданнями, що виконуються [21].

Іншою значною перевагою C# є її взаємодія з великим екосистемним середовищем .NET, що надає доступ до багатої стандартної бібліотеки класів, фреймворків та інструментів для розробки різних типів програм – від веб- і серверних застосунків до мобільних і десктопних рішень. Крім того, сучасні реалізації C# підтримують крос-платформенність, що дозволяє створювати програми для Windows, Linux та macOS, а також розробляти мобільні додатки за допомогою технологій Xamarin/.NET MAUI.

Серед інших практичних переваг мови C# варто відзначити велику спільноту розробників і наявність розвинених інструментів розробки (Visual Studio, VS Code), що значно прискорює процес створення, налагодження та підтримки програм [18].

Таким чином, поєднання об’єктно-орієнтованого підходу, автоматичного керування пам’яттю, сильної типізації, широкого набору бібліотек і крос-платформності робить мову C# універсальним інструментом для розробки ПЗ різного призначення.

1.2 Моделі та методи, що застосовуються для оцінювання розміру програмного забезпечення, створеного мовою C#

Як було зазначено раніше, для оцінювання розміру ПЗ, створеного мовою C#, існують певні математичні моделі, наприклад, приведені у роботах [7-11]. Вони грунтуються на використанні певних метрик ПЗ (software metrics) – числових характеристик програмного коду, зокрема, кількість рядків коду (lines of code), покриття коду (code coverage), зв'язність (coupling), функціональні точки (function points), та багато інших. Що ж до об'єктно-орієнтованих мов програмування, до яких належить і мова C#, можна виділити такі метрики, як кількість класів та кількість методів у класі [22].

Кількість рядків коду є метрикою, що відображає загальну кількість рядків у вихідному коді програми, написаної певною мовою програмування. Зазвичай вона позначається абревіатурою LOC (Lines of Code). Існують різні підходи до обчислення цієї метрики, зокрема виділяють фізичні та логічні рядки коду. Фізичні рядки охоплюють усі рядки програмного тексту, включно з порожніми, і є відносно простими для підрахунку. Натомість логічні рядки коду відповідають кількості виконуваних інструкцій, визначення яких часто є складним, оскільки допоміжні елементи – такі як фігурні дужки, коментарі чи порожні рядки – відіграють важливу роль у розробці та супроводі програм, хоча й не є безпосередньо виконуваними командами. Водночас на створення цих елементів також витрачається значна частина часу розробника. З практичної точки зору, метрика LOC може застосовуватися для прогнозування трудових витрат на реалізацію програмного проєкту, а також для оцінювання продуктивності роботи програмістів.

Кількість класів є метрикою, орієнтованою на програмні продукти, що створюються з використанням об’єктно-орієнтованого підходу в мовах

програмування, які підтримують цю парадигму. Як правило, число класів визначається ще на початкових етапах проєктування, що дозволяє використовувати цю метрику як незалежну змінну в моделях оцінювання розміру ПЗ. Слід зазначити, що в сучасних мовах програмування практично весь код зосереджений у межах класів: реалізуються методи, властивості, події та інші елементи, які взаємодіють між собою.

У контексті даної роботи для оцінювання розміру ПЗ, що створюється мовою C#, були використані таки метрики: кількість рядків коду у якості залежної змінної (Y), виражена у тисячах, та кількість класів (X_1) і середня кількість методів у класі (X_2) у якості незалежних змінних.

Тепер розглянемо більш детально деякі знайдені математичні моделі для оцінювання розміру ПЗ, що створюється мовою C# [7-11].

Модель, представлена у роботі [7], є трифакторною. В якості незалежних змінних було використано кількість класів, глибину дерева наслідування та зв’язність класів. В якості нормалізуючого перетворення в цій моделі було використано десятковий логарифм. Тобто з огляду на ті метрики, які планується використовувати у кваліфікаційній роботі, дана модель є непридатною для використанні в розрізі поставленої задачі.

Модель, представлена у роботі [8], використовує в якості нормалізуючого перетворення десятковий логарифм та ті ж самі незалежні фактори, що плануються і у даній кваліфікаційній роботі, але вона була побудована для окремого типу ПЗ, а саме: вебзастосунків. Зважаючи на співпадіння незалежних факторів та нормалізуючого перетворення, можливо перевірити якість цієї моделі з даними, зібраними у рамках кваліфікаційної роботи.

Модель, представлена у роботі [9], представляє собою однофакторну нелінійну регресійну модель, яка також була розроблена для певного типу ПЗ – вебзастосунків, створених за допомогою платформи ASP.NET. В якості незалежної змінної регресійної моделі було обрано кількість зв’язків між

класами проектів. В якості нормалізуючого перетворення в цій моделі також було використано десятковий логарифм. З огляду на наведені відомості, використання цієї моделі для задачі, поставленої в кваліфікаційній роботі, неможливо.

Рівняння, представлені у роботі [10], також представляють собою однофакторні нелінійні регресійні рівняння для оцінювання розміру прикладного ПЗ з відкритим кодом на C# із застосуванням універсального одновимірного нормалізуючого перетворення сім'ї S_B Джонсона та нормалізуючого перетворення на основі натурального логарифму. В якості незалежної змінної регресійних рівнянь було обрано кількість класів проектів. Тому зважаючи на кількість незалежних змінних та нормалізуючі перетворення, які були використані у розглянутій роботі, її теж неможливо використовувати у кваліфікаційній роботі.

Модель, представлена у роботі [11], в якості незалежної змінної використовує Predictive Object Points (POP), та лінійне рівняння регресії, що також робить неможливим її використання для рішення задачі, поставленої у даній кваліфікаційній роботі.

1.3 Обгрунтування актуальності дослідження по обраній темі

Для обгрунтування актуальності подальшого дослідження по обраній темі, виконаємо оцінювання розміру ПЗ, що створюється мовою C#, із використанням нелінійної регресійної моделі, яка приведена у роботі [8].

В якості вихідних емпіричних даних для побудови моделі було взято відповідні метрики з 39 проєктів ПЗ, що створюється мовою C#. Дані проєкти знаходяться у відкритому доступі на інтернет-ресурсі GitHub [23].

За допомогою вбудованого аналізатора для калькуляції метрик коду в інтегрованому середовищі розробки Visual Studio [24] для кожного проекту були розраховані наступні метрики програмного коду: KLOC – фізична кількість тисяч рядків коду програми (Y), NC – загальна кількість класів (X_1), MpC – середня кількість методів у класі (X_2). Фрагмент вихідних емпіричних даних наведено у таблиці 1.1, описову статистику вихідних емпіричних даних наведено у таблиці 1.2.

Таблиця 1.1 – Фрагмент вихідних емпіричних даних

№ проєкту	Назва проєкту	KLOC (Y)	NC (X_1)	MpC (X_2)
1	Xamarin.Android.Tools.JniMarshalMethod Generator	1,256	11	4,91
2	Java.Interop.Dynamic	1,197	14	4,79
3	Xamarin.Android.Tools.Aidl	1,244	16	3,00
4	API-merge	1,912	16	8,19
5	Xamarin.SourceWriter	1,357	18	5,50
6	SwiftCopyLibs	1,482	24	10,29
7	Java.Interop.Tools.JavaSource	1,656	24	2,42
8	Xamarin.Forms.Material.Tizen	2,550	24	4,00
9	Microsoft.Android.Sdk.ILLink	2,442	25	6,68
10	Xamarin.Forms.Material.Android	2,950	25	7,92
11	Java.Interop.Tools.JavaCallableWrappers	2,644	27	5,70
12	DylibBinder	1,462	35	3,34
13	Xamarin.Android.Tools.ApiXmlAdjuster	2,124	44	2,89
14	plist-swifty	2,462	59	1,93
15	Mono.Android.Export	5,130	66	6,12
16	Java.Interop	9,607	138	5,91
17	Tools\generator	14,234	147	5,54
18	SwiftRuntimeLibrary	12,239	151	7,16
19	Generator	14,702	154	3,32
20	Mono.Android	17,811	239	5,05

Таблиця 1.2 – Описова статистика вихідних емпіричних даних (n=39)

Назва метрики	Мінімальне значення	Максимальне значення	Середнє значення	Середньоквадратичне відхилення
KLOC (Y)	0,939	48,281	10,706	12,149
NC (X_1)	2	448	110,615	114,511
MpC (X_2)	1,93	197	10,225	30,824

Двофакторне нелінійне рівняння регресії, яке представлено у роботі [8], має вигляд:

$$\hat{Y} = 10^{1,5805} X_1^{0,7562} X_2^{0,6147}. \quad (1.1)$$

Для перевірки якості та порівняння нелінійних регресійних моделей будемо використовувати такі параметри якості: коефіцієнт детермінації R^2, середнє значення відносної похибки *MMRE*, рівень прогнозування *PRED*(0,25), формули для розрахунку яких візьмемо у роботі [25]:

$$R^2 = 1 - \left(\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \right), \quad (1.2)$$

де y_i – емпіричне значення y,

$\hat{y}$ – розрахункове значення y,

$\bar{y}$ – середнє значення y, $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$,

n – кількість значень у вибірці,

$$MMRE = \frac{1}{n}\sum_{i=1}^{N} MRE_i \,, \tag{1.3}$$

де MRE – величина відносної похибки, $MRE_i = \left|\frac{y_i - \hat{y}_i}{y_i}\right|$,

$$Pred(0{,}25) = \frac{k}{n}, \tag{1.4}$$

де k – кількість значень з $MRE \leq 0{,}25$.

Необхідні умови, за яких можна вважати якість побудованої моделі прийнятною: R^2>0,75, $MMRE$<0,25, $PRED$(0,25)>0,75 [25].

Крім перевірки якості рівняння нелінійної регресії за наведеними параметрами, потрібно також перевірити діапазони значень незалежних змінних. У разі, коли ці діапазони не співпадають, використовувати знайдену модель не можливо.

Для моделі, наведеної в [8], діапазони незалежних змінних наступні: для X_1 [2; 530], для X_2 [1,25; 13,47]. Для даних, на яких будується модель в рамках даної кваліфікаційної роботи, діапазони незалежних змінних наступні: для X_1 [2; 448], для X_2 [1,93; 197].

За отриманими результатами можна зробити такий висновок: за всіма параметрами якості R^2, $MMRE$ та $PRED$(0,25) побудована модель не відповідає необхідним умовам якості та не може бути використана для оцінювання розміру ПЗ, що створюється мовою C#. За аналізом діапазонів незалежних змінних використовувати модель, наведену у роботі [8], також не можливо.

Тобто можна зробити висновок, що подальше дослідження по обраній темі є актуальним та необхідним. Побудова нелінійної регресійної моделі для оцінювання розміру ПЗ, що створюється мовою C#, з використанням

нормалізуючого перетворення у вигляді десяткового логарифму та врахуванням викидів у даних , дозволить підвищити достовірність оцінювання розміру ПЗ, що створюється мовою C#, в порівнянні з існуючими моделями.

2 НЕЛІНІЙНА РЕГРЕСІЙНА МОДЕЛЬ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЩО СТВОРЮЄТЬСЯ МОВОЮ C#

2.1 Методи побудови нелінійних регресійних моделей для оцінювання розміру програмного забезпечення

У загальному вигляді регресійна модель для оцінювання розміру ПЗ має вигляд [25]:

$$\boldsymbol{y} = \hat{\boldsymbol{y}} + \boldsymbol{\varepsilon} = \boldsymbol{f}(\mathbf{X}) + \boldsymbol{\varepsilon}, \tag{2.1}$$

де: y – залежна змінна,

$\hat{y}$ – значення залежної змінної y, розраховане за рівнянням регресії,

ε – випадкова помилка, яка розподілена за нормальним законом,

$f(\mathbf{X})$ – функція, яка визначає вид регресійної моделі,

$\mathbf{X}$ – вектор незалежних змінних, $\mathbf{X}=\{x_1, x_2, \ldots x_n\}$.

Якщо функція $f(\mathbf{X})$ з (2.1) має нелінійний характер, отримуємо нелінійну регресійну модель, якщо функція $f(\mathbf{X})$ з (2.1) має лінійний характер, відповідно отримуємо лінійну регресійну модель.

Якщо дані для оцінювання розміру ПЗ не підпорядковуються нормальному розподілу, тоді для побудови нелінійної регресійної моделі необхідно застосовувати відповідні нормалізуючі перетворення.

Якщо існує нормалізуюче перетворення негаусівського випадкового вектору $\mathbf{P} = \{Y, X_1, X_2, \ldots, X_k\}^T$ у гаусівський випадковий вектор $\mathbf{T} = \{Z_Y, Z_1, Z_2, \ldots, Z_k\}^T$, яке задається рівнянням [25]:

$$\mathbf{T} = \boldsymbol{\psi}(\mathbf{P}), \tag{2.2}$$

тоді маємо зворотне перетворення до (2.2), яке має вигляд [25]:

$$\mathbf{P} = \boldsymbol{\psi}^{-1}(\mathbf{T}), \tag{2.3}$$

де: $\boldsymbol{\psi}$ – вектор, $\boldsymbol{\psi} = \{\psi_Y, \psi_1, \psi_2, \ldots, \psi_k\}^T$.

Найпростішим прикладом нормалізуючого перетворення (2.2) з точки зору реалізації є логарифмічне перетворення на основі десяткового логарифму, таке нормалізуюче перетворення має вигляд [25]:

$$z = \lg(x). \tag{2.4}$$

Тоді зворотне перетворення до (2.4) з урахуванням (2.3) має вигляд [25]:

$$x = 10^z. \tag{2.5}$$

У разі нормально розподілених даних, отриманих за допомогою нормалізуючого перетворення (2.4), можна побудувати багатофакторне лінійне рівняння регресії, яке має вигляд [25]:

$$\hat{Z}_Y = b_0 + b_1 Z_{X1} + b_2 Z_{X2} + \cdots + b_k Z_{Xk}, \tag{2.7}$$

де: $\hat{Z}$ – розрахункове значення залежної змінної,

b_0, b_1, b_2, …, b_k – коефіцієнти лінійного рівняння регресії,

k – кількість факторів (незалежних змінних),

X_1, X_2, …, X_k – значення незалежних змінних.

Багатофакторна лінійна регресійна модель має вигляд [25]:

$$Z_y = \hat{Z}_Y + \varepsilon = b_0 + b_1 Z_{X1} + b_2 Z_{X2} + \cdots + b_k Z_{Xk} + \varepsilon, \quad (2.8)$$

де ε – випадкова помилка, яка розподілена за нормальним законом.

Далі за лінійним рівнянням регресії (2.7) із застосуванням відповідного зворотного нормалізуючого перетворення (2.5) можна отримати нелінійне регресійне рівняння [25]:

$$\hat{Y} = \psi_Y^{-1}\left(\hat{Z}_Y\right) = \psi_Y^{-1}\left(b_0 + b_1 Z_1 + b_2 Z_2 + \ldots + b_k Z_k\right), \quad (2.9)$$

де ψ_Y – перша компонента вектору $\boldsymbol{\psi}$ з перетворення (2.2).

Аналогічно (2.8), багатофакторна нелінійна регресійна модель має вигляд [25]:

$$Y = \hat{Y} + \varepsilon = \psi_Y^{-1}\left(\hat{Z}_Y\right) + \varepsilon = \psi_Y^{-1}\left(b_0 + b_1 Z_1 + b_2 Z_2 + \ldots + b_k Z_k\right) + \varepsilon, \quad (2.10)$$

де ε – випадкова помилка, яка розподілена за нормальним законом.

Коефіцієнти лінійного рівняння регресії (2.7) b_0, b_1, b_2, …, b_k можна визначити по методу найменших квадратів (МНК) [25]:

$$\begin{aligned} b_i &= \frac{\sum(Z_{x_i} - \overline{Z_{x_i}})(Z_y - \overline{Z_y})}{\sum(Z_{x_i} - \overline{Z_{x_i}})^2}, \\ b_0 &= \overline{Z_y} - b_1\overline{Z_{x_\iota}} - b_2\overline{Z_{x_\iota}} - \cdots - b_k\overline{Z_{x_\iota}}, \end{aligned} \quad (2.11)$$

де *i* приймає значення від 1 до *k*.

Довірчий інтервал лінійної регресії має вигляд [25]:

$$\hat{Z}_y \pm t_{\alpha/2,\nu} S_{Z_y} \left\{ \frac{1}{n} + (z_X^+)^T [(Z_X^+)^T Z_X^+]^{-1} (z_X^+) \right\}^{1/2}, \quad (2.12)$$

а інтервал передбачення лінійної регресії має вигляд [25]:

$$\hat{Z}_y \pm t_{\alpha/2,\nu} S_{Z_y} \left\{ 1 + \frac{1}{n} + (z_X^+)^T [(Z_X^+)^T Z_X^+]^{-1} (z_X^+) \right\}^{1/2}, \quad (2.13)$$

де: $t_{\alpha/2,\nu}$ – квантіль *t*-розподілу Стьюдента з рівнем значущості α/2 та $\nu = n - k - 1$ ступінями вільності,

n – кількість елементів у вибірці,

$$S_{Z_y} = \left\{ \frac{1}{\nu} \sum_{i=1}^{n} \left(Z_{Y_i} - \hat{Z}_{Y_i} \right)^2 \right\}^{1/2},$$

$(Z_X^+)^T Z_X^+$ – матриця розміру *k* x *k* центрованих факторів (регресорів), яка містить значення $X_{1_i} - \overline{X}_1$, $X_{2_i} - \overline{X}_2$, …, $X_{k_i} - \overline{X}_k$ [25]:

$$(Z_X^+)^T Z_X^+ = \begin{pmatrix} S_{Z_1 Z_1} & S_{Z_1 Z_2} & \cdots & S_{Z_1 Z_k} \\ S_{Z_1 Z_2} & S_{Z_2 Z_2} & \cdots & S_{Z_2 Z_k} \\ \cdots & \cdots & \cdots & \cdots \\ S_{Z_1 Z_k} & S_{Z_2 Z_k} & \cdots & S_{Z_k Z_k} \end{pmatrix}, \quad (2.14)$$

де: $S_{Z_q Z_r} = \sum_{i=1}^{n} \left[Z_{q_i} - \bar{Z}_q \right] \left[Z_{r_i} - \bar{Z}_r \right]$, $q,r = 1,2,\ldots,k$.

Відповідні інтервали, довірчий та передбачення, для нелінійної регресії можна побудувати за допомогою формул (2.12) – (2.14) та відповідного зворотного нормалізуючого перетворення (2.5).

2.2 Методи для попередньої обробки даних

У разі, коли будується багатофакторне рівняння регресії, потрібно переконатися у відсутності мультиколінеарності, тобто залежності між незалежними змінними. У нашому випадку рівняння регресії буде двофакторним, тобто, пересвідчимося у відсутності мультиколінеарності між змінними X_1 та X_2.

Пересвідчитися у відсутності мультиколінеарності можна за значеннями діагональних елементів оберненої коваріаційної матриці k x k незалежних змінних, які називають коефіцієнтами впливу дисперсії (VIFs). Якщо значення цих коефіцієнтів лежать в межах від 1 до 5, можна зробити висновок про відсутність мультиколінеарності між незалежними змінними [26].

Оскільки дані для побудови нелінійної регресійної моделі для оцінювання розміру ПЗ, що створюється мовою C#, є багатовимірними, їх потрібно перевірити на багатовимірну нормальність розподілу, для чого можна використати багатовимірні критерії Мардіа, а саме: багатовимірну асиметрію β_1 та багатовимірний ексцес β_2 [27].

Багатовимірна асиметрія β_1 має вигляд [27]:

$$\beta_1 = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}[(X_i - \bar{X})^T S_n^{-1}(X_j - \bar{X})]^3, \tag{2.15}$$

де $X - k$-вимірний вектор змінних, $X = (X_1, X_2, \ldots, X_k)$,

$\bar{X}$ – вектор середніх значень,

S_n – вибіркова коваріаційна матриця, яка має вигляд [27]:

$$S_n = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})\,(X_i - \bar{X})^T. \tag{2.16}$$

Багатовимірний ексцес β_2 має вигляд [27]:

$$\beta_2 = \frac{1}{n}\sum_{i=1}^{n}[(X_i - \bar{X})^T S_n^{-1}(X_i - \bar{X})]^2. \quad (2.17)$$

Для перевірки вихідних емпіричних даних на багатовимірну нормальність за багатовимірними критеріями Мардіа, додатково потрібно розрахувати тестові статистики для цих критеріїв.

Тестова статистика для багатовимірної асиметрії β_1 визначається за співвідношенням [27]:

$$\frac{n*\beta_1}{6} \leq \chi^2, \quad (2.18)$$

де χ^2 – верхній квантиль розподілу χ^2 зі ступенями свободи $k(k+1)(k+2)/6$ та рівнем значущості α=0,005.

Тестова статистика для багатовимірного ексцесу β_2 визначається за співвідношенням [27]:

$$\beta_2 \leq N_{1-\alpha}(\mu, \sigma^2), \quad (2.19)$$

де вона порівнюється з квантилем нормального розподілу N із математичним сподіванням μ та дисперсією σ^2 із довірчою ймовірністю 1-α.

Якщо виконуються обидві умови, а саме: значення багатовимірної асиметрії β_1, розраховане за формулою (2.15), не перевищує значення відповідної тестової статистики, розрахованої за формулою (2.18), та значення багатовимірного ексцесу β_2, розраховане за формулою (2.17), не перевищує значення відповідної тестової статистики, розрахованої за формулою (2.19),

багатовимірні вихідні емпіричні дані відповідають багатовимірному нормальному розподілу.

Далі вихідні емпіричні дані потрібно перевірити та на багатовимірні викиди. Зробити це можна за допомогою квадрату відстані Махаланобіса (SMD), який має вигляд [28, 29]:

$$d_i^2 = (Z_i - \bar{Z})^T S_Z^{-1} (Z_i - \bar{Z}), \tag{2.20}$$

де: Z_i – i-ий компонент даних,

$\bar{Z}$ – вектор вибіркових середніх,

S_Z – коваріаційна матриця, яка має вигляд відповідно до (2.16).

Значення SMD, отримані за (2.20), можна порівняти з критерієм $\chi^2_{k,\alpha}$ Пірсона для *k* ступеней свободи та рівнем значущості α=0,005.

Дані, для яких значення SMD будуть більші, аніж квантиль розподілу$\chi^2_{k,\alpha}$, є викидами. Їх потрібно видалити з вибірки та повторити розрахунок SMD, допоки всі отримані значення не будуть меншими, аніж відповідний квантиль.

Для перевірки одновимірних даних на їх відповідність нормальному закону розподілу, також можна скористатися критерієм χ^2 Пірсона, який має вигляд [25]:

$$\chi^2 = \sum_{j=1}^{m} \frac{\left(n_j - \mathrm{n}p_j\right)^2}{\mathrm{n}p_j}, \tag{2.21}$$

де: m – кількість підінтервалів для діапазону від x_{min} до x_{max}, яка має вигляд [25]:

$$m = \log_2 n + 1 = 3{,}3 \lg n + 1,$$

n_j – кількість значень, які потрапляють у j-ий підінтервал,

p_j – ймовірність потрапляння у j-ий підінтервал,

n – кількість елементів у вибірці,

x_{min} та x_{max} – мінімальне та максимальне значення відповідно.

2.3 Побудова нелінійної регресійної моделі для оцінювання розміру програмного забезпечення, створеного мовою C#

Як вже було показано в підрозділі 1.3, використання нелінійної регресійної моделі, яка приведена в роботі [8], неможливо для оцінювання розміру ПЗ, створеного мовою C#, тому побудуємо нову нелінійну регресійну модель для оцінювання розміру ПЗ, яке створене мовою C#, на основі вихідних емпіричних даних метрики 39 проєктів відповідного ПЗ, які були взяті з інтернет-ресурсу GitHub [23] у відкритому доступі. Фрагмент вихідних емпіричних даних було наведено у таблиці 1.1, а описову статистику вихідних емпіричних даних – у таблиці 1.2.

Перевіримо вихідні емпіричні дані на відсутність мультиколінеарності, для чого розрахуємо обернену коваріаційну матрицю для незалежних змінних X_1 та X_2:

$$\begin{pmatrix} 1{,}0267 & 0{,}1656 \\ 0{,}1656 & 1{,}0267 \end{pmatrix}.$$

Як видно із наведених даних, діагональні елементи оберненої коваріаційної матриці лежать в межах від 1 до 5, а, отже, можна зробити висновок, що між незалежними змінними X_1 та X_2 відсутня

мультиколінеарність.

Перевіримо вихідні емпіричні дані на багатовимірну нормальність розподілу за допомогою багатовимірних критеріїв Мардіа згідно з (2.15) – (2.19).

Багатовимірна асиметрія β_1, розрахована згідно (2.15), дорівнює 42,80, в той час, як тестова статистика для багатовимірної асиметрії β_1, розрахована згідно (2.18), дорівнює 25,19.

Багатовимірний ексцес β_2, розрахований згідно (2.17), дорівнює 51,54, в той час, як тестова статистика для багатовимірного ексцесу β_2, розрахована згідно (2.19), дорівнює 22,93.

Із наведених даних можна зробити висновок, що вихідні емпіричні дані не відповідають багатовимірному нормальному розподілу, оскільки не виконується жодної умови перевірки згідно багатовимірним критеріям нормальності Мардіа.

Для перевірки вихідних емпіричних даних на багатовимірні викиди за допомогою SMD згідно з (2.20), використаємо нормалізацію даних за допомогою логарифмічного перетворення на основі десяткового логарифму згідно з (2.4).

Фрагмент нормалізованих даних, а також значення SMD наведено у таблиці 2.1, описову статистику нормалізованих даних наведено у таблиці 2.2.

Таблиця 2.1 – Фрагмент нормалізованих даних та значення SMD

№	Назва проєкту	Z_Y	Z_{X1}	Z_{X2}	SMD
1	2	3	4	5	6
1	Xamarin.Android.Tools.JniMarshalMethodGenerator	0,10	1,04	0,69	1,7008
2	Java.Interop.Dynamic	0,08	1,15	0,68	1,5392
3	Xamarin.Android.Tools.Aidl	0,09	1,20	0,48	27,0045
4	API-merge	0,28	1,20	0,91	1,0546
5	Xamarin.SourceWriter	0,13	1,26	0,74	0,5747
6	SwiftCopyLibs	0,17	1,38	1,01	2,2550
7	Java.Interop.Tools.JavaSource	0,22	1,38	0,38	0,4106

Закінчення таблиці 2.1

1	2	3	4	5	6
8	Xamarin.Forms.Material.Tizen	0,41	1,38	0,60	2,5151
9	Microsoft.Android.Sdk.ILLink	0,39	1,40	0,82	2,4188
10	Xamarin.Forms.Material.Android	0,47	1,40	0,90	1,0565
11	Java.Interop.Tools.JavaCallableWrappers	0,42	1,43	0,76	1,1886
12	DylibBinder	0,16	1,54	0,52	1,6008
13	Xamarin.Android.Tools.ApiXmlAdjuster	0,33	1,64	0,46	1,9023
14	plist-swifty	0,39	1,77	0,29	0,6905
15	Mono.Android.Export	0,71	1,82	0,79	2,6635
16	Java.Interop	0,98	2,14	0,77	2,1729
17	Tools\generator	1,15	2,17	0,74	6,6371
18	SwiftRuntimeLibrary	1,09	2,18	0,85	3,2955
19	Generator	1,17	2,19	0,52	1,7056
20	Mono.Android	1,25	2,38	0,70	0,8718

Таблиця 2.2 – Описова статистика нормалізованих даних (*n*=39)

Назва метрики	Мінімальне значення	Максимальне значення	Середнє значення	Середньоквадратичне відхилення
Z_Y	-0,0273	1,6838	0,7313	0,5368
Z_{X1}	0,3010	2,6513	1,7349	0,5973
Z_{X2}	0,2856	2,2945	0,7221	0,3195

Критерій $\chi^2_{k,\alpha}$ Пірсона дорівнює 12,84, максимальне значення SMD дорівнює 27,0045. Отже, виходячи із аналізу отриманих даних SMD, маємо 1 викид, який видаляємо з набору даних.

Далі знов виконаємо нормалізацію даних та перевіримо їх на багатовимірні викиди. Для *n*=38 отримаємо максимальне значення SMD, яке дорівнює 19,7271, отже знов маємо 1 викид, який видаляємо з набору даних.

Знову виконуємо нормалізацію та перевірку на багатовимірні викиди. Для *n*=37 отримаємо максимальне значення SMD, яке дорівнює 14,9917, тобто з набору даних знов видаляємо 1 викид.

Для n=36 отримаємо максимальне значення SMD, яке дорівнює 10,4016, що менше, ніж критерій $\chi^2_{k,\alpha}$ Пірсона, тобто багатовимірних викидів немає. Далі для цього кроку будуємо багатофакторне лінійне рівняння регресії згідно з (2.7), коефіцієнти якого знаходимо згідно з (2.11): b_0=-1,5124, b_1=0,9955, b_2=0,6338. Далі будуємо багатофакторну лінійну регресійну модель згідно з (2,8) та перевіряємо випадкову похибку ε на відповідність нормальному розподілу за допомогою критерію χ^2 Пірсона згідно з (2.21). Відповідний квантиль критерію χ^2 Пірсона дорівнює 7,81, максимальне значення випадкової похибки ε дорівнює 0,8032. Випадкова похибка ε відповідає нормальному розподілу. Далі будуємо інтервали – довірчий та передбачення – для лінійної регресії згідно з (2.12) – (2.14). Після цього переходимо до побудови нелінійного рівняння регресії згідно з (2.9), багатофакторної нелінійної регресійної моделі згідно з (2.10), довірчого інтервалу та інтервалу передбачення для нелінійної регресії з використанням зворотного нормалізуючого перетворення (2.5). Якщо є значення, які виходять за границі інтервалу передбачення, вони інтерпретуються як викиди та видаляються з набору даних. На даному кроці знайдено 1 таке значення, яке і було видалено з набору, що розглядається.

Для n=35 отримаємо максимальне значення SMD, яке дорівнює 10,7405, що менше, ніж критерій $\chi^2_{k,\alpha}$ Пірсона, тобто багатовимірних викидів немає. Після побудови лінійного рівняння регресії та лінійної регресійної моделі і перевірки випадкової похибки ε на відповідність нормальному розподілу (максимальне значення випадкової похибки ε дорівнює 4,1174, що менше, ніж відповідний квантиль критерію χ^2 Пірсона), побудови нелінійного рівняння регресії та багатофакторної нелінійної регресійної моделі і відповідного інтервалу передбачення, викидів не знайдено.

Таким чином, остаточно двохфакторна нелінійна регресійна модель для оцінювання розміру ПЗ, що створюється мовою C#, була побудована на основі

скоригованого набору даних, який містив 35 програмних проєктів, та має вигляд:

$$Y = 10^{\varepsilon - 1{,}5866} \mathrm{X}_1^{1{,}0151} \mathrm{X}_2^{0{,}6757}.$$

Побудована модель має наступні параметри якості, які розраховані згідно з (1.2) – (1.4): R^2=0,8613, *MMRE*=0,2825, *Pred*(0,25)=0,4857. Як видно із наведених даних, два з трьох параметрів якості, *MMRE* та *Pred*(0,25), не задовольняють необхідним вимогам, а саме: *MMRE*≤0,25 та *Pred*(0,25)≥0,75, що свідчить про необхідність застосування інших нормалізуючих перетворень, наприклад, однопараметричного перетворення Бокса-Кокса або чотирьохпараметричного перетворення Джонсона.

3 ПРОЕКТ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЩО СТВОРЮЄТЬСЯ МОВОЮ C#

3.1 Постановка задачі на розробку програми для оцінювання розміру програмного забезпечення, створеного мовою C#

Приступимо до розробки проєкту програми для оцінювання розміру ПЗ, створеного мовою C#, використовуючи побудовану раніше двохфакторну нелінійну регресійну модель для оцінювання розміру ПЗ, створеного мовою C#.

Після аналізу існуючих моделей, що застосовуються для оцінювання розміру відповідного ПЗ, було сформульовано постановку задачі на розробку програми для оцінювання розміру ПЗ, створеного мовою C#.

Вимоги до складу виконуваних функцій

Програма повинна забезпечувати виконання нижче наведених функцій:

– ввод даних про параметри нелінійної регресії: значення коефіцієнтів регресійного рівняння b_0, b_1 та b_2;

– ввод значення довірчої ймовірності α для розрахунку;

– ввод даних про ПЗ, створене мовою C#, для якого потрібно виконати оцінювання розміру: значення метрик X_1 (кількість класів) та X_2 (середня кількість методів у класі);

– оцінювання розміру ПЗ, створеного мовою C#: обчислення залежної змінної Y (кількість рядків коду у тисячах) та границь довірчого інтервалу і інтервалу прогнозування;

– збереження вхідних даних в БД;

– збереження результатів оцінювання розміру ПЗ, створеного мовою C#, в БД;

– експорт результатів розрахунків у файл формату .CSV.

Вимоги до організації вхідних та вихідних даних

Вимоги до організації вхідних даних

Вхідні дані: параметри нелінійної регресії b_0, b_1 та b_2, значення довірчої ймовірності α, значення метрик X_1 та X_2.

Вхідні дані вносяться у відповідні поля форм згідно з допустимим типом даних: числовий.

Вимоги до організації вихідних даних

Вихідні дані: прогнозне значення розміру ПЗ, створеного мовою C#, значення границь довірчого інтервалу та інтервалу прогнозування.

Вихідні дані виводяться у відповідні поля форм та зберігаються у БД.

Вимоги до складу та параметрів технічних засобів

Система повинна задовольняти вказаним вимогам на комп’ютері наступної мінімальної комплекції:

– CPU: Intel(R) i5 2,16 МГц або аналогічний за параметрами AMD;

– RAM: 4 GB;

– HDD: 10 GB вільного місця.

Вимоги до інформаційної та програмної сумісності

Для повноцінного функціонування програми необхідно використовувати ОС Windows версії не нижче 10, .Net Framework 4.8.1.

Детальна постановка задачі сформульована в технічному завданні, яке наведене у Додатку А.

3.2 Архітектура програми для оцінювання розміру програмного забезпечення, створеного мовою C#

Загальна характеристика архітектури

Програма для оцінювання розміру ПЗ, створеного мовою C#, реалізується у вигляді віконного застосунку та має багаторівневу (шарову) архітектуру. Обрана архітектура забезпечує чітке розділення відповідальностей між компонентами програми, підвищує зручність супроводу, масштабованість та повторне використання програмного коду.

Функціональне призначення програми полягає в обчисленні оцінки розміру ПЗ на основі нелінійної регресійної моделі, де залежною змінною є кількість рядків коду (Y), виражена у тисячах, а незалежними змінними – кількість класів (X1) та середня кількість методів у класі (X2). Усі значення метрик вводяться користувачем вручну через графічний інтерфейс, без автоматичного аналізу вихідного коду.

Загальна діаграма компонентів програми для оцінювання розміру ПЗ, що створюється мовою C#, представлена на рисунку 3.1.

Загальна структура програми

Архітектура програми складається з таких основних рівнів:

– рівень представлення (Presentation Layer);

– рівень прикладної логіки (Application / Business Logic Layer);

– рівень доступу до даних (Data Access Layer);

– рівень експорту результатів (Export Layer).

Кожен із зазначених рівнів виконує чітко визначені функції та взаємодіє з іншими рівнями через добре визначені інтерфейси.

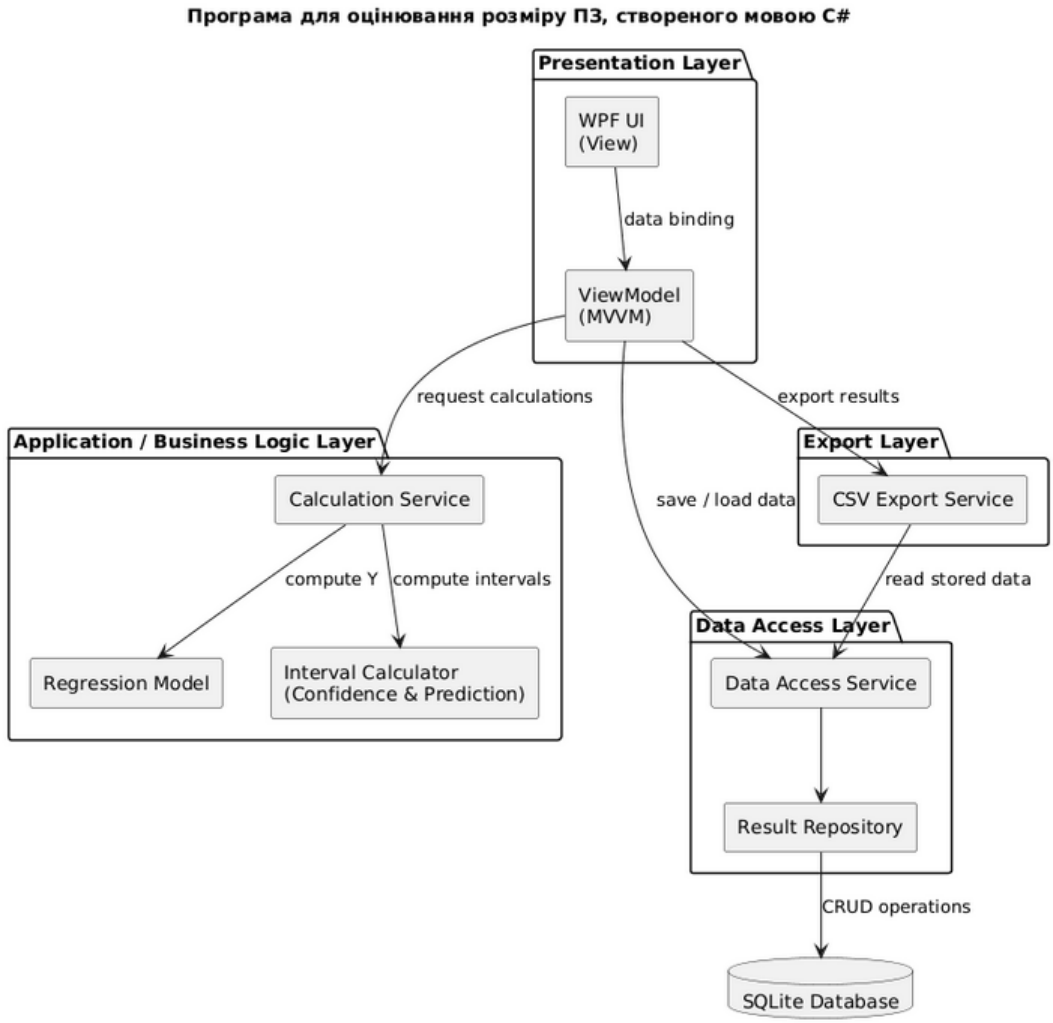

Рисунок 3.1 – Загальна діаграма компонентів програми для оцінювання розміру ПЗ, що створюється мовою C#

Рівень представлення (Presentation Layer)

Рівень представлення відповідає за взаємодію з користувачем і реалізується у вигляді графічного інтерфейсу віконного застосунку.

Для реалізації графічного інтерфейсу користувача доцільно обрати технологію WPF, оскільки вона забезпечує чітке розділення логіки представлення та прикладної логіки, підтримує патерн MVVM (Model-View-ViewModel) та дозволяє створювати масштабовані й зручні у супроводі настільні застосунки.

Основні функції даного рівня:

– введення значень коефіцієнтів регресійного рівняння b0, b1 та b2;

– введення значення довірчої ймовірності α;

– введення значень метрик X1 (кількість класів) та X2 (середня кількість методів у класі);

– відображення результатів обчислення залежної змінної Y (кількість рядків коду у тисячах);

– відображення значень границь довірчого інтервалу та інтервалу прогнозування;

– ініціація операцій збереження результатів у БД;

– ініціація експорту результатів у файл формату CSV;

– відображення повідомлень про помилки введення даних.

Рівень представлення не містить логіки обчислень або роботи з БД, а лише передає введені дані до рівня прикладної логіки та відображає отримані результати.

Рівень прикладної логіки (Application / Business Logic Layer)

Рівень прикладної логіки є центральним елементом архітектури програми та відповідає за реалізацію математичної моделі оцінювання розміру ПЗ, що створюється мовою C#.

Основні завдання рівня:

– перевірка коректності введених користувачем числових значень;

– обчислення оцінки розміру ПЗ, що створюється мовою C#, на основі регресійної моделі;

– формування об’єкта результату розрахунку для подальшого збереження у БД або експорту.

Цей рівень не залежить від конкретної реалізації інтерфейсу користувача чи БД, що дозволяє за потреби повторно використовувати його в інших типах застосунків.

Рівень доступу до даних (Data Access Layer)

Рівень доступу до даних призначений для збереження результатів обчислень у БД. Він інкапсулює всю логіку роботи з СУБД та забезпечує незалежність прикладної логіки від конкретної технології збереження даних.

Основні функції рівня:

– підключення до БД;

– збереження введених значень коефіцієнтів регресійного рівняння b0, b1 та b2 та значення довірчої ймовірності α;

– збереження введених значень метрик X1 (кількість класів) та X2 (середня кількість методів у класі) та обчисленого значення Y (кількість рядків коду у тисячах);

– зчитування збережених результатів для подальшого експорту.

У якості СУБД доцільно обрати SQLite, оскільки вона є вбудованою файловою СУБД, не потребує встановлення серверної частини та забезпечує достатню функціональність для збереження результатів розрахунків у настільному застосунку. Використання SQLite спрощує розгортання програмного засобу та підвищує його портативність.

Рівень експорту результатів (Export Layer)

Рівень експорту відповідає за формування звітів у форматі .CSV. Він реалізує перетворення даних, отриманих з рівня доступу до даних або безпосередньо з прикладної логіки, у текстовий файл зі структурою, придатною для подальшого аналізу в табличних процесорах.

Основні функції рівня:

– формування файлу формату .CSV з результатами розрахунків;

– збереження файлу у файловій системі;

– забезпечення коректного форматування числових значень.

Взаємодія компонентів системи

Взаємодія між рівнями архітектури відбувається у такій послідовності:

– користувач вводить значення метрик через графічний інтерфейс;

– дані передаються до рівня прикладної логіки для перевірки та обчислення;

– результат відображається користувачу;

– за потреби результат зберігається у БД;

– збережені дані можуть бути експортовані у файл формату .CSV.

Таке розділення забезпечує логічну цілісність системи та спрощує її подальше розширення.

3.3 Ескізний проект програми для оцінювання розміру програмного забезпечення, створеного мовою C#

При розробці ескізного проєкту програми для оцінювання розміру ПЗ, створеного мовою C#, виявлено акторів програми, сформульовано варіанти використання та розроблено специфікації варіантів використання програми, розроблено сценарії варіантів використання та прототип інтерфейсу користувача програми.

У програмі для оцінювання розміру ПЗ, створеного мовою C#, передбачається один актор: Користувач, який і буде мати повний доступ до функціональних можливостей програми.

Варіанти використання програми для оцінювання розміру ПЗ, створеного мовою C#, наведено у таблиці 3.1.

Таблиця 3.1 – Варіанти використання програми для оцінювання розміру ПЗ, створеного мовою C#

Прецедент	Короткий опис
Ввод даних про параметри розрахунку	Призначений для введення значень коефіцієнтів регресійного рівняння b_0, b_1 та b_2 і значення довірчої ймовірності α
Ввод даних про ПЗ, створене мовою C#	Призначений для введення значень метрик X_1 та X_2
Збереження вхідних даних в БД	Призначений для збереження значень вхідних даних b_0, b_1 та b_2 і α в БД
Оцінювання розміру ПЗ, створеного мовою C#	Призначений для обчислення залежної змінної Y (кількість рядків коду у тисячах) та границь довірчого інтервалу і інтервалу прогнозування
Збереження результатів оцінювання в БД	Призначений для збереження значень вихідних даних Y та границь довірчого інтервалу і інтервалу прогнозування в БД
Експорт результатів розрахунків	Призначений для експорту результатів розрахунків у файл формату .CSV

Діаграма варіантів використання програми для оцінювання розміру ПЗ, створеного мовою C#, наведена на рисунку 3.2.

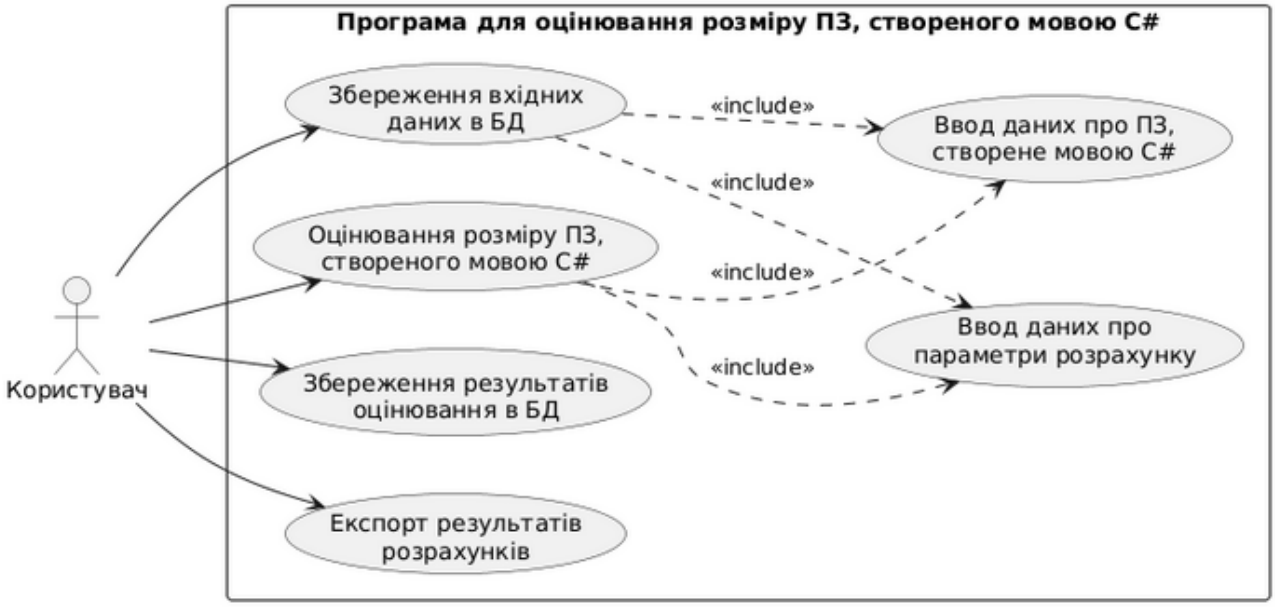

Рисунок 3.2 – Діаграма варіантів використання програми для оцінювання розміру ПЗ, створеного мовою C#

Специфікації варіантів використання програми для оцінювання розміру ПЗ, створеного мовою C#, наведені у таблицях 3.2 – 3.7.

Таблиця 3.2 – Специфікація варіанту використання "Ввод даних про параметри розрахунку"

Складова	Опис
Опис прецеденту	Даний варіант використання ініціює користувач. Призначений для введення значень коефіцієнтів регресійного рівняння b_0, b_1 та b_2 і значення довірчої ймовірності α
Передумови	Відсутні
Базовий потік	Користувач вводить значення коефіцієнтів регресійного рівняння b_0, b_1 та b_2 і значення довірчої ймовірності α у відповідні поля форм згідно з допустимим типом даних: числовий
Альтернативні потоки	У разі введення значення недопустимого формату, програма видає відповідне попередження та дає можливість повторити ввод.
Постумови	Відсутні

Таблиця 3.3 – Специфікація варіанту використання "Ввод даних про ПЗ, створене мовою C#"

Складова	Опис
Опис прецеденту	Даний варіант використання ініціює користувач. Призначений для введення значень метрик X_1 (кількість класів) та X_2 (середня кількість методів у класі)
Передумови	Відсутні
Базовий потік	Користувач вводить значення метрик X_1 (кількість класів) та X_2 (середня кількість методів у класі) у відповідні поля форм згідно з допустимим типом даних: числовий
Альтернативні потоки	У разі введення значення недопустимого формату, програма видає відповідне попередження та дає можливість повторити ввод.
Постумови	Відсутні

Таблиця 3.4 – Специфікація варіанту використання "Збереження вхідних даних в БД"

Складова	Опис
Опис прецеденту	Даний варіант використання ініціює користувач. Призначений для збереження вхідних даних: коефіцієнтів регресійного рівняння b_0, b_1 та b_2 і значення довірчої ймовірності α, а також метрик X_1 (кількість класів) та X_2 (середня кількість методів у класі) в БД програми
Передумови	Перед виконанням базового потоку варіанта використання повинні бути виконані потоки варіантів використання "Ввод даних про параметри розрахунку" та "Ввод даних про ПЗ, створене мовою C#"
Базовий потік	Користувач обирає варіант використання, програма виконує збереження вхідних даних в БД програми
Альтернативні потоки	Вхідні дані не введені. Програма пропонує виконати потоки варіантів використання "Ввод даних про параметри розрахунку" та "Ввод даних про ПЗ, створене мовою C#" повторно
Постумови	Відсутні

Таблиця 3.5 – Специфікація варіанту використання "Оцінювання розміру ПЗ, створеного мовою C#"

Складова	Опис
Опис прецеденту	Даний варіант використання ініціює користувач. Програма відображає результат обчислення залежної змінної Y (кількість рядків коду у тисячах) та границь довірчого інтервалу і інтервалу прогнозування у відповідних полях форм
Передумови	Перед виконанням базового потоку варіанта використання повинні бути виконані потоки варіантів використання "Ввод даних про параметри розрахунку" та "Ввод даних про ПЗ, створене мовою C#"
Базовий потік	Користувач обирає варіант використання, програма виконує оцінювання розміру ПЗ, створеного мовою C#, згідно вхідних даних
Альтернативні потоки	Відсутні
Постумови	Відсутні

Таблиця 3.6 – Специфікація варіанту використання "Збереження результатів оцінювання в БД"

Складова	Опис
Опис прецеденту	Даний варіант використання ініціює користувач. Призначений для збереження результатів оцінювання: залежної змінної *Y* (кількість рядків коду у тисячах) та границь довірчого інтервалу і інтервалу прогнозування в БД програми
Передумови	Перед виконанням базового потоку варіанта використання повинен бути виконаний потік варіанта використання «Оцінювання розміру ПЗ, створеного мовою C#»
Базовий потік	Користувач обирає варіант використання, програма виконує збереження результатів оцінювання в БД програми
Альтернативні потоки	Результати оцінювання не розраховані. Програма пропонує виконати поток варіанту використання "Оцінювання розміру ПЗ, створеного мовою C#" повторно
Постумови	Відсутні

Таблиця 3.7 – Специфікація варіанту використання "Експорт результатів розрахунків"

Складова	Опис
Опис прецеденту	Даний варіант використання ініціює користувач. Призначений для експорту результатів розрахунків у файл формату .CSV
Передумови	Перед виконанням базового потоку варіанта використання повинен бути виконаний потік варіанта використання «Оцінювання розміру ПЗ, створеного мовою C#»
Базовий потік	Користувач обирає варіант використання, програма виконує експорт результатів розрахунків у файл формату .CSV
Альтернативні потоки	Результати оцінювання не розраховані. Програма пропонує виконати поток варіанту використання "Оцінювання розміру ПЗ, створеного мовою C#" повторно
Постумови	Відсутні

Діаграма діяльності програми для оцінювання розміру ПЗ, створеного мовою C#, наведена на рисунку 3.3.

Рисунок 3.3 – Діаграма діяльності програми для оцінювання розміру ПЗ, створеного мовою C#

Далі розробимо прототип інтерфейсу користувача програми для оцінювання розміру ПЗ, створеного мовою C#, який наведено на рисунку 3.4.

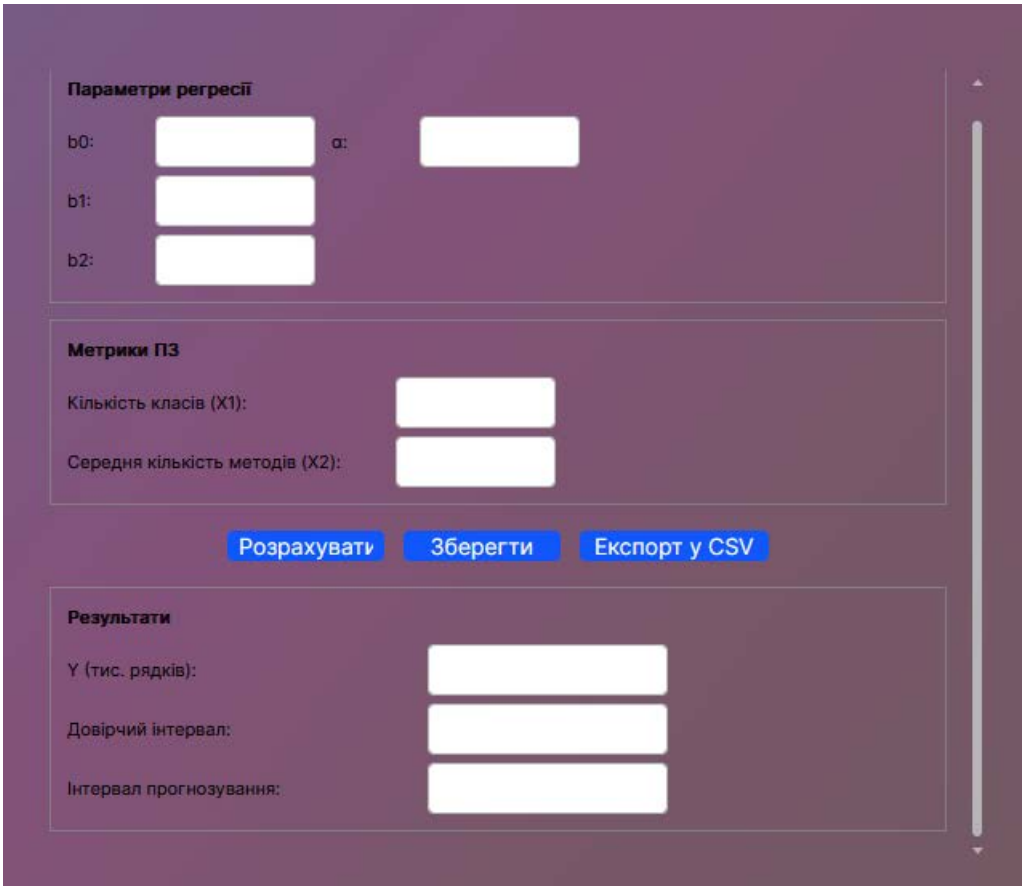

Рисунок 3.4 – Прототип інтерфейсу користувача програми для оцінювання розміру ПЗ, створеного мовою C#

Інтерфейс користувача програми для оцінювання розміру ПЗ, створеного мовою C# виконано у вигляді одного головного вікна. Мова інтерфейсу – українська. Основною метою інтерфейсу користувача є забезпечення зручного введення вхідних даних, відображення результатів обчислень та ініціації операцій збереження й експорту результатів.

Програма буде реалізована з використанням WPF у середовищі C#, з організацією елементів керування у логічні групи.

Інтерфейс складається з наступних блоків:

– блок «Параметри регресії» призначений для введення значень коефіцієнтів регресійного рівняння та довірчої ймовірності, кожен параметр

вводиться у відповідне текстове поле. Заголовок блоку відображає його призначення та виділений жирним шрифтом.

– блок «Метрики ПЗ» призначений для введення значень метрик ПЗ, кожне поле має підпис, що пояснює зміст значення, та забезпечує однозначне введення даних користувачем.

– блок кнопок дій містить три кнопки для керування процесом роботи програми:

«Розрахувати» – ініціює обчислення залежної змінної Y (кількість рядків коду у тисячах) та довірчих інтервалів;

«Зберегти» – виконує збереження введених параметрів та результатів обчислень у БД;

«Експорт у CSV» – здійснює формування та збереження звітного файлу у форматі CSV.

Кнопки розташовані горизонтально та вирівняні по центру вікна для зручності користувача.

– блок «Результати» відображає результати обчислень, отриманих після натискання кнопки «Розрахувати», всі поля результатів є тільки для читання, що запобігає випадковій зміні даних користувачем.

Додаткові елементи інтерфейсу користувача

Усі блоки виділені рамками для візуального розмежування логічних частин інтерфейсу. Прокрутка сторінки реалізована через ScrollViewer, що забезпечує доступ до всіх елементів інтерфейсу на екранах з різною роздільною здатністю. Інтерфейс підтримує перевірку коректності введених даних і відображення повідомлень про помилки у відповідних текстових полях або повідомленнях (реалізація логіки перевірки передбачена у коді C#).

Таким чином, спроєктований інтерфейс користувача програми для оцінювання розміру ПЗ, створеного мовою C#, є інтуїтивно зрозумілим, логічно структурованим та функціонально повним, що дозволяє користувачу:

– швидко вводити необхідні дані для розрахунку розміру ПЗ;

– отримувати результати у наочному вигляді;

– виконувати операції збереження та експорту результатів.

3.4 Технічний проект програми для оцінювання розміру програмного забезпечення, створеного мовою C#

Для початку роботи з технічним проєктом розробимо діаграму класів програми для оцінювання розміру ПЗ, створеного мовою C#, яка відповідає необхідній функціональності та її архітектурі (Presentation Layer + Data Access Layer).

Діаграма класів відображає структуру програми для оцінювання розміру ПЗ, створеного мовою C#, а також взаємозв’язки між основними класами, що реалізують функціональність системи. Архітектура програми побудована з урахуванням розподілу відповідальностей між рівнем представлення, рівнем бізнес-логіки та рівнем доступу до даних.

Ця діаграма класів свідомо не перевантажена технічними деталями WPF, та відображає логіку обчислень, роботу з БД, експорт у CSV, розподіл відповідальностей між класами. Діаграма класів програми для оцінювання розміру ПЗ, створеного мовою C#, наведена на рисунку 3.5.

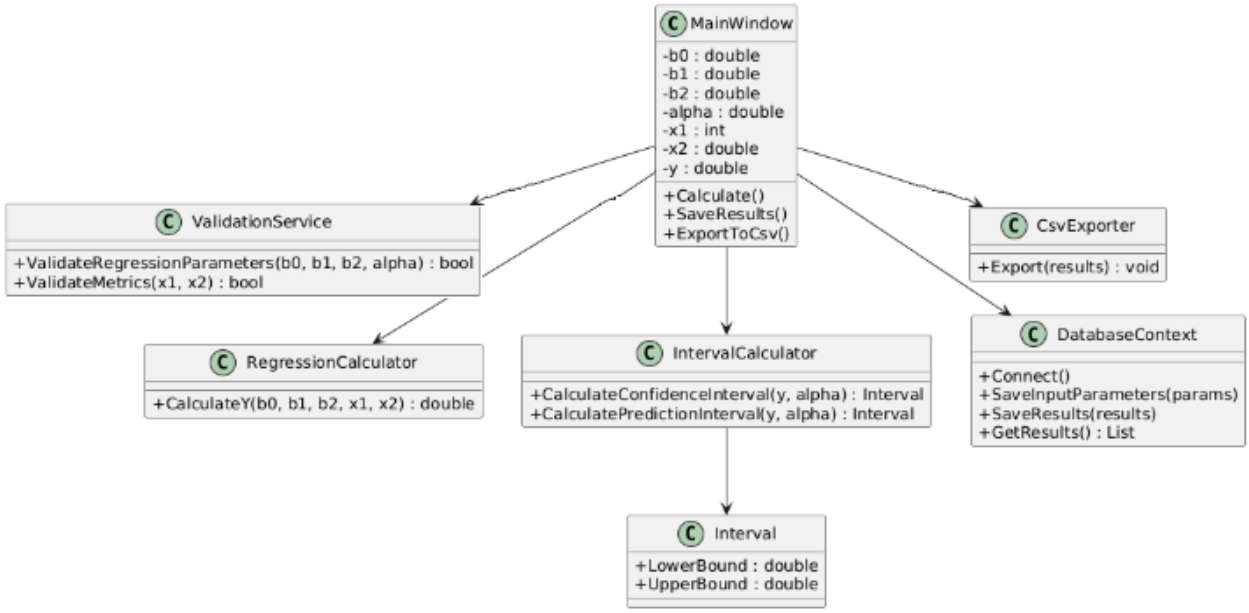

Рисунок 3.5 – Діаграма класів програми для оцінювання розміру ПЗ, створеного мовою C#

Специфікація класів програми для оцінювання розміру ПЗ, створеного мовою C#

1. Класи рівня представлення (Presentation Layer)

Клас MainWindow

Клас MainWindow є головним класом користувацького інтерфейсу та відповідає за взаємодію з користувачем. У ньому зберігаються введені значення коефіцієнтів регресійного рівняння (b_0, b_1, b_2), довірчої ймовірності (α), значення метрик ПЗ (X_1, X_2), а також обчислене значення залежної змінної Y.

Основні методи класу:

– Calculate() – ініціює процес обчислення розміру ПЗ та інтервальних оцінок;

– SaveResults() – виконує збереження введених даних та результатів обчислень у БД;

– ExportToCsv() – ініціює експорт результатів розрахунків у файл формату CSV.

Клас MainWindow використовує сервіси перевірки даних, обчислення та доступу до даних, що забезпечує модульність та спрощує супровід програми.

Клас ValidationService

Клас ValidationService призначений для перевірки коректності введених користувачем даних. Він реалізує методи перевірки:

– значень коефіцієнтів регресійного рівняння та довірчої ймовірності;

– значень метрик ПЗ.

Використання окремого класу для валідації дозволяє відокремити логіку перевірки від інтерфейсу користувача та підвищує надійність програми.

2. Класи рівня бізнес-логіки (Business Logic Layer)

Клас RegressionCalculator

Клас RegressionCalculator відповідає за реалізацію регресійної моделі оцінювання розміру ПЗ. Основним його методом є:

– CalculateY(b_0, b_1, b_2, x_1, x_2) – обчислення значення залежної змінної Y на основі введених коефіцієнтів та метрик.

Таким чином, у даному класі зосереджена основна математична логіка програми.

Клас IntervalCalculator

Клас IntervalCalculator реалізує обчислення для отриманого значення Y з урахуванням заданої довірчої ймовірності α:

– довірчого інтервалу рівняння регресії;

– інтервалу прогнозування рівняння регресії.

Результати обчислень повертаються у вигляді об’єкта класу Interval.

Клас Interval

Клас Interval є допоміжним і використовується для зберігання меж інтервалів. Він містить два атрибути:

– LowerBound – нижня межа інтервалу;

– UpperBound – верхня межа інтервалу.

Даний клас забезпечує зручне представлення інтервальних оцінок у програмі.

3. Класи рівня доступу до даних (Data Access Layer)

Клас DatabaseContext

Клас DatabaseContext інкапсулює логіку роботи з БД. Він забезпечує:

– встановлення з’єднання з БД;

– збереження введених параметрів регресійної моделі;

– збереження результатів оцінювання розміру ПЗ;

– зчитування збережених даних для подальшого експорту.

Використання окремого класу доступу до даних дозволяє ізолювати роботу з БД від інших компонентів програми.

Клас CsvExporter

Клас CsvExporter відповідає за формування звітів у форматі .CSV. Він отримує збережені результати обчислень та виконує їх експорт у файл, придатний для подальшого аналізу або обробки в сторонніх програмних засобах.

Взаємодія класів

Клас MainWindow є центральним елементом взаємодії та використовує:

– ValidationService для перевірки коректності введених даних;

– RegressionCalculator для обчислення значення Y;

– IntervalCalculator для визначення довірчого інтервалу та інтервалу прогнозування рівняння регресії;

– DatabaseContext для збереження та отримання даних з БД;

– CsvExporter для експорту результатів у CSV-файл.

Такий підхід забезпечує логічну структурованість, масштабованість та зручність супроводу ПЗ.

Далі розробимо діаграму послідовності програми для оцінювання розміру ПЗ, створеного мовою C#, яка логічно відображає роботу програми та узгоджується з уже створеною діаграмою класів і архітектурою. Діаграма описує типовий сценарій: введення даних – розрахунок – збереження – експорт. Діаграма послідовності програми для оцінювання розміру ПЗ, створеного мовою C#, наведена на рисунку 3.6.

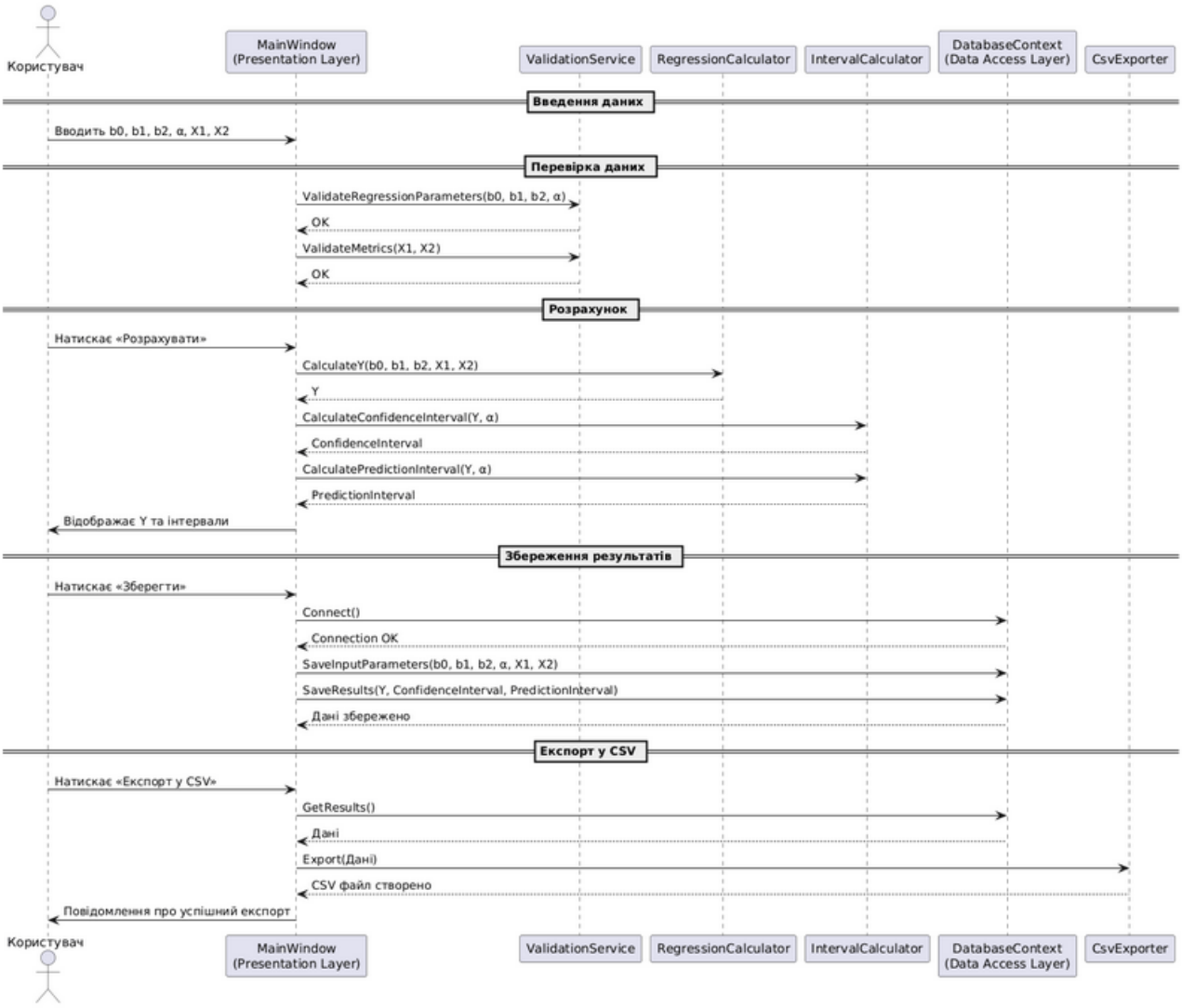

Рисунок 3.6 – Діаграма послідовності програми для оцінювання розміру ПЗ, створеного мовою C#

Діаграма послідовності демонструє потік управління та обміну даними між компонентами програми, що підтверджує правильність обраної архітектури та логіки роботи програми. Вона ілюструє типову взаємодію користувача з програмою під час оцінювання розміру ПЗ, створеного мовою C#, включаючи введення даних, виконання розрахунків, збереження та експорт результатів.

Актор Користувач ініціює всі дії у програмі. MainWindow координує взаємодію між компонентами. ValidationService забезпечує коректність введених даних. RegressionCalculator виконує обчислення залежної змінної Y. IntervalCalculator визначає довірчий інтервал та інтервал прогнозування. DatabaseContext відповідає за збереження та зчитування даних. CsvExporter формує вихідний файл у форматі .CSV.

3.5 Робочий проект програми для оцінювання розміру програмного забезпечення, створеного мовою C#

Під час виконання робочого проєкту було виконано: обґрунтування вибору мови програмування та стеку розробки, кодування, тестування, випробування програми для оцінювання розміру ПЗ, створеного мовою C#, розробка усієї необхідної програмної документації.

Обґрунтування вибору мови програмування та стеку розробки

Для реалізації програми для оцінювання розміру ПЗ, створеного мовою C#, було обрано мову програмування C#, що є однією з основних мов платформи .NET [1]. Даний вибір зумовлений низкою технічних та методологічних переваг.

По-перше, мова C# забезпечує високий рівень абстракції, що дозволяє ефективно реалізувати об’єктно-орієнтований підхід до розробки ПЗ. Це є важливим для побудови чіткої архітектури програми, реалізації діаграми класів та подальшого супроводу системи.

По-друге, C# має потужну стандартну бібліотеку, яка містить засоби для виконання математичних та статистичних обчислень, роботи з файлами та потоками даних, взаємодії з реляційними БД, реалізації графічного інтерфейсу користувача. Це дозволяє реалізувати всі функціональні вимоги до програми без використання сторонніх або спеціалізованих бібліотек.

По-третє, мова C# є типобезпечною, що зменшує ймовірність помилок під час обчислень та обробки даних, а також підвищує надійність програмного продукту.

Платформа .NET була обрана як середовище виконання програми, оскільки вона забезпечує високу продуктивність виконання коду, ефективне управління пам’яттю за рахунок автоматичного збирання сміття, зручні механізми обробки винятків, кросверсійну стабільність та довготривалу підтримку. Платформа .NET широко використовується для розробки настільних застосунків у середовищі Windows, що повністю відповідає вимогам до даного програмного продукту.

Для реалізації графічного інтерфейсу користувача обрано WPF через такі основні причини:

– підтримка декларативного опису інтерфейсу за допомогою мови XAML;

– можливість чіткого розділення логіки програми та інтерфейсу користувача;

– підтримка шаблонів проєктування, зокрема MVVM;

– зручне масштабування інтерфейсу під різні роздільні здатності екранів.

WPF є сучасною технологією розробки віконних застосунків та дозволяє створювати інтуїтивно зрозумілий та логічно структурований інтерфейс користувача.

Для збереження результатів розрахунків використовується SQLite – вбудована реляційна система керування базами даних, яка була обрана через таке обґрунтування:

– відсутність необхідності встановлення окремого сервера БД;

– зберігання всієї БД в одному файлі;

– простота розгортання та використання;

– достатня продуктивність для зберігання результатів розрахунків.

Оскільки програма не передбачає багатокористувацький режим роботи або складні транзакційні операції, використання SQLite є технічно та економічно доцільним. Для роботи з БД у програмі використовується стандартний механізм доступу до даних платформи .NET, що забезпечує надійне підключення до БД, виконання SQL-запитів, збереження та зчитування результатів розрахунків.

Для експорту результатів обчислень обрано формат .CSV, оскільки він є простим у реалізації, підтримується більшістю табличних та аналітичних програм, забезпечує зручний подальший аналіз результатів. Робота з CSV-файлами реалізується стандартними засобами введення-виведення мови C#.

Для розробки програми використовується середовище розробки Microsoft Visual Studio, яке забезпечує повну підтримку мови C# та платформи .NET, зручні засоби проєктування інтерфейсу, інтегровані інструменти налагодження та тестування, підтримку UML-проєктування та рефакторингу коду.

Таким чином, вибір мови програмування C# та всього стеку розробки є обґрунтованим з точки зору функціональних вимог до програми, простоти реалізації, надійності та масштабованості, відповідності сучасним підходам до розробки ПЗ. Обраний стек розробки дозволяє створити ефективну, зручну у

використанні та легко супроводжувану програму для оцінювання розміру ПЗ, створеного мовою C#. Код програми наведений у додатку Б. Опис програми для оцінювання розміру ПЗ, що створюється мовою C#, наведений у додатку В. Інструкція користувача програми для оцінювання розміру ПЗ, що створюється мовою C#, наведена у додатку Г.

Тестування розробленої програми для оцінювання розміру ПЗ, створеного мовою C#, було проведено методом еквівалентного розбиття. Для перевірки було використано, в тому числі, дані з таблиці 2.1.

При тестуванні використовувалися такі типи класів еквівалентності:

– коректні значення: значення, які є припустимими для системи згідно з її специфікацією;

– некоректні значення: значення, які повинні бути відхилені системою.

Були протестовані всі функції програми. Наведено приклад тестування функції «Ввод даних про параметри розрахунку». Класи еквівалентності вхідних даних для функції «Ввод даних про параметри розрахунку» представлені в таблиці 3.8.

Таблиця 3.8 – Класи еквівалентності вхідних даних для функції «Ввод даних про параметри розрахунку»

Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
Коефіцієнти регресійного рівняння b_0, b_1 та b_2 і значення довірчої ймовірності α	1 Всі коефіцієнти регресійного рівняння і значення довірчої ймовірності задані вірно	2 Одне чи більше значень задані невірно
		3 Відсутність одного чи декількох значень

В таблиці 3.9 наведено тести з правильних класів еквівалентності для функції «Ввод даних про параметри розрахунку».

В таблиці 3.10 наведено тести з неправильних класів еквівалентності для функції «Ввод даних про параметри розрахунку».

Таблиця 3.9 – Тести з правильних класів еквівалентності для функції «Ввод даних про параметри розрахунку»

№ покритого класу еквівалентності	Вхідні дані	Результат
1	Всі коефіцієнти регресійного рівняння і значення довірчої ймовірності задані вірно	Результат вірний

Таблиця 3.10 – Тести з неправильних класів еквівалентності для функції «Ввод даних про параметри розрахунку»

№ покритого класу еквівалентності	Вхідні дані	Результат
2	Одне чи більше значень задані невірно	Програма виводить повідомлення про помилку. Результат підтверджено
3	Відсутність одного чи декількох значень	Програма виводить повідомлення про помилку. Результат підтверджено

По результатам тестування можна зробити висновок про те, що програма для оцінювання розміру ПЗ, створеного мовою C#, працює згідно вимог, сформульованих у технічному завданні.

Зовнішній вигляд основного вікна програми для оцінювання розміру ПЗ, створеного мовою C#, представлено на рисунку 3.7, розрахунок оцінювання розміру ПЗ, створеного мовою C#, представлено на рисунку 3.8.

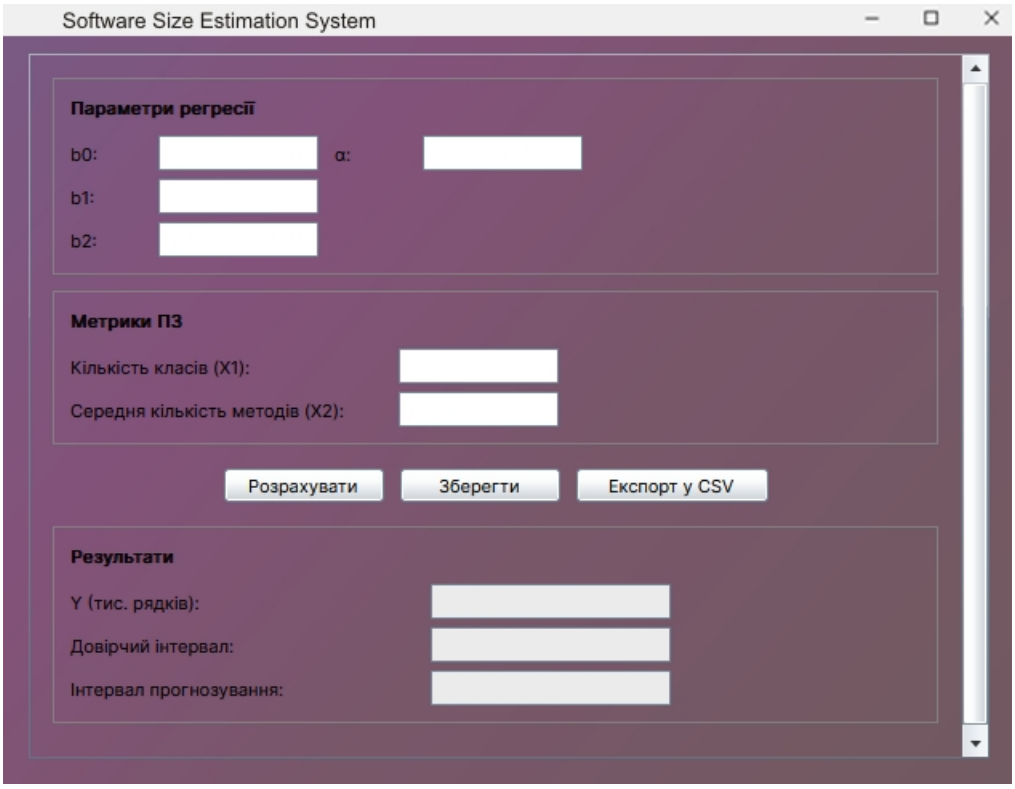

Рисунок 3.7 – Зовнішній вигляд основного вікна програми для оцінювання розміру ПЗ, створеного мовою C#

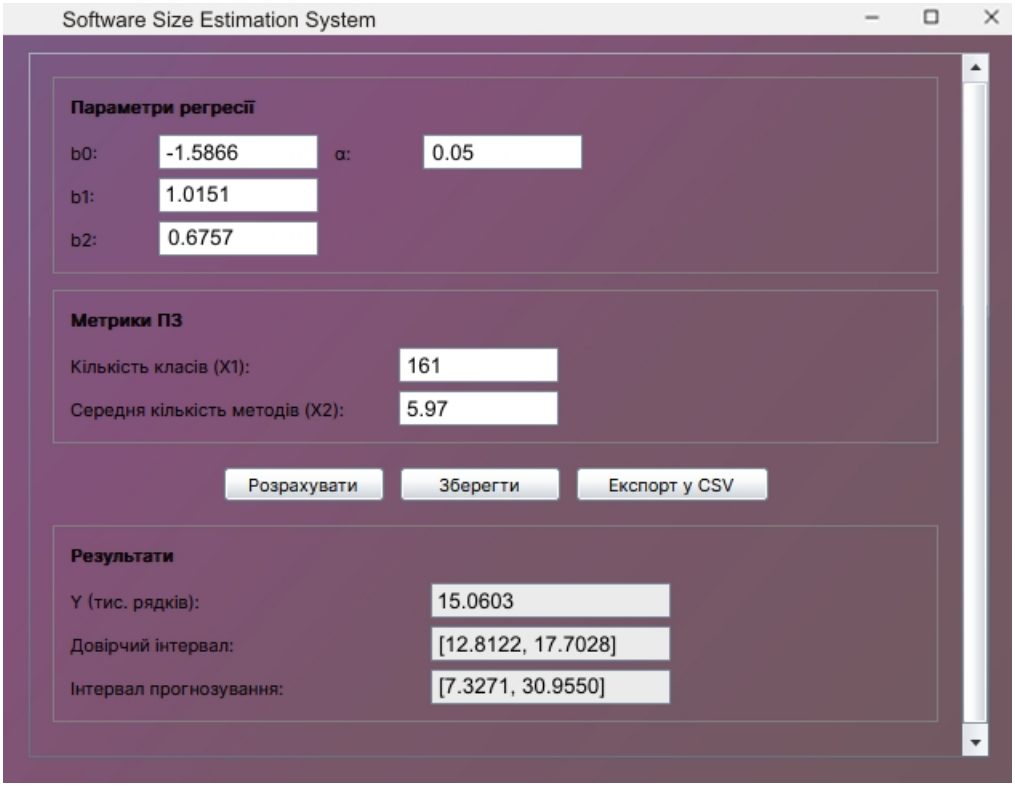

Рисунок 3.8 – Розрахунок оцінювання розміру ПЗ, створеного мовою C#

Випробування програми для оцінювання розміру ПЗ, створеного мовою C#, виконано у відповідності з Програмою та методикою випробувань, яка представлена у Додатку Д.

4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВІД РОЗРОБКИ І ВПРОВАДЖЕННЯ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЩО СТВОРЮЄТЬСЯ МОВОЮ C#

Дана кваліфікаційна робота присвячена розробці програми для оцінювання розміру програмного забезпечення, створеного мовою C#.

Розробка програмного забезпечення потребує разових витрат, пов’язаних із його створенням, придбанням необхідних технічних засобів, а також поточних витрат, що виникають у процесі його експлуатації. Економічний ефект від використання програмного забезпечення визначається з урахуванням витрат на його функціонування. Співвідношення отриманої економії до витрат на розробку програмного забезпечення є показником економічної ефективності здійснених капіталовкладень. Розрахунок економічних показників здійснюється на основі чинних на момент обчислення оптових цін, тарифів і ставок оплати праці.

У даному розділі розраховується собівартість розробки програмного забезпечення, а також розрахунки економічної ефективності та строку окупності.

4.1 Розрахунок витрат на створення й експлуатацію програми для оцінювання розміру програмного забезпечення, створеного мовою C#

Витрати на розробку програмного забезпечення складаються з витрат на заробітну плату, експлуатацію комп`ютера, на якому виконується розробка, амортизацію комп`ютера, додаткові засоби розробки, матеріали і комплектуючі.

Розробка програмного забезпечення виконується програмістом, місячна заробітна плата якого складає 27500 грн. Вартість комп'ютера складає 37500 грн. Витрати на допоміжні матеріали наведені в таблиці 4.1.

Таблиця 4.1 – Витрати на допоміжні матеріали

№	Пункти витрат	Сума, грн.
1	Доступ до мережі Internet	250
2	Папір А4	200
3	Заправка картриджа до принтера	280
4	Непередбачувані витрати	350
	Разом	1080

Вартість розробки програмного забезпечення розраховується за формулою:

$$К_{пр} = (В_{зп} + В_{есв} + В_{зг} + В_{е}) * Т + В_{дм}, \quad (4.1)$$

де: $В_{зп}$ – витрати на оплату праці, встановлені зарплатнею працівника, грн.,

$В_{есв}$ – єдиний соціальний внесок, що складає 38% від витрат на заробітну плату, грн.,

$В_{зг}$ – загальногосподарські витрати, що складають 10% від витрат на оплату праці без урахування податкового навантаження, грн.;

$В_{е}$ – витрати за електроенергію, грн.;

Т – тривалість розробки, міс.;

$В_{дм}$ – витрати на допоміжні матеріали, грн.

Розрахунок вартості розробки програмного забезпечення виконується при врахуванні наступної вартості кіловат-години електроенергії: 4,32 грн. За умови споживання потужності 0,75 кВт та тривалості розробки на місяць, а саме 22×8=176 годин розрахуємо значення $З_е$:

$$В_е = 176 \cdot 0{,}75 \cdot 4{,}32 = 570{,}24 \text{ грн.}$$

Витрати на розробку програмного забезпечення наведені в таблиці 4.2.

Таблиця 4.2 – Витрати на розробку програмного забезпечення

№	Найменування витрат	Одиниця	Кількість
1	Тривалість розробки	міс.	1
2	Основна і додаткова заробітна плата	грн.	33000
3	Відрахування в соціальні фонди	грн.	12540
4	Загальногосподарські витрати	грн.	3300
5	Витрати на допоміжні матеріали	грн.	1080
6	Витрати на електроенергію	грн.	570,24

Оскільки за методом експертної оцінки на основі розробки аналогічних систем визначено, що загальна тривалість розробки Т складає 1 місяць, а кількість необхідних спеціалістів дорівнює 1 людині, то вартість розробки програмного забезпечення розрахуємо за формулою (4.1) і вона складає:

$$К_{пр} = (33000 + 12540 + 3300 + 570{,}24) * 1 + 1080 = 50490{,}24 \text{ грн.}$$

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

$$А_{уст} = 37500 \cdot 0,6 = 22500 \text{ грн.}$$

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

$$В_{м} = 37500 \cdot 0,05 = 1875 \text{ грн.}$$

Оскільки в середньому устаткування споживає 0,1 кВт електроенергії і навантажується близько 6,5 годин на день, а в календарному році маємо 256 робочих днів, можемо розрахувати річне споживання електроенергії устаткуванням:

$$В_{е} = 256 \cdot 0,1 \cdot 6,5 \cdot 4,32 = 718,85 \text{ грн.}$$

Експлуатаційні витрати на устаткування на рік складуть:

$$В_{уст} = 22500 + 1875 + 718,85 = 25093,85 \text{ грн.}$$

Отже, у перший рік витрати на створення й експлуатацію програмного продукту складуть:

$$В_{р} = 50490,24 + 25093,85 = 75584,09 \text{ грн.}$$

4.2 Економічна ефективність розробки та впровадження програми для оцінюваня розміру програмного забезпечення, створеного мовою C#

Головним показником економічної ефективності програмного забезпечення є зростання результативності управління інформацією, що проявляється у скороченні витрат на управлінські процеси за одночасного підвищення швидкості та якості отримання необхідних результатів.

Ручна обробка даних – основний негативний фактор, якого планується позбутися від впровадження програмного забезпечення, що розробляється – окрім інших недоліків, призводить до низки негативних економічних наслідків, зокрема значних витрат на зберігання паперової документації (сейфи, шафи, папки тощо), збільшення витрат на канцелярські матеріали, а також додаткових витрат, пов’язаних із виявленням і виправленням раніше допущених помилок.

Серед ключових чинників, що зумовлюють зростання прибутку внаслідок впровадження програмного забезпечення, можна виокремити підвищення продуктивності праці та вивільнення робочого часу працівників. Водночас низка позитивних ефектів, таких як зростання оперативності управління, покращення якості отриманих результатів і вдосконалення організації праці, не піддаються безпосередній грошовій оцінці.

Необхідною умовою визначення економічної ефективності програмного забезпечення є забезпечення порівнянності всіх показників у часі, за рівнем цін та іншими нормативами, а також єдність підходів до їх розрахунку за змістом і складом витрат.

Визначення прямої економічної ефективності ґрунтується на тому, що впровадження програмного забезпечення за експертною оцінкою фахівців підприємства дозволить вивільнити 1 працівника.

Зарплата 1 працівника в рік складає:

$$(25000{,}00*12)*0{,}6 = 330000{,}00 \text{ грн.}$$

Річний економічний ефект розраховується за наступною формулою:

$$\mathrm{E}_{\text{год}} = \Delta\mathrm{C}_n - E_n * k, \quad (4.2)$$

де $\Delta\mathrm{C}_n$– вивільнені кошти після впровадження програмного забезпечення (330000,00 грн) мінус експлуатаційні витрати (25093,85) = 304906,15 грн.;

E_n– коефіцієнт ефективності, який дорівнює коефіцієнту амортизації та має значення 0,6;

k – одноразові витрати на впровадження продукту (50490,24).

$$\mathrm{E}_{\text{год}} = 304906{,}15 - 0{,}6 * 50490{,}24 = 274612{,}01 \text{ грн.}$$

Строк окупності програмного забезпечення розраховується по формулі:

$$\mathrm{T} = k/\Delta\mathrm{C}_n \text{ року.} \quad (4.3)$$

$$\mathrm{T} = 50490{,}24/304906{,}15 \approx 0{,}1656 \text{ року.}$$

Отже, строк окупності програмного забезпечення, що розробляється, становить приблизно 0,1656 року, або приблизно 1,99 місяців, що можливо округлити до показника в 2 місяці.

5 ОХОРОНА ПРАЦІ

Охорона праці є одним із ключових соціально-економічних напрямів розвитку держави та важливим елементом забезпечення безпеки трудової діяльності. Її основне завдання полягає у формуванні та впровадженні комплексу технічних, економічних, правових і санітарно-гігієнічних заходів, спрямованих на створення безпечних і належних умов праці. У процесі виконання трудових обов’язків людина постійно взаємодіє з виробничим середовищем, яке включає соціальні чинники, а також елементи технічного й природного походження.

Питання охорони праці нерозривно пов’язані з пожежною безпекою та станом повітряного середовища в приміщеннях різного призначення, рівнем освітлення, ефективністю вентиляційних систем, впливом вібрації, шуму, електромагнітних і радіоактивних випромінювань. Основною метою охорони праці є дослідження та організація трудового процесу з позицій збереження життя і здоров’я працівників, а також зменшення ризику виникнення нещасних випадків до мінімального рівня. Усі вимоги та норми у сфері охорони праці закріплені в Конституції України та Законі України «Про охорону праці» [30] і мають обов’язкову законодавчу силу.

5.1 Загальні вимоги з охорони праці при роботі на комп'ютері

Робота з комп’ютерною технікою на сьогодні є невід’ємною складовою професійної діяльності фахівців у галузі інформаційних технологій, управління, освіти та інших сфер. Разом із тим, тривале використання персональних комп’ютерів пов’язане з впливом низки потенційно небезпечних і шкідливих виробничих факторів, що зумовлює необхідність дотримання загальних вимог з

охорони праці. Метою таких вимог є збереження здоров’я працівників, профілактика професійних захворювань та зниження ризику виникнення нещасних випадків у процесі трудової діяльності.

Правові основи охорони праці при роботі на комп’ютері визначаються Конституцією України, Законом України «Про охорону праці», Кодексом законів про працю України, а також спеціальними нормативно-правовими актами, зокрема, Вимогами щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями [31]. Вказані документи встановлюють обов’язкові вимоги до організації робочих місць, умов праці, режимів роботи та відпочинку, а також до безпеки обладнання.

Однією з основних вимог з охорони праці є правильна організація робочого місця користувача комп’ютера. Робоче місце повинно відповідати ергономічним вимогам і забезпечувати зручне та безпечне розташування обладнання. Стіл має бути достатньої площі для розміщення монітора, клавіатури, маніпулятора типу «миша» та іншого допоміжного обладнання. Висота робочої поверхні повинна відповідати зросту працівника, а робоче крісло – бути регульованим за висотою, кутом нахилу спинки та мати опору для попереку.

Монітор комп’ютера необхідно розміщувати на відстані 50–70 см від очей користувача, при цьому верхня частина екрана має знаходитися на рівні очей або трохи нижче. Таке розташування дозволяє зменшити напруження зору та навантаження на шийний відділ хребта. Клавіатура повинна бути розміщена на відстані 10–30 см від краю столу, що забезпечує зручне положення кистей рук і запобігає розвитку захворювань опорно-рухового апарату.

Важливою складовою охорони праці при роботі на комп’ютері є забезпечення оптимальних параметрів мікроклімату в робочому приміщенні. Температура повітря, відносна вологість і швидкість руху повітря повинні відповідати встановленим санітарним нормам. Для приміщень із постійною роботою за комп’ютером рекомендована температура повітря в межах 22-24°С у

холодний період року та 23–25°C у теплий період, а відносна вологість – 40-60% [30]. Недотримання цих вимог може призвести до зниження працездатності та погіршення самопочуття працівників.

Особливу увагу слід приділяти освітленню робочих місць. Природне та штучне освітлення повинні забезпечувати достатній рівень освітленості без появи відблисків на екрані монітора. Світловий потік має падати зліва або зліва-ззаду від користувача, а джерела світла не повинні знаходитися в полі прямого зору. Неправильне освітлення сприяє швидкій втомі очей, головному болю та зниженню ефективності праці.

При роботі з комп’ютером працівник зазнає впливу електромагнітного випромінювання, шуму та вібрації, рівні яких, хоча й зазвичай не перевищують допустимих норм, повинні контролюватися. Сучасні комп’ютерні пристрої відповідають міжнародним стандартам безпеки, однак важливо використовувати сертифіковане обладнання та дотримуватися правил його експлуатації. Забороняється використовувати несправну техніку, працювати з пошкодженими кабелями живлення або самостійно виконувати ремонтні роботи без відповідної кваліфікації.

Значне місце у системі охорони праці займає раціональний режим праці та відпочинку. Тривала безперервна робота за комп’ютером може спричиняти перенапруження зору, нервової системи та опорно-рухового апарату. Відповідно до санітарних норм, для працівників, які постійно працюють за комп’ютером, передбачаються регламентовані перерви протягом робочого дня. Під час перерв рекомендується виконувати вправи для очей, розминку для м’язів спини, шиї та рук, а також змінювати вид діяльності.

Не менш важливою є пожежна безпека при роботі з комп’ютерною технікою. Робочі приміщення повинні бути обладнані справними електромережами, автоматичними вимикачами та засобами пожежогасіння. Забороняється перевантажувати електричні розетки, використовувати саморобні

подовжувачі та залишати увімкнену техніку без нагляду. Дотримання вимог пожежної безпеки значно знижує ризик виникнення аварійних ситуацій.

Отже, загальні вимоги з охорони праці при роботі на комп’ютері охоплюють комплекс організаційних, технічних, санітарно-гігієнічних і правових заходів. Їх дотримання є необхідною умовою забезпечення безпечних і комфортних умов праці, збереження здоров’я працівників та підвищення ефективності професійної діяльності.

5.2 Розробка заходів щодо зменшення впливу шкідливих та небезпечних факторів при роботі на комп'ютері

Робота з комп’ютерною технікою супроводжується впливом комплексу шкідливих і небезпечних факторів, які за тривалої або неправильно організованої праці можуть негативно позначатися на стані здоров’я працівників. До основних таких факторів належать підвищене навантаження на зорову та нервову системи, статичні навантаження на опорно-руховий апарат, несприятливі параметри мікроклімату, недоліки освітлення, вплив електромагнітних випромінювань, шуму, а також ризики, пов’язані з електро- та пожежною безпекою. У зв’язку з цим виникає необхідність розробки та впровадження комплексу заходів, спрямованих на мінімізацію негативного впливу зазначених факторів та створення безпечних умов праці.

Заходи щодо зниження зорового навантаження

Одним із найбільш поширених шкідливих факторів при роботі на комп’ютері є перенапруження зору. Для його зменшення необхідно забезпечити правильний вибір і розташування монітора. Рекомендується використовувати сучасні рідкокристалічні монітори з високою роздільною здатністю, оптимальною частотою оновлення та антивідблисковим покриттям. Яскравість і

контрастність екрана повинні регулюватися відповідно до умов освітлення приміщення та індивідуальних особливостей користувача.

Важливим заходом є дотримання нормативної відстані між очима користувача та екраном монітора (50–70 см), а також правильне розміщення монітора по висоті. Для зниження втоми очей рекомендується застосовувати регламентовані перерви в роботі, під час яких виконуються спеціальні вправи для очей, спрямовані на розслаблення акомодаційних м’язів. Зазначені вимоги та рекомендації визначені у Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями [31].

Заходи щодо зменшення навантаження на опорно-руховий апарат

Тривала робота в сидячому положенні спричиняє статичні навантаження на м’язи спини, шиї та верхніх кінцівок, що може призводити до розвитку професійних захворювань опорно-рухового апарату. Для зменшення цього впливу необхідно правильно організувати робоче місце відповідно до ергономічних вимог. Робоче крісло повинно бути регульованим за висотою та кутом нахилу спинки, мати підлокітники й опору для поперекового відділу хребта.

Рекомендується забезпечити таке розташування клавіатури та маніпулятора «миша», яке дозволяє зберігати природне положення кистей рук і зменшувати напруження м’язів. Важливим заходом профілактики є чергування видів діяльності, виконання виробничої гімнастики та фізкультурних пауз під час регламентованих перерв. Дані заходи сприяють зниженню втоми, підвищенню працездатності та збереженню здоров’я працівників.

Заходи щодо оптимізації мікроклімату приміщень

Невідповідність параметрів мікроклімату нормативним значенням негативно впливає на самопочуття та працездатність людини. З метою зменшення впливу цього фактора необхідно забезпечити підтримання оптимальної температури, вологості та швидкості руху повітря в робочих

приміщеннях. Для цього використовуються системи опалення, вентиляції та кондиціонування повітря.

Рекомендується регулярне провітрювання приміщень, застосування зволожувачів повітря в опалювальний період, а також проведення контролю параметрів мікроклімату відповідно до чинних санітарних норм [30]. Забезпечення комфортного мікроклімату сприяє зниженню втоми, підвищенню концентрації уваги та загального рівня продуктивності праці.

Заходи щодо поліпшення освітлення робочих місць

Недостатнє або надмірне освітлення, а також наявність відблисків на екрані монітора є суттєвими чинниками зорової втоми. Для зменшення їх впливу необхідно раціонально поєднувати природне та штучне освітлення. Робочі місця слід розташовувати таким чином, щоб світло падало зліва або зліва-ззаду від користувача, а джерела світла не створювали прямих відблисків на екрані.

Штучне освітлення повинно забезпечувати рівномірну освітленість робочої поверхні відповідно до нормативних вимог. Доцільним є використання світильників із розсіяним світлом та можливістю регулювання яскравості. Дотримання цих заходів дозволяє зменшити напруження зору та підвищити комфортність умов праці.

Заходи щодо зниження впливу електромагнітних випромінювань, шуму та вібрації

Сучасні комп’ютерні пристрої, як правило, відповідають міжнародним стандартам безпеки та мають низькі рівні електромагнітних випромінювань. Проте для зменшення потенційного впливу цього фактора рекомендується використовувати лише сертифіковане обладнання, дотримуватися нормативних відстаней між робочими місцями та не розміщувати системні блоки й інші джерела випромінювання безпосередньо поблизу робочої зони користувача.

Для зниження шуму доцільно застосовувати малошумні комп’ютерні комплектуючі, а також розміщувати периферійні пристрої (принтери, сканери) у відокремлених приміщеннях або зонах. Дані заходи сприяють зменшенню нервового напруження та підвищенню комфортності праці.

Заходи з електро- та пожежної безпеки

Важливою складовою системи охорони праці при роботі на комп’ютері є забезпечення електробезпеки та пожежної безпеки. З цією метою необхідно використовувати справне електрообладнання, заземлені розетки та автоматичні вимикачі. Забороняється експлуатація пошкоджених кабелів, перевантаження електричних мереж та використання саморобних подовжувачів.

Робочі приміщення повинні бути обладнані первинними засобами пожежогасіння, а працівники – проінструктовані щодо дій у разі виникнення аварійних ситуацій. Дотримання вимог пожежної безпеки регламентується чинним законодавством України та відповідними нормативними документами [32].

Організаційні та профілактичні заходи

Окрім технічних і санітарно-гігієнічних заходів, важливу роль відіграють організаційні та профілактичні заходи. До них належать проведення інструктажів з охорони праці, навчання працівників безпечним методам роботи, а також періодичні медичні огляди. Раціональне планування робочого часу, дотримання режимів праці та відпочинку, а також формування культури безпечної праці є важливими умовами збереження здоров’я працівників.

Таким чином, комплексне впровадження заходів щодо зменшення впливу шкідливих і небезпечних факторів при роботі на комп’ютері дозволяє створити безпечні та комфортні умови праці, знизити ризик професійних захворювань і підвищити ефективність трудової діяльності.

6 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

6.1 Вплив людини на навколишнє середовище

Охорона навколишнього середовища являє собою форму відносин між суспільством і природою. Вона здійснюється різними засобами: економічними, правовими, науково-технічними, санітарно-гігієнічними, біологічними та іншими [33].

Наслідком обмеженості природних ресурсів є конкуренція за їх застосування, тобто суперництво між альтернативними цілями використання ресурсів. Адже майже всі ресурси можуть використовуватися для задоволення найрізноманітніших потреб.

У сучасних умовах розвитку цивілізації вплив людини на навколишнє середовище набув глобального характеру та став одним із ключових чинників, що визначають стан природних екосистем і якість життя населення. Інтенсивний розвиток промисловості, транспорту, енергетики, інформаційних технологій і зростання споживання природних ресурсів призвели до суттєвих змін у навколишньому природному середовищі. Антропогенний вплив проявляється у порушенні природної рівноваги, виснаженні ресурсів, забрудненні повітря, води й ґрунтів, а також у зміні клімату.

Одним із найбільш значущих напрямів впливу людини на довкілля є забруднення атмосферного повітря. Викиди шкідливих речовин унаслідок роботи промислових підприємств, теплових електростанцій та транспортних засобів спричиняють погіршення якості повітря, підвищення концентрації парникових газів і утворення смогу. Забруднення атмосфери негативно впливає на здоров’я людини, сприяє розвитку захворювань органів дихання та серцево-

судинної системи, а також призводить до деградації рослинного і тваринного світу.

Значний вплив на навколишнє середовище має також господарська діяльність людини, пов’язана з використанням водних ресурсів. Скидання недостатньо очищених або неочищених стічних вод у річки, озера та моря призводить до забруднення водних екосистем, зменшення біорізноманіття та погіршення якості питної води. Надмірне водокористування та регулювання річкового стоку порушують природні гідрологічні режими, що негативно позначається на стані прибережних територій і водних організмів.

Не менш важливою проблемою є забруднення та деградація ґрунтів, що виникають унаслідок інтенсивного землекористування, застосування мінеральних добрив і пестицидів, а також накопичення промислових і побутових відходів. Порушення структури ґрунтів, зменшення їх родючості та хімічне забруднення призводять до зниження врожайності сільськогосподарських культур і створюють загрозу для продовольчої безпеки.

Особливе місце серед екологічних проблем посідає утворення та накопичення відходів. Зростання обсягів виробництва та споживання сприяє збільшенню кількості твердих побутових і промислових відходів, значна частина яких не переробляється належним чином. Накопичення відходів на полігонах займає великі площі, спричиняє забруднення ґрунтів і підземних вод, а також негативно впливає на стан атмосферного повітря.

Вплив людини на навколишнє середовище проявляється і через зміни клімату, які значною мірою зумовлені антропогенними чинниками. Збільшення концентрації парникових газів у атмосфері призводить до підвищення середньої температури на планеті, зміни режимів опадів, зростання частоти екстремальних погодних явищ. Такі зміни створюють серйозні ризики для екосистем, економіки та соціальної сфери, а також потребують розробки й реалізації комплексних заходів адаптації та пом’якшення наслідків.

Разом із негативними аспектами, діяльність людини може мати й позитивний вплив на навколишнє середовище, за умови впровадження принципів сталого розвитку. Раціональне використання природних ресурсів, розвиток відновлюваних джерел енергії, упровадження екологічно безпечних технологій і систем очищення викидів сприяють зменшенню антропогенного навантаження. Важливу роль відіграє екологічна політика держави, спрямована на охорону довкілля, а також підвищення рівня екологічної свідомості населення.

У сучасному інформаційному суспільстві значення набуває також вплив інформаційних технологій на навколишнє середовище. З одного боку, розвиток цифрових технологій сприяє зменшенню використання паперових носіїв, оптимізації виробничих процесів і зниженню енергоспоживання. З іншого боку, виробництво та утилізація електронного обладнання створюють додаткові екологічні ризики, пов’язані з утворенням електронних відходів і використанням рідкісних ресурсів.

Отже, вплив людини на навколишнє середовище є багатогранним і комплексним явищем, що охоплює всі сфери суспільного життя. Усвідомлення масштабів антропогенного впливу та відповідальності за збереження природного середовища є необхідною умовою сталого розвитку. Реалізація екологічно орієнтованих рішень, дотримання природоохоронного законодавства та формування екологічної культури суспільства сприятимуть збереженню довкілля для майбутніх поколінь.

6.2 Розробка заходів щодо зменшення негативного впливу людини на навколишнє середовище

В загальному випадку проблема охорони навколишнього середовища зводиться до вирішення двох завдань:

– організації раціонального природокористування;

– забезпечення чистоти природних (екологічних) систем. При здійсненні різних видів економічної діяльності суб'єкти господарювання використовують різноманітні природні ресурси: землю, воду, корисні копалини тощо. Проте ресурси ці обмежені. Обмеженість природних ресурсів була і залишається головною і дуже жорсткою умовою, що накладається на розвиток економіки і відповідно зростання суспільного добробуту.

Сучасний рівень антропогенного навантаження на навколишнє середовище зумовлює необхідність розробки та впровадження системних заходів, спрямованих на зменшення негативного впливу діяльності людини на природні екосистеми. Такі заходи повинні охоплювати технічні, організаційні, економічні, правові та освітні аспекти і базуватися на принципах сталого розвитку, раціонального природокористування та екологічної відповідальності. Комплексний підхід до вирішення екологічних проблем дозволяє не лише зменшити шкоду довкіллю, а й забезпечити збалансований соціально-економічний розвиток суспільства.

Заходи щодо зменшення забруднення атмосферного повітря

Одним із пріоритетних напрямів зменшення негативного впливу людини на навколишнє середовище є скорочення викидів забруднюючих речовин в атмосферу. Для цього доцільним є впровадження сучасних екологічно безпечних технологій на промислових підприємствах, використання ефективних систем очищення газових викидів, а також модернізація застарілого обладнання.

Значну роль відіграє підвищення енергоефективності виробництва та зменшення споживання викопних видів палива.

У транспортній сфері ефективними заходами є розвиток громадського транспорту, стимулювання використання електромобілів і гібридних транспортних засобів, а також удосконалення транспортної інфраструктури з метою зменшення заторів і, відповідно, обсягів шкідливих викидів. Реалізація таких заходів відповідає вимогам екологічного законодавства України та міжнародним екологічним стандартам [33, 34].

Заходи щодо охорони та раціонального використання водних ресурсів

Зменшення негативного впливу на водні екосистеми потребує впровадження комплексу заходів, спрямованих на раціональне використання та охорону водних ресурсів. До таких заходів належать удосконалення систем очищення стічних вод, впровадження замкнених циклів водопостачання на промислових підприємствах, а також контроль за скиданням забруднюючих речовин у поверхневі та підземні води.

Важливим напрямом є також зменшення побутового водоспоживання шляхом використання водозберігаючих технологій і обладнання. Підвищення екологічної свідомості населення щодо раціонального використання води сприяє зниженню антропогенного навантаження на водні ресурси та покращенню їх екологічного стану [35].

Заходи щодо запобігання деградації ґрунтів

З метою зменшення негативного впливу людини на ґрунтовий покрив необхідно впроваджувати заходи, спрямовані на збереження родючості ґрунтів і запобігання їх забрудненню. У сільському господарстві такими заходами є застосування науково обґрунтованих сівозмін, зменшення використання

хімічних добрив і пестицидів, а також впровадження екологічно безпечних агротехнологій.

У промисловій та будівельній діяльності важливим є дотримання вимог щодо поводження з відходами, рекультивація порушених земель і контроль за розміщенням промислових об’єктів. Реалізація зазначених заходів дозволяє зменшити деградацію ґрунтів і зберегти їх екологічні функції [36].

Заходи щодо зменшення обсягів утворення та накопичення відходів

Проблема відходів є однією з найгостріших екологічних проблем сучасності. Зменшення негативного впливу відходів на навколишнє середовище можливе шляхом впровадження принципів циркулярної економіки, що передбачає зменшення обсягів утворення відходів, повторне використання матеріалів і розвиток системи їх переробки.

Ефективними заходами є роздільний збір побутових відходів, створення інфраструктури для їх сортування та переробки, а також стимулювання виробників до зменшення використання пакувальних матеріалів. Особливу увагу слід приділяти утилізації небезпечних і електронних відходів, які містять токсичні речовини та можуть завдавати значної шкоди довкіллю [37].

Заходи щодо протидії зміні клімату

Зміна клімату є глобальною проблемою, що потребує скоординованих дій на міжнародному, державному та локальному рівнях. Основними заходами щодо зменшення негативного впливу людини на клімат є скорочення викидів парникових газів, розвиток відновлюваних джерел енергії та підвищення енергоефективності в усіх секторах економіки.

Важливу роль відіграє також адаптація до кліматичних змін, що передбачає розробку та впровадження заходів із зменшення вразливості екосистем і населення до негативних наслідків глобального потепління.

Реалізація кліматичної політики базується на міжнародних угодах і національних стратегіях сталого розвитку [34, 38].

Організаційні, правові та освітні заходи

Значний вплив на зменшення негативного антропогенного впливу мають організаційні та правові заходи. До них належать удосконалення екологічного законодавства, посилення контролю за його дотриманням, а також впровадження економічних механізмів стимулювання екологічно відповідальної діяльності. Важливим є розвиток системи екологічного моніторингу та оцінки впливу на довкілля.

Не менш важливу роль відіграють освітні та просвітницькі заходи, спрямовані на формування екологічної культури населення. Підвищення рівня екологічної свідомості сприяє зміні поведінкових моделей, що, у свою чергу, зменшує негативний вплив людини на навколишнє середовище.

Таким чином, зменшення негативного впливу людини на навколишнє середовище можливе лише за умови комплексного підходу, який поєднує технічні, економічні, правові та соціальні заходи. Реалізація таких заходів є необхідною передумовою сталого розвитку та збереження природного середовища для майбутніх поколінь.

ВИСНОВКИ

Дана кваліфікаційна робота присвячена вирішенню актуального завдання інженерії програмного забезпечення – підвищенню достовірності оцінювання розміру ПЗ, що створюється мовою C#, та розробці програми для її реалізації.

Удосконалено нелінійну регресійну модель для оцінювання розміру ПЗ, створеного мовою C#, за рахунок застосування нормалізуючого перетворення на основі десяткового логарифму і викидів регресії, що дозволило підвищити достовірність оцінювання розміру ПЗ, створеного мовою C#, в порівнянні з існуючими моделями.

Для досягнення мети, поставленої в роботі, було вирішено такі завдання:

– визначено метрики, необхідні для оцінювання розміру ПЗ, що створюється мовою C#;

– проаналізовано існуючі моделі та методи оцінювання розміру ПЗ, що створюється мовою C#;

– обгрунтовано актуальності дослідження по обраній темі;

– проаналізовано методи, які застосовуються для побудови нелінійних регресійних моделей для оцінювання розміру ПЗ;

– проаналізовано методи, які застосовуються для попередньої обробки даних;

– відібрано проекти ПЗ з відкритим вихідним кодом, що створюються мовою C#, які використовувалися для побудови нелінійної регресійної моделі;

– побудовано нелінійну регресійну модель для оцінювання розміру ПЗ, що створюється мовою C#;

– перевірено якість побудованої нелінійної регресійної моделі;

– розроблено програму для оцінювання розміру ПЗ, створеного мовою C#, в якій реалізовано побудовану модель;

– проведено тестування та випробування розробленої програми згідно з методикою випробувань ПЗ;

– була розроблена уся необхідна програмна документація.

В кваліфікаційній роботі також були виконані спеціальні розділи з розрахунку економічної ефективності від розробки і впровадження програмного забезпечення для оцінювання розміру ПЗ, що створюється мовою C#, охорони праці, охорони навколишнього середовища.

Мета роботи досягнута, усі завдання виконано в повному обсязі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. C#. The modern, innovative, open-source programming language for building all your apps.. URL: https://dotnet.microsoft.com/en-us/languages/csharp (дата звернення: 11.09.2025 р.).

2. TIOBE Index for December 2024. URL: https://www.tiobe.com/tiobe-index/ (дата звернення: 11.09.2025 р.).

3. The C# Programming Language. URL: https://www.tiobe.com/tiobe-index/csharp/ (дата звернення: 11.09.2025 р.).

4. Рейтинг мов програмування 2025. TypeScript і Python – найпопулярніші, частки C# та Java зменшуються. URL: https://dou.ua/lenta/articles/language-rating-2025/ (дата звернення: 11.09.2025 р.).

5. Приходько С.Б., Приходько Н.В., Фаріонова Т.А., Ворона М.В. Трьохфакторна нелінійна регресійна модель для оцінювання розміру PHP-застосунків із відкритим кодом. *Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки.* Київ, 2020. № 1(70) Ч.1. С. 124–131.

6. Prykhodko N.V., Prykhodko S.B. The non-linear regression model to estimate the software size of open-source Java-based systems. *Радіоелектроніка, інформатика, управління.* Запоріжжя: ЗНТУ, 2018. № 3(46). С. 158–166.

7. Каіров В.О., Ларіонов В.А, Ласяк Р.О. Ранне оцінювання розміру програмних застосунків мовою C#. *Сучасні інформаційні системи та технології: матеріали VII Всеукр. наук.-практ. інтернет-конф. за тематикою «Сучасні комп'ютерні системи та мережі в управлінні»* (29 листопада 2024 р., м. Херсон, м. Хмельницький) / за ред. А.А. Григорової. Херсон: Книжкове видавництво ФОП Вишемирський В.С., 2024. С. 176-177.

8. Петриченко С.А., Пухалевич А.В. Багатофакторна нелінійна регресійна модель для оцінювання розміру вебзастосунків, що створюються з використанням мови програмування C#. *Інформаційні технології: моделі, алгоритми, системи (ITMAS – 2023): Матеріали IV Всеукраїнської науково-практичної інтернет конференції* (30-31 жовтня 2023 р.). Миколаїв: НУК імені адмірала Макарова, 2023. С. 40-42.

9. Латанська Л.О., Устенко А.С. Розробка регресійної моделі для оцінювання розміру програмного забезпечення, створеного за допомогою платформи ASP.NET. *Матеріали XIII Міжнародної науково-практичної конференції "Free and open source software" (FOSS-2021)* (м. Харків, 16-18 листопада 2021 р.). Харків, 2021. С 53–54.

10. Латанська, Л.О., Макарова, Л.М., Нікітіна, О.Ю., Нікітін, О.В. (). Математичне моделювання в задачах оцінювання розміру прикладного програмного забезпечення з відкритим кодом на C#. *Computer Science and Applied Mathematics,* 2022. С. 66-74. https://doi.org/10.26661/2413-6549-2022-1-08

11. Kiewkanya M., Surak S. Constructing C++ software size estimation model from class diagram. *13th International Joint Conference on Computer Science and Software Engineering (JCSSE),* Khon Kaen, Thailand, 13-15 July 2016. URL: https://doi.org/10.1109/jcsse.2016.7748880

12. Доценко А.С., Макарова Л.М., Фаріонова Т.А. Нелінійна регресійна модель для оцінювання розміру програмного забезпечення, що створюється мовою C#. *Матеріали VIII Всеукраїнської науково-практичної інтернет-конференції студентів, аспірантів та молодих вчених за тематикою «Сучасні комп'ютерні системи та мережі в управлінні»* (24 листопада 2025 р., м. Херсон, м. Хмельницький) / за ред. А.А. Григорової. Херсон: Книжкове видавництво ФОП Вишемирський В.С., 2025. С. 145-146.

13. Beginner's Guide to C# Programming: Features, Advantages, and Practical Uses. URL: https://www.examsnap.com/certification/beginners-guide-to-c-

programming-features-advantages-and-practical-uses/ (дата звернення: 15.09.2025 р.).

14. Interfaces. URL: https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/object-oriented/inheritance#interfaces (дата звернення: 15.09.2025 р.).

15. Об'єктно-орієнтовна парадигма. URL: https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/tutorials/oop (дата звернення: 15.09.2025 р.).

16. 5 Reasons to Choose C# for Your Next Programming Project. URL: https://docsallover.com/blog/general-purpose-programming/5-reasons-to-choose-c-sharp/ (дата звернення: 15.09.2025 р.).

17. Managed code. URL: https://docs.microsoft.com/en-us/dotnet/standard/managed-code (дата звернення: 15.09.2025 р.).

18. Exploring the Power of C#: A Comprehensive Guide. URL: https://www.aclti.com/en/cutting-edge-technologies/exploring-the-power-of-c-a-comprehensive-guide (дата звернення: 15.09.2025 р.).

19. Exception Class. URL: https://docs.microsoft.com/en-us/dotnet/api/system.exception?view=net-5.0 (дата звернення: 15.09.2025 р.).

20. Language Integrated Query (LINQ). URL: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/ (дата звернення: 15.09.2025 р.).

21. Task asynchronous programming model. URL: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/async/task-asynchronous-programming-model (дата звернення: 15.09.2025 р.).

22. Fenton N, Bieman J. Software Metrics: A Rigorous and Practical Approach, Third Edition. CRC Press, 2014. 617 p.

23. GitHub. URL: https://github.com (дата звернення: 18.09.2025 р.).

24. Dream big. Achieve more. Visual Studio. URL: https://visualstudio.microsoft.com/ (дата звернення: 18.09.2025 р.).

25. Приходько С.Б., Макарова Л.М., Приходько Н.В., Пухалевич А.В. Методичні вказівки та завдання до виконання лабораторних робіт з дисципліни "Емпіричні методи програмної інженерії". Миколаїв: НУК, 2023. 56 с.

26. Chatterjee S., Hadi A.S. Regression analysis by example. Fifth Edition. New Jersey: A John Wiley & sons, Inc., Publication, 2012. 403 p.

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

28. Aggarwal C.C. Outlier analysis. 2nd ed. New York : Springer International Publishing AG, 2017. 481 p.

29. Prykhodko S., Prykhodko N., Makarova L., Pukhalevych A. Application of the Squared Mahalanobis Distance for Detecting Outliers in Multivariate Non-Gaussian Data. *Proceedings of 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET).* Lviv-Slavske, 2018. P. 962-965.a

30. Про охорону праці: Закон України від 15.11.2024 № 2694-XII. URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text (дата звернення: 10.11.2025 р.).

31. Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями: Наказ Міністерства соціальної політики України від 14.02.2018 № 207. URL: https://zakon.rada.gov.ua/laws/show/z0508-18#Text (дата звернення: 10.11.2025 р.).

32. Правила пожежної безпеки в Україні: Наказ МВС України від 30.12.2014 р. № 1417 (зі змінами). URL: https://zakon.rada.gov.ua/laws/show/z0252-15 (дата звернення: 10.11.2025 р.).

33. Про охорону навколишнього природного середовища: Закон України від 15.11.2024 № 1264-XII. URL: https://zakon.rada.gov.ua/laws/show/1264-12#Text (дата звернення: 13.11.2025 р.).

34. Стратегія сталого розвитку України до 2030 року: Указ Президента України від 30.09.2019 р. № 722/2019. URL: https://zakon.rada.gov.ua/laws/show/722/2019 (дата звернення: 13.11.2025 р.).

35. Водний кодекс України: Закон України від 6.06.1995 р. № 213/95-ВР (зі змінами). URL: https://zakon.rada.gov.ua/laws/show/213/95-вр (дата звернення: 13.11.2025 р.).

36. Земельний кодекс України: Закон України від 25.10.2001 р. № 2768-III (зі змінами). URL: https://zakon.rada.gov.ua/laws/show/2768-14 (дата звернення: 13.11.2025 р.).

37. Про управління відходами: Закон України від 20.06.2022 р. № 2320-IX. URL: https://zakon.rada.gov.ua/laws/show/2320-20 (дата звернення: 13.11.2025 р.).

38. Паризька угода: Міжнародний договір, ратифікований Законом України від 14.07.2016 р. № 1469-VIII. URL: https://zakon.rada.gov.ua/laws/show/995_l61 (дата звернення: 13.11.2025 р.).

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЩО СТВОРЮЄТЬСЯ МОВОЮ C#

Вступ

Повна назва розроблюваного проекту: Програма для оцінювання розміру ПЗ, створеного мовою C#.

Коротка назва: програма.

Галузь застосування: програма призначена для застосування в компаніях та командах розробників, що займаються розробкою ПЗ мовою C#.

1 Підстави для розробки

Підставою для розробки є завдання на кваліфікаційну роботу, видане кафедрою ПЗАС НУК ім. адмірала Макарова.

2 Призначення розробки

2.1 Функціональне призначення програми

Дана програма призначена для оцінювання розміру ПЗ, створеного мовою C#.

2.2 Експлуатаційне призначення програми

Програма може використовуватися для оцінювання розміру ПЗ, створеного мовою C#, в компаніях та командах розробників, що займаються розробкою відповідного ПЗ.

3 Вимоги до програмного забезпечення

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу виконуваних функцій

Програма повинна забезпечувати можливість виконання наведених нижче функцій:

– ввод даних про параметри нелінійної регресії: значення коефіцієнтів регресійного рівняння b_0, b_1 та b_2;

– ввод значення довірчої ймовірності α для розрахунку;

– ввод даних про ПЗ, створене мовою C#, для якого потрібно виконати оцінювання розміру: значення метрик X_1 (кількість класів) та X_2 (середня кількість методів у класі);

– оцінювання розміру ПЗ, створеного мовою C#: обчислення залежної змінної Y (кількість рядків коду у тисячах) та границь довірчого інтервалу і інтервалу прогнозування;

– збереження вхідних даних в БД;

– збереження результатів оцінювання розміру ПЗ, створеного мовою C#, в БД;

– експорт результатів розрахунків у файл формату .CSV.

3.1.2 Вимоги до організації вхідних та вихідних даних

Вимоги до організації вхідних даних

Вхідні дані: параметри нелінійної регресії b_0, b_1 та b_2, значення довірчої ймовірності α, значення метрик X_1 та X_2.

Вхідні дані вносяться у відповідні поля форм згідно з допустимим типом даних: числовий.

Вимоги до організації вихідних даних

Вихідні дані: прогнозне значення розміру ПЗ, створеного мовою C#, значення границь довірчого інтервалу та інтервалу прогнозування.

Вихідні дані виводяться у відповідні поля форм та зберігаються у БД.

3.2 Вимоги до надійності

Вимоги до забезпечення надійного функціонування ПЗ має бути забезпеченим виконанням сукупності організаційно-технічних заходів, перелік яких наведений нижче:

– організацією безперебійного живлення ПК або сервера;

– безперебійним каналом зв’язку.

Час відновлення після відмови

Час відновлення після відмови, викликаної проблемами з електроживленням або каналом зв’язку, не фатальним збоєм (не крахом) операційної системи, не перевищує 3 хвилин, які необхідні для перезавантаження ПК або відновлення каналу зв’язку.

3.3 Вимоги до умов експлуатації

3.3.1 Кліматичні умови експлуатації

Кліматичні умови експлуатації, при яких повинні забезпечуватися задані характеристики, повинні задовольняти вимогам, пропонованим до технічних засобів у частині умов їхньої експлуатації.

3.3.2 Вимоги до видів обслуговування

Програмний продукт не вимагає проведення будь-яких видів обслуговування.

3.3.3 Вимоги до чисельності та кваліфікації персоналу

З програмою працює один користувач. Необхідний рівень підготовки користувачів: кінцевий користувач повинен мати практичні навички роботи з комп`ютером та розумітися на предметній області.

3.4 Вимоги до складу та параметрів технічних засобів

Система повинна задовольняти вказаним вимогам на комп’ютері наступної мінімальної комплекції:

– CPU: Intel(R) i5 2,16 МГц або аналогічний за параметрами AMD;

– RAM: 4 GB;

– HDD: 10 GB вільного місця.

3.5 Вимоги до інформаційної та програмної сумісності

Для повноцінного функціонування програми необхідно використовувати ОС Windows версії не нижче 10, .Net Framework 4.8.1.

3.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред’являються.

3.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред’являються.

4 Вимоги до програмної документації

Програмна документація повинна містити:

- технічне завдання;
- код програми;
- опис програми;

– інструкцію користувача;
– програму та методику випробувань ПЗ.

5 Техніко-економічні показники

Економічна ефективність впровадження ПЗ розрахована в розділі 4 кваліфікаційної роботи. Згідно з отриманими результатами впровадження даного ПЗ є доцільним.

6 Стадії та етапи розробки

Стадії та етапи розробки програми для оцінювання розміру ПЗ, створеного мовою C#, представлені в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки програмного забезпечення

Стадії розробки	Етапи розробки	Термін виконання
1. Технічне завдання	Розробка та затвердження технічного завдання	21.09.2025
2. Ескізний проект	Розробка ескізного проекту	18.10.2025
	Затвердження ескізного проекту	21.10.2025
3. Технічний проект	Розробка технічного проекту	03.11.2025
	Затвердження технічного проекту	06.11.2025
4. Робочий проект	Розробка програми	16.11.2025
	Розробка програмної документації	26.11.2025
	Випробування програми	01.12.2025

7 Порядок контролю та приймання

Контроль за аналізом та проектуванням кожної окремої частини програмного забезпечення на кожному етапі з урахуванням вимог, визначених у технічному завдані. Кожна стадія розробки повинна бути представлена в зазначені строки та узгоджена із замовником.

Приймання проводяться відповідно до програми і методики випробувань і документують за допомогою протоколу проведення випробувань. У разі знаходження помилок підчас прийому програмного виробу складається акт про знайдені помилки, який підписуються представниками замовника і розробника і затверджуються керівником організації – замовника та організації – розробника. Розробник повинен протягом не більше ніж двох тижнів виправити зазначені помилки і оповістити замовника про повторне проведення перевірки.

ДОДАТОК Б – КОД ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЩО СТВОРЮЄТЬСЯ МОВОЮ C#

```
MainWindow.xaml
<Window x:Class="SoftwareSizeEstimator.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Software Size Estimation System" Height="500"
Width="450" ResizeMode="NoResize"
WindowStartupLocation="CenterScreen">
    <Grid Margin="10">

        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!-- Розділ 1: Параметри регресії -->
        <GroupBox Header="Параметри регресії" Grid.Row="0"
Margin="0,0,0,10">
            <Grid Margin="10">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="150"/>
                    <ColumnDefinition Width="*"/>
                </Grid.ColumnDefinitions>

                <TextBlock Text="b0:" VerticalAlignment="Center"/>
                <TextBox x:Name="B0TextBox" Grid.Column="1"
Width="200"/>

                <TextBlock Text="b1:" VerticalAlignment="Center"
Grid.Row="1"/>
                <TextBox x:Name="B1TextBox" Grid.Column="1"
Grid.Row="1" Width="200"/>

                <TextBlock Text="b2:" VerticalAlignment="Center"
Grid.Row="2"/>
                <TextBox x:Name="B2TextBox" Grid.Column="1"
Grid.Row="2" Width="200"/>

                <TextBlock Text="α (довірча ймовірність):"
VerticalAlignment="Center" Grid.Row="3"/>
```

```
                <TextBox x:Name="AlphaTextBox" Grid.Column="1"
Grid.Row="3" Width="200"/>
            </Grid>
        </GroupBox>

        <!-- Розділ 2: Метрики ПЗ -->
        <GroupBox Header="Метрики ПЗ" Grid.Row="1"
Margin="0,0,0,10">
            <Grid Margin="10">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="150"/>
                    <ColumnDefinition Width="*"/>
                </Grid.ColumnDefinitions>

                <TextBlock Text="Кількість класів (X1):"
VerticalAlignment="Center"/>
                <TextBox x:Name="X1TextBox" Grid.Column="1"
Width="200"/>

                <TextBlock Text="Середня кількість методів (X2):"
VerticalAlignment="Center" Grid.Row="1"/>
                <TextBox x:Name="X2TextBox" Grid.Column="1"
Grid.Row="1" Width="200"/>
            </Grid>
        </GroupBox>

        <!-- Розділ 3: Результати -->
        <GroupBox Header="Результати оцінювання" Grid.Row="2"
Margin="0,0,0,10">
            <Grid Margin="10">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="180"/>
                    <ColumnDefinition Width="*"/>
                </Grid.ColumnDefinitions>

                <TextBlock Text="Кількість рядків коду (Y):"
VerticalAlignment="Center"/>
                <TextBlock x:Name="YTextBlock" Grid.Column="1"
Width="200"/>

                <TextBlock Text="Довірчий інтервал:"
VerticalAlignment="Center" Grid.Row="1"/>
                <TextBlock x:Name="ConfidenceIntervalTextBlock"
Grid.Column="1" Grid.Row="1" Width="200"/>

                <TextBlock Text="Інтервал прогнозування:"
VerticalAlignment="Center" Grid.Row="2"/>
                <TextBlock x:Name="PredictionIntervalTextBlock"
Grid.Column="1" Grid.Row="2" Width="200"/>
```

```
            </Grid>
        </GroupBox>

        <!-- Розділ 4: Кнопки дій -->
        <StackPanel Grid.Row="3" Orientation="Horizontal"
HorizontalAlignment="Center" Margin="0,10,0,0" Spacing="10">
            <Button x:Name="CalculateButton" Content="Розрахувати"
Width="120" Height="30" Click="CalculateButton_Click"/>
            <Button x:Name="SaveButton" Content="Зберегти"
Width="120" Height="30" Click="SaveButton_Click"/>
            <Button x:Name="ExportButton" Content="Експорт у CSV"
Width="120" Height="30" Click="ExportButton_Click"/>
        </StackPanel>

    </Grid>
</Window>


MainWindow.xaml.cs
using System;
using System.Collections.Generic;
using System.Windows;
using Microsoft.Win32;
using SoftwareSizeEstimator.DataAccess;
using SoftwareSizeEstimator.Models;
using SoftwareSizeEstimator.Services;
using SoftwareSizeEstimator.Export;

namespace SoftwareSizeEstimator
{
    public partial class MainWindow : Window
    {
        private readonly ValidationService _validationService;
        private readonly RegressionCalculator
_regressionCalculator;
        private readonly IntervalCalculator _intervalCalculator;
        private readonly DatabaseContext _dbContext;
        private readonly CsvExporter _csvExporter;

        public MainWindow()
        {
            InitializeComponent();

            _validationService = new ValidationService();
            _regressionCalculator = new RegressionCalculator();
            _intervalCalculator = new IntervalCalculator();
            _dbContext = new DatabaseContext("SoftwareSize.db");
            _csvExporter = new CsvExporter();
        }
```

```
        private void CalculateButton_Click(object sender,
RoutedEventArgs e)
        {
            try
            {
                // Читання введених даних
                double b0 = double.Parse(B0TextBox.Text);
                double b1 = double.Parse(B1TextBox.Text);
                double b2 = double.Parse(B2TextBox.Text);
                double alpha = double.Parse(AlphaTextBox.Text);
                int x1 = int.Parse(X1TextBox.Text);
                double x2 = double.Parse(X2TextBox.Text);

                // Валідація
                if
(!_validationService.ValidateRegressionParameters(b0, b1, b2,
alpha, out string error1))
                {
                    MessageBox.Show(error1, "Помилка введення",
MessageBoxButton.OK, MessageBoxImage.Error);
                    return;
                }

                if (!_validationService.ValidateMetrics(x1, x2, out
string error2))
                {
                    MessageBox.Show(error2, "Помилка введення",
MessageBoxButton.OK, MessageBoxImage.Error);
                    return;
                }

                // Розрахунок
                double y = _regressionCalculator.CalculateY(b0, b1,
b2, x1, x2);
                Interval ci =
_intervalCalculator.CalculateConfidenceInterval(y, alpha);
                Interval pi =
_intervalCalculator.CalculatePredictionInterval(y, alpha);

                // Відображення результатів
                YTextBlock.Text = y.ToString("F3");
                ConfidenceIntervalTextBlock.Text = ci.ToString();
                PredictionIntervalTextBlock.Text = pi.ToString();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка обробки даних: " +
ex.Message, "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);
```

```
            }
        }

        private void SaveButton_Click(object sender,
RoutedEventArgs e)
        {
            try
            {
                // Зчитування даних (аналогічно
CalculateButton_Click)
                double b0 = double.Parse(B0TextBox.Text);
                double b1 = double.Parse(B1TextBox.Text);
                double b2 = double.Parse(B2TextBox.Text);
                double alpha = double.Parse(AlphaTextBox.Text);
                int x1 = int.Parse(X1TextBox.Text);
                double x2 = double.Parse(X2TextBox.Text);
                double y = double.Parse(YTextBlock.Text);
                Interval ci =
_intervalCalculator.CalculateConfidenceInterval(y, alpha);
                Interval pi =
_intervalCalculator.CalculatePredictionInterval(y, alpha);

                var result = new EstimationResult
                {
                    B0 = b0,
                    B1 = b1,
                    B2 = b2,
                    Alpha = alpha,
                    X1 = x1,
                    X2 = x2,
                    Y = y,
                    ConfidenceInterval = ci,
                    PredictionInterval = pi
                };

                _dbContext.SaveResult(result);
                MessageBox.Show("Результати успішно збережено.",
"Інформація", MessageBoxButton.OK, MessageBoxImage.Information);
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка збереження: " +
ex.Message, "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }

        private void ExportButton_Click(object sender,
RoutedEventArgs e)
        {
```

```
            try
            {
                List<EstimationResult> results =
_dbContext.GetResults();

                SaveFileDialog saveFileDialog = new SaveFileDialog
                {
                    Filter = "CSV files (*.csv)|*.csv",
                    FileName = "Results.csv"
                };

                if (saveFileDialog.ShowDialog() == true)
                {
                    _csvExporter.Export(results,
saveFileDialog.FileName);
                    MessageBox.Show("Файл CSV успішно створено.",
"Інформація", MessageBoxButton.OK, MessageBoxImage.Information);
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка експорту: " + ex.Message,
"Помилка", MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }
    }
}
```

App.xaml

```
<Application x:Class="SoftwareSizeEstimator.App"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             StartupUri="MainWindow.xaml">
    <Application.Resources>
        <!-- Місце для глобальних ресурсів (стилі, шаблони) -->
    </Application.Resources>
</Application>
```

App.xaml.cs

```
using System.Windows;

namespace SoftwareSizeEstimator
{
    public partial class App : Application
    {
```

```
        // Можна додати код для глобальної обробки помилок або
ініціалізації
    }
}


DataAccess/DatabaseContext.cs
using System.Collections.Generic;
using System.Data.SQLite;
using SoftwareSizeEstimator.Models;

namespace SoftwareSizeEstimator.DataAccess
{
    public class DatabaseContext
    {
        private readonly string _connectionString;

        public DatabaseContext(string databasePath)
        {
            _connectionString = $"Data
Source={databasePath};Version=3;";
            InitializeDatabase();
        }

        private void InitializeDatabase()
        {
            using var connection = new
SQLiteConnection(_connectionString);
            connection.Open();

            string sql = @"
                CREATE TABLE IF NOT EXISTS EstimationResults (
                    Id INTEGER PRIMARY KEY AUTOINCREMENT,
                    B0 REAL,
                    B1 REAL,
                    B2 REAL,
                    Alpha REAL,
                    X1 INTEGER,
                    X2 REAL,
                    Y REAL,
                    ConfidenceLower REAL,
                    ConfidenceUpper REAL,
                    PredictionLower REAL,
                    PredictionUpper REAL
                );";

            using var command = new SQLiteCommand(sql, connection);
            command.ExecuteNonQuery();
        }
```

```
        public void SaveResult(EstimationResult result)
        {
            using var connection = new
SQLiteConnection(_connectionString);
            connection.Open();

            string sql = @"
                INSERT INTO EstimationResults
                (B0, B1, B2, Alpha, X1, X2, Y,
                 ConfidenceLower, ConfidenceUpper,
                 PredictionLower, PredictionUpper)
                VALUES
                (@B0, @B1, @B2, @Alpha, @X1, @X2, @Y,
                 @CL, @CU, @PL, @PU);";

            using var command = new SQLiteCommand(sql, connection);
            command.Parameters.AddWithValue("@B0", result.B0);
            command.Parameters.AddWithValue("@B1", result.B1);
            command.Parameters.AddWithValue("@B2", result.B2);
            command.Parameters.AddWithValue("@Alpha",
result.Alpha);
            command.Parameters.AddWithValue("@X1", result.X1);
            command.Parameters.AddWithValue("@X2", result.X2);
            command.Parameters.AddWithValue("@Y", result.Y);
            command.Parameters.AddWithValue("@CL",
result.ConfidenceInterval.LowerBound);
            command.Parameters.AddWithValue("@CU",
result.ConfidenceInterval.UpperBound);
            command.Parameters.AddWithValue("@PL",
result.PredictionInterval.LowerBound);
            command.Parameters.AddWithValue("@PU",
result.PredictionInterval.UpperBound);

            command.ExecuteNonQuery();
        }

        public List<EstimationResult> GetResults()
        {
            var results = new List<EstimationResult>();
            using var connection = new
SQLiteConnection(_connectionString);
            connection.Open();

            string sql = "SELECT * FROM EstimationResults;";
            using var command = new SQLiteCommand(sql, connection);
            using var reader = command.ExecuteReader();

            while (reader.Read())
```

```
            {
                var result = new EstimationResult
                {
                    B0 = reader.GetDouble(1),
                    B1 = reader.GetDouble(2),
                    B2 = reader.GetDouble(3),
                    Alpha = reader.GetDouble(4),
                    X1 = reader.GetInt32(5),
                    X2 = reader.GetDouble(6),
                    Y = reader.GetDouble(7),
                    ConfidenceInterval = new
Interval(reader.GetDouble(8), reader.GetDouble(9)),
                    PredictionInterval = new
Interval(reader.GetDouble(10), reader.GetDouble(11))
                };
                results.Add(result);
            }

            return results;
        }
    }
}
```

Models/Interval.cs

```
namespace SoftwareSizeEstimator.Models
{
    public class Interval
    {
        public double LowerBound { get; set; }
        public double UpperBound { get; set; }

        public Interval(double lowerBound, double upperBound)
        {
            LowerBound = lowerBound;
            UpperBound = upperBound;
        }

        public override string ToString()
        {
            return $"[{LowerBound:F3}; {UpperBound:F3}]";
        }
    }
}
```

Models/EstimationResult.cs

```
using SoftwareSizeEstimator.Models;
```

```
namespace SoftwareSizeEstimator.Models
{
    public class EstimationResult
    {
        public double B0 { get; set; }
        public double B1 { get; set; }
        public double B2 { get; set; }
        public double Alpha { get; set; }
        public int X1 { get; set; }
        public double X2 { get; set; }
        public double Y { get; set; }
        public Interval ConfidenceInterval { get; set; }
        public Interval PredictionInterval { get; set; }
    }
}


Services/ValidationService.cs
namespace SoftwareSizeEstimator.Services
{
    public class ValidationService
    {
        public bool ValidateRegressionParameters(double b0, double
b1, double b2, double alpha, out string errorMessage)
        {
            if (alpha <= 0 || alpha >= 1)
            {
                errorMessage = "Значення α повинно належати
інтервалу (0; 1).";
                return false;
            }

            errorMessage = string.Empty;
            return true;
        }

        public bool ValidateMetrics(int x1, double x2, out string
errorMessage)
        {
            if (x1 <= 0)
            {
                errorMessage = "Кількість класів (X1) повинна бути
додатним цілим числом.";
                return false;
            }

            if (x2 <= 0)
            {
```

```
                errorMessage = "Середня кількість методів у класі
(X2) повинна бути додатним числом.";
                return false;
            }

            errorMessage = string.Empty;
            return true;
        }
    }
}


Export/CsvExporter.cs
using System.Collections.Generic;
using System.Globalization;
using System.IO;
using System.Text;
using SoftwareSizeEstimator.Models;

namespace SoftwareSizeEstimator.Export
{
    public class CsvExporter
    {
        public void Export(List<EstimationResult> results, string
filePath)
        {
            var sb = new StringBuilder();

sb.AppendLine("B0;B1;B2;Alpha;X1;X2;Y;ConfidenceLower;ConfidenceUpp
er;PredictionLower;PredictionUpper");

            foreach (var r in results)
            {

sb.AppendLine(string.Format(CultureInfo.InvariantCulture,
                    "{0};{1};{2};{3};{4};{5};{6};{7};{8};{9};{10}",
                    r.B0, r.B1, r.B2, r.Alpha, r.X1, r.X2, r.Y,
                    r.ConfidenceInterval.LowerBound,
r.ConfidenceInterval.UpperBound,
                    r.PredictionInterval.LowerBound,
r.PredictionInterval.UpperBound));
            }

            File.WriteAllText(filePath, sb.ToString(),
Encoding.UTF8);
        }
    }
}
```

ДОДАТОК В – ОПИС ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЩО СТВОРЮЄТЬСЯ МОВОЮ C#

1 Загальні відомості

Повна назва розроблюваного проекту: Програма для оцінювання розміру ПЗ, створеного мовою C#.

Коротка назва: програма.

Галузь застосування: програма призначена для застосування в компаніях та командах розробників, що займаються розробкою ПЗ мовою C#.

2 Функціональне призначення

Дана програма призначена для оцінювання розміру ПЗ, створеного мовою C#.

Програма повинна забезпечувати можливість виконання наведених нижче функцій:

– ввод даних про параметри нелінійної регресії: значення коефіцієнтів регресійного рівняння b_0, b_1 та b_2;

– ввод значення довірчої ймовірності α для розрахунку;

– ввод даних про ПЗ, створене мовою C#, для якого потрібно виконати оцінювання розміру: значення метрик X_1 (кількість класів) та X_2 (середня кількість методів у класі);

– оцінювання розміру ПЗ, створеного мовою C#: обчислення залежної змінної Y (кількість рядків коду у тисячах) та границь довірчого інтервалу і інтервалу прогнозування;

– збереження вхідних даних в БД;

– збереження результатів оцінювання розміру ПЗ, створеного мовою C#, в БД;

– експорт результатів розрахунків у файл формату .CSV.

3 Експлуатаційне призначення

Програма може використовуватися для оцінювання розміру ПЗ, створеного мовою C#, в компаніях та командах розробників, що займаються розробкою відповідного ПЗ.

4 Технічні засоби, що використовуються

Система повинна задовольняти вказаним вимогам на комп’ютері наступної мінімальної комплекції:

– CPU: Intel(R) i5 2,16 МГц або аналогічний за параметрами AMD;

– RAM: 4 GB;

– HDD: 10 GB вільного місця.

5 Програмні засоби, що використовуються

Для повноцінного функціонування програми необхідно використовувати ОС Windows версії не нижче 10, .Net Framework 4.8.1.

6 Вхідні дані

Вхідні дані: параметри нелінійної регресії b_0, b_1 та b_2, значення довірчої ймовірності α, значення метрик X_1 та X_2.

Вхідні дані вносяться у відповідні поля форм згідно з допустимим типом даних: числовий.

7 Вихідні дані

Вихідні дані: прогнозне значення розміру ПЗ, створеного мовою C#, значення границь довірчого інтервалу та інтервалу прогнозування.

Вихідні дані виводяться у відповідні поля форм та зберігаються у БД.

ДОДАТОК Г – ІНСТРУКЦІЯ КОРИСТУВАЧА ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЩО СТВОРЮЄТЬСЯ МОВОЮ C#

Програма для оцінювання розміру ПЗ, створеного мовою C#, встановлюється безпосередньо на комп’ютер користувача. Програма запускається або за допомогою ярлика, який знаходиться на робочому столі, або за допомогою вибору відповідного пункту меню «Пуск» – «Програми». Програма має один режим використання.

Після запуску програми з’являється головне вікно програми, для оцінювання розміру ПЗ, створеного мовою C#, яке представлено на рисунку Г.1.

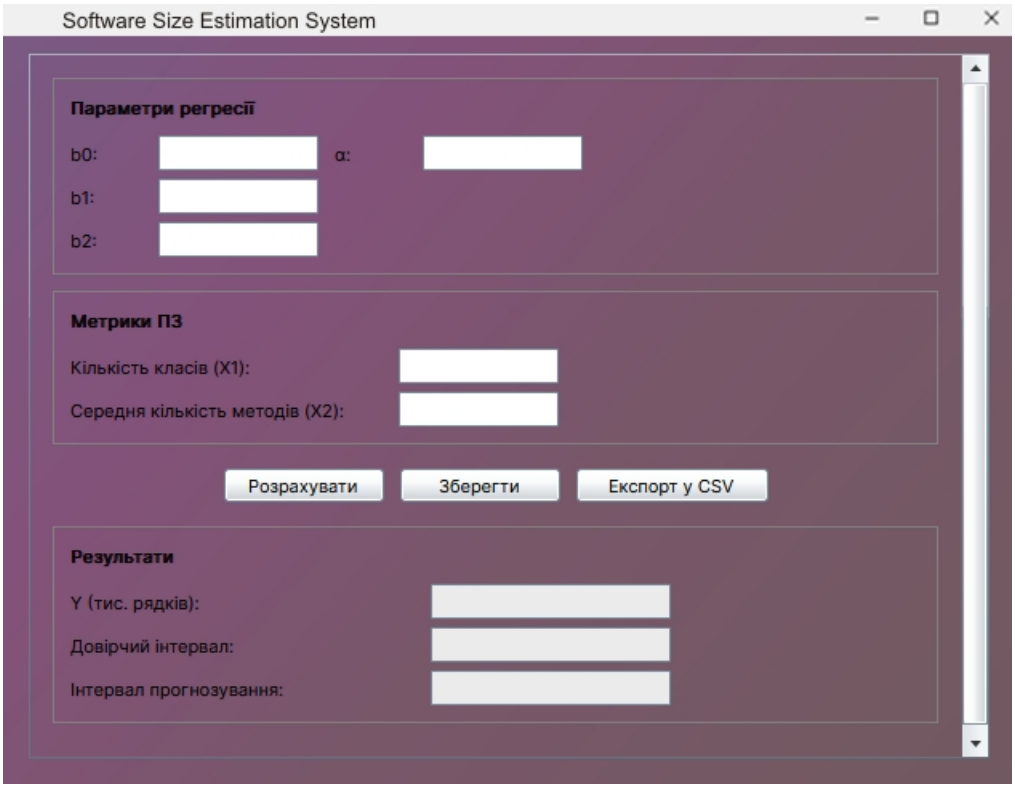

Рисунок Г.1 – Головне вікно програми для оцінювання розміру ПЗ, створеного мовою C#

Головне вікно програми має три умовних частини: верхня – "Параметри регресії", середня – "Метрики ПЗ", нижня – "Результати". Між середньою та нижньою частинами вікна знаходиться блок кнопок управління: "Розрахувати", "Зберегти", "Експорт у CSV".

Верхня частина вікна призначена для введення даних про параметри регресії і містить такі поля вводу: "b_0", "b_1" та "b_2" і значення довірчої ймовірності "α".

Середня частина вікна призначена для введення даних про метрики ПЗ і містить такі поля вводу: "Кількічть класів (X1)", "Середня кількість методів (X2).

Нижня частина вікна призначена для виводу результатів розрахунку і містить такі поля: "Y (тис. рядків)", "Довірчий інтервал", "Інтервал прогнозування". Ці поля мають ознаку "Тільки для читання".

Для початку роботи треба ввести дані про параметри регресії та метрики ПЗ і натиснути кнопку "Розрахувати" (див. рис. Г.2).

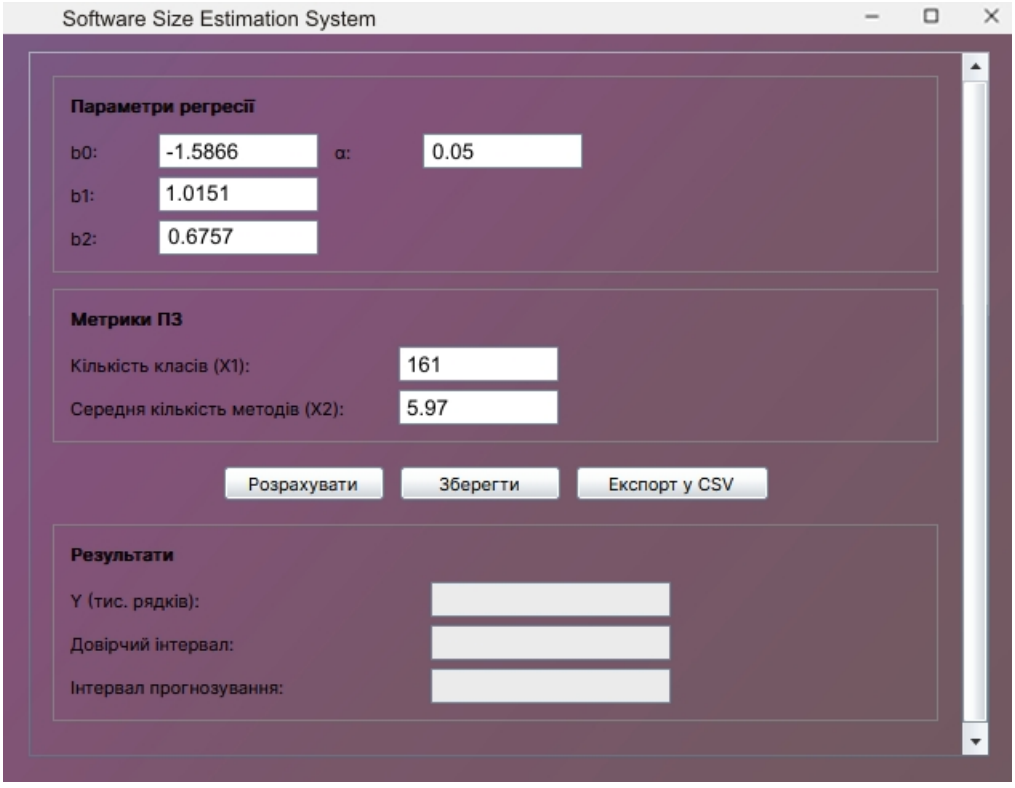

Рисунок Г.2 – Зовнішній вигляд вікна з введеними даними

Розраховані значення розміру ПЗ, створеного мовою C#, та значення границь інтервалів з`являться у відповідних полях.

Зовнішній вигляд вікна з розрахунками представлено на рисунку Г.3.

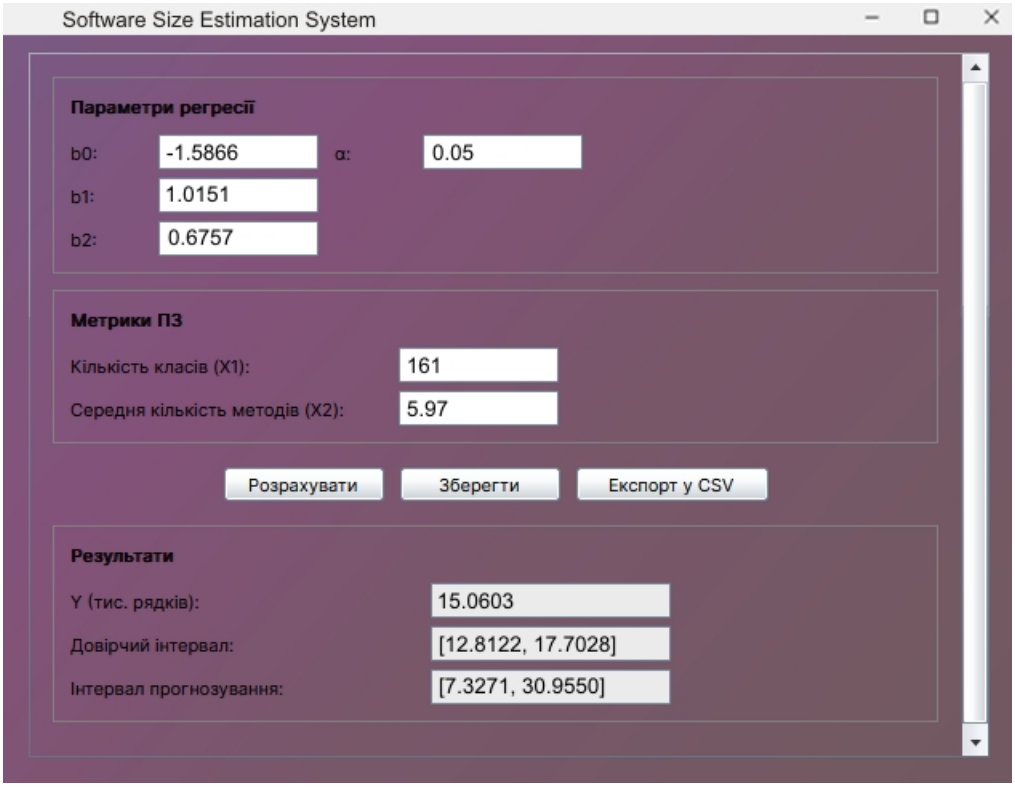

Рисунок Г.3 – Зовнішній вигляд вікна з розрахунками

Для збереження результатів розрахунків у БД потрібно натиснути кнопку "Зберегти". Для експорту результатів розрахунків у файл формату .CSV потрібно натиснути кнопку "Експорт у CSV".

ДОДАТОК Д – ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАННЯ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЩО СТВОРЮЄТЬСЯ МОВОЮ C#

1 Об’єкт випробовування

Об’єкт випробувань: Програма для оцінювання розміру ПЗ, створеного мовою C#.

2 Мета випробовування

Перевірка функціональних можливостей програмного забезпечення для оцінювання розміру ПЗ, створеного мовою C#, на відповідність вимогам технічного завдання.

3 Вимоги до програми

При проведенні випробувань, функціональні характеристики програмного забезпечення підлягають перевірці на відповідність вимогам, викладеним у пункті «Вимоги до функціональних характеристик» технічного завдання.

Для проведення випробувань підлягають перевірці наступні функції:

– Ввод даних про параметри розрахунку (значень коефіцієнтів регресійного рівняння b_0, b_1 та b_2 і значення довірчої ймовірності α).

– Ввод даних про ПЗ, створене мовою C# (значень метрик X_1 та X_2).

– Збереження вхідних даних в БД.

– Оцінювання розміру ПЗ, створеного мовою C#.

– Збереження результатів оцінювання в БД.

– Експорт результатів розрахунків.

4 Вимоги до програмної документації

Програмна документація повинна містити:

– технічне завдання;

– текст програми;

– опис програми;

– інструкцію користувача;

– програму та методику випробування ПЗ.

5 Засоби та порядок випробувань

Система повинна задовольняти вказаним вимогам на комп’ютері наступної мінімальної комплекції:

– CPU: Intel(R) i5 2,16 МГц або аналогічний за параметрами AMD;

– RAM: 4 GB;

– HDD: 10 GB вільного місця.

Програмні засоби, що повинні використовуватися під час випробувань

Для повноцінного функціонування програми необхідно використовувати ОС Windows версії не нижче 10, .Net Framework 4.8.1.

Випробування програмного забезпечення повинні виконуватися 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 програмного продукту складається їх перелік і обговорюється термін їх виправлення розробником. Після цього замовник проводить повторне тестування в повному обсязі (можливе використання нових чи додаткових тестів).

6 Методи випробувань

Випробування будемо проводити за стратегією «чорного ящика». Всі функції програми будуть проходити випробовування. В таблиці Д.1 наведено перелік випробувань, що пропонуються до виконання:

Таблиця Д.1 – Методика випробувань

№	Дія	Очікуваний результат
1	Ввод даних про параметри розрахунку	Відображення результату введення значень коефіцієнтів регресійного рівняння b_0, b_1 та b_2 і значення довірчої ймовірності α
2	Ввод даних про ПЗ, створене мовою C#	Відображення результату введення значень метрик X_1 та X_2
3	Збереження вхідних даних в БД	Запис в БД значень вхідних даних b_0, b_1 та b_2 і α
4	Оцінювання розміру ПЗ, створеного мовою C#	Відображення результату обчислення залежної змінної Y (кількість рядків коду у тисячах) та границь довірчого інтервалу і інтервалу прогнозування
5	Збереження результатів оцінювання в БД	Запис в БД значень вихідних даних Y та границь довірчого інтервалу і інтервалу прогнозування
6	Експорт результатів розрахунків	Експорт результатів розрахунків у файл формату .CSV