МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут
комп'ютерних наук та управління проектами
(повне найменування інституту, назва факультету)

Програмного забезпечення автоматизованих систем
(повна назва кафедри)

Пояснювальна записка

до кваліфікаційної (магістерської) роботи

за темою

«Удосконалення однофакторної регресійної моделі для оцінювання кількості дефектів програмного забезпечення та розробка програми для її реалізації»

Виконав: студент 6 курсу, групи 6151м
спеціальності
121 «Інженерія програмного забезпечення»
(шифр і назва спеціальності)

Пушкін О.С.
(підпис, прізвище та ініціали)
Керівник Макарова Л.М.
(підпис, прізвище та ініціали)
Рецензент Приходько С.Б.
(підпис, прізвище та ініціали)
Завідувач кафедри Приходько С.Б.
(підпис, прізвище та ініціали)

м. Миколаїв – 2020 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
Освітній ступень Магістр
Галузь 12 «Інформаційні технології»
(шифр і назва)
Спеціальність 121 «Інженерія програмного забезпечення»
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри

" 26 " 10 2020 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ (МАГІСТЕРСЬКУ) РОБОТУ СТУДЕНТУ

Пушкін Олександр Сергійович
(прізвище, ім'я, по батькові)

1.Тема магістерської роботи «Удосконалення однофакторної регресійної моделі для оцінювання кількості дефектів програмного забезпечення та розробка програми для її реалізації»
керівник роботи Макарова Лідія Миколаївна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від " 26 " жовтня 2020 року №1037-уч
2. Строк подання студентом роботи 01.12.2020 року
3. Вихідні дані до роботи

4.Зміст магістерської роботи (МР):
• Титульний аркуш, завдання на кваліфікаційну (магістерську) роботу, реферат (українською, англійською), зміст, перелік умовних позначень, символів, одиниць та термінів (за необхідності).
• 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 за темою, обґрунтування необхідності проведення досліджень за обраною темою, вибір напрямків досліджень, мета дослідження, основні задачі дослідження
• Викладення результатів власних досліджень з висвітленням того нового, що пропонується
• Проект програмного забезпечення
• Результати досліджень та розробки проекту 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 і методика випробувань програмного забезпечення)

5. Перелік графічного матеріалу

_____
_____
_____
_____
_____
_____

6. Консультанти розділів магістерської роботи

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

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

Назва етапів кваліфікаційної (магістерської) роботи (МР)	Термін виконання	Примітка
1. Підготовка вступної частини МР	18.10.2020	
2. Підготовка розділу (ів) МР з огляду літератури за темою, обґрунтування необхідності проведення досліджень за обраною темою, вибір напрямів досліджень	19.10.2020	
3. Підготовка розділу (ів) МР з результатів власних досліджень	22.10.2020	
4. Підготовка розділу МР з проекту програмного забезпечення	16.11.2020	
5. Підготовка організаційно-економічного розділу	18.11.2020	
6. Підготовка розділу з охорони праці	20.11.2020	
7. Підготовка розділу з охорони навколишнього середовища	23.11.2020	
8. Підготовка розділу МР – Висновки	25.11.2020	
9. Оформлення списку використаних джерел	27.11.2020	
10. Оформлення додатків	30.11.2020	
11. Підготовка презентації МР та доповіді	01.12.2020	
12. Подання МР на попередній захист	01.12.2020	
13. Подання МР рецензенту	11.12.2020	
14. Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, разом з заявою щодо самостійності виконання роботи та ідентичності друкованої та електронної версії роботи	14.12.2020	
15. Подання на кафедру ПЗАС електронних версії наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК від 19.05.2020 р. за №287-уч); презентації доповіді	18.12.2020	
16. Подання на кафедру ПЗАС письмового відгуку та рецензії на кваліфікаційну роботу	18.12.2020	

Студент _____ Пушкін О.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Макарова Л.М.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пушкін Олександр Сергійович

«Удосконалення однофакторної регресійної моделі для оцінювання кількості дефектів програмного забезпечення та розробка програми для її реалізації»

Кваліфікаційна робота на здобуття освітнього рівня магістра зі спеціальності 121 – «Інженерія програмного забезпечення». Національний університет кораблебудування імені адмірала Макарова. Миколаїв, 2020 р.

Обсяг роботи: 104 стор., 14 табл., 19 рис., 29 використаних джерел, 5 додатків.

Актуальність теми роботи: підвищення достовірності оцінювання кількості дефектів ПЗ в залежності від його розміру є важливим завданням, яке потребує удосконалення існуючої регресійної моделі, оскільки існуючі апріорні моделі є або недостовірними або не досить точними.

Мета та завдання дослідження: підвищення достовірності оцінювання кількості дефектів ПЗ в залежності від його розміру.

Об'єкт дослідження: процес оцінювання кількісті дефектів ПЗ.

Предмет дослідження: однофакторна регресійна модель для оцінювання кількості дефектів ПЗ.

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

Наукова новизна одержаних результатів: удосконалено однофакторну регресійну модель для оцінювання кількості дефектів ПЗ за рахунок використання нормалізуючого перетворення на основі десяткового логарифму, що дає змогу отримати більш точну оцінку кількості дефектів ПЗ в порівнянні з існуючими моделями.

Практичне значення одержаних результатів: програма для оцінювання кількості дефектів ПЗ.

Апробація результатів досліджень: основні положення і результати досліджень, викладені у кваліфікаційній роботі, пройшли апробацію на III Всеукраїнській науково-практичній інтернет-конференції студентів, аспірантів та молодих вчених за тематикою «Сучасні комп'ютерні системи та мережі в управлінні» (м. Херсон, 30 листопада 2020 р.).

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

Ключові слова: регресійна модель, нелінійна регресія, нормалізуюче перетворення, кількість дефектів ПЗ, достовірність оцінювання.

ABSTRACT

Pushkin Oleksandr Serhiiovych

«Improvement of the univariate regression model for estimating the number of software defects and developing the software for its implementation»

The qualification work in obtaining a master's degree in specialty 121 – "Software Engineering". Admiral Makarov National University of Shipbuilding. Mykolayiv, 2020.

Volume of work: 104 pages of typewritten text, 14 tables, 19 figures, a list of references from 29 names, 5 appendices.

Relevance of the theme: improving the reliability of estimating the number of software defects depending on its size is an important task that requires improvement of the existing regression model, because existing a priori models are either unreliable or not accurate enough.

The purpose and objectives of the study: increasing the reliability of estimating the number of software defects depending on its size.

Object of study: the process of estimating the number of software defects.

Subject of study: a univariate nonlinear regression model for estimating the number of software defects.

Research methods: methods of probability theory and mathematical statistics, mathematical modeling, regression analysis, object-oriented programming.

Scientific novelty of the obtained results: the improved of univariate regression model for estimating of the number of software defects by using a normalizing transformation based on the decimal logarithm, which made it possible to increase the accuracy of estimating of the number of software defects compared to existing model.

The practical significance of the obtained results: the software for estimating of the number of software defects.

Approbation of research results: the main provisions and research results presented in the qualification work were tested at the III All-Ukrainian scientific-practical Internet conference of students, graduate students and young scientists on "Modern computer systems and networks in management" (Kherson, November 30, 2020).

Publications: the main results of the qualification work are presented in 1 scientific paper – conference proceedings.

Keywords: regression model, nonlinear regression, normalizing transformation, the number of software defects, reliability of estimating.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ВВ	випадкова величина
ЕД	емпіричні дані
ООП	об'єктно-орієнтоване програмування
ПЗ	програмне забезпечення
ПК	персональний комп`ютер
KSLOCs	thousand Source/Software Lines of Code
MMRE	Mean of Magnitude of Relative Errors
RUP	Rational Unified Process
UCP	use casepoints
UML	Unified Modeling Language

ВСТУП

Актуальність теми. ПЗ сучасних інформаційних систем являє собою надзвичайно складний комплекс, в створенні якого можуть брати участь десятки і навіть сотні фахівців. Багато дослідників називають складність сучасного ПЗ ключовим фактором виникнення дефектів. З ростом складності і появою більшої кількості дефектів ПЗ збільшуються розміри втрат і залежність кінцевих користувачів ПЗ від його безвідмовної роботи.

З іншого боку, зниження кількості дефектів в сучасному ПЗ пояснюється прогресом методів, засобів і технологій інженерії ПЗ, які дозволяють проводити більш якісне тестування ПЗ та виявлення помилок ще на стадії розробки, збільшенням ступеня повторного використання бездефектного коду, можливістю сучасних середовищ розробки автоматично генерувати бездефектний код. Але удосконалення методів для зменшення та моделей для оцінювання кількості дефектів ПЗ є актуальними задачами сучасної інженерії ПЗ.

На даний момент існує проблема достовірного і точного оцінювання кількості дефектів ПЗ. Тому розробка моделей для оцінювання кількості дефектів ПЗ є актуальною задачею, рішення якої дозволить отримати покращену оцінку кількості дефектів, що в подальшому допоможе оптимальним чином спланувати необхідні ресурси для усунення дефектів, досягти необхідної надійності, підвищити ефективність розробки і експлуатації ПЗ. В якості оцінювання кількості дефектів на етапі розробки використовуються апріорні моделі. Вони дозволяють оцінити кількість дефектів, внесених розробниками на етапі проектування і кодування. Для оцінювання кількості дефектів апріорні моделі використовують показник кількості рядків програмного коду (Lines of Code). Найбільш відомою апріорної моделлю є модель Холстеда (Halstead) [1].

В основу розробки моделі покладені дві базові характеристики ПЗ: словник операторів, операндів мови програмування, число використання операторів і операндів в програмних реалізаціях, а також гіпотеза, що частота використання операторів і операндів в ПЗ пропорційна бінарному логарифму кількості їх типів. Недоліком даної моделі є велика похибка обчислень.

Ще однією апріорною моделью оцінювання кількості дефектів є модель Акіями, яка досліджена на предмет достовірності і точності оцінювання кількості дефектів [2]. Дослідження показало, що лінійні моделі деяких простих метрик забезпечують оцінки для всіх дефектів, які фактично визначаються як сума дефектів, виявлених в ході випробувань, і дефектів найдених протягом двох місяців [3]. Виявлена закономірність перевищення розрахункової кількості дефектов над фактичним в кілька разів.

Оскільки при розробці сучасного ПЗ спостерігається тенденція зниження кількості дефектів ПЗ, у тому числі за рахунок повторного використання бездефектного коду та можливості сучасних середовищ розробки автоматично генерувати бездефектний код, ці моделі втратили свою актуальність, їх оцінка является недостовірною. Для оцінювання кількості дефектів було запропоновано використовувати багатофакторну модель. Дана модель представляє собою лінійну модель оцінки по пяти емпіричним характеристикам ПЗ: логічна складність, складність взаємозв'язку, складності розрахунків, складності вводу – виводу, і зрозумілості [4].

Дана модель не враховує фактору повторного використання бездефектного коду. Для отримання значимих коефіцієнтів кореляції моделі необхідний експериментальний ряд показників складності і дефектів ПЗ, що ускладнює використання моделі.

Таким чином, існуючі апріорні моделі, які оцінюють кількість дефектів ПЗ, є або недостовірними (модель Акіями), або не досить точними (моделі Холстеда і багатофакторна модель). Вони показують значні відхилення розрахункової кількості дефектів від дійсних показників.

Удосконалення регресійної моделі для оцінювання кількості дефектів ПЗ в залежності від його розміру дозволить отримати покращену оцінку кількості дефектів ПЗ.

Оскільки використання багатофакторної моделі ускладнено великою кількістю розрахунків і трудомісткістю, в якості початкового наближення було обрано однофакторну модель для оцінювання кількості дефектів ПЗ.

Таким чином, для підвищення достовіності оцінювання кількості дефектів в залежності від розміру ПЗ необхідно удосконалити однофакторну регресійну модель для оцінювання кількості дефектів ПЗ і розробити програму для її реалізації.

Мета і завдання дослідження. Метою роботи є підвищення достовірності оцінювання кількості дефектів ПЗ в залежності від його розміру.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

– дослідити існуючі моделі оцінювання кількості дефектів ПЗ;

– обґрунтувати необхідність удосконалення однофакторної регресійної моделі для оцінювання кількості дефектів ПЗ;

– побудувати однофакторну регресійну модель для оцінювання кількості дефектів ПЗ на основі логарифмічного перетворення;

– перевірити емпіричні дані на викиди;

– побудувати лінійне рівняння регресії, довірчий інтервал та інтервал прогнозування для нормалізованих даних;

– від лінійної регресії перейти до нелінійної та побудувати рівняння регресії, довірчий інтервал та інтервал прогнозування для вхідних даних;

– за допомогою мови програмування C# розробити програму для оцінювання кількісті дефектів ПЗ в залежності від його розміру.

Об'єкт дослідження: процес оцінювання кількісті дефектів ПЗ.

Предмет дослідження: однофакторна регресійна модель для оцінювання кількості дефектів ПЗ.

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

Наукова новизна одержаних результатів полягає в удосконаленні однофакторної регресійної моделі для оцінювання кількості дефектів ПЗ за рахунок використання нормалізуючого перетворення на основі десяткового логарифму, що дає змогу отримати більш точну оцінку кількості дефектів ПЗ в порівнянні з існуючими моделями.

Практичне значення одержаних результатів. Програма для оцінювання кількості дефектів ПЗ, реалізована за допомогою об’єктно-орієнтованої мови програмування C#.

Особистий внесок здобувача. Кваліфікаційна робота є самостійно виконаною працею. Усі результати, викладені у роботі, отримані автором особисто. У праці, яка опублікована у співавторстві з керівником кваліфікаційної роботи [5], здобувачеві належить удосконалення однофакторної регресійної моделі для оцінювання кількості дефектів ПЗ.

Апробація результатів роботи. Основні положення і результати досліджень, викладені у кваліфікаційній роботі, пройшли апробацію на III Всеукраїнській науково-практичній інтернет-конференції студентів, аспірантів та молодих вчених за тематикою «Сучасні комп’ютерні системи та мережі в управлінні» (м. Херсон, 30 листопада 2020 р.).

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

1 АНАЛІЗ ІСНУЮЧИХ МОДЕЛЕЙ ДЛЯ ОЦІНЮВАННЯ КІЛЬКОСТІ ДЕФЕКТІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Метрики програмного забезпечення

Процес створення ПЗ настільки багатогранний і одночасно складний для сприйняття, що навіть незначні недоліки системи неминуче відіб'ються на надійності ПЗ. Основу будь-якої системи вимірювання складають окремі показники, які називаються метриками. Метрики ПЗ є засобом для вимірювання якості ПЗ. Стандарт ISO 9126:2001 регламентує зовнішні і внутрішні характеристики якості [6]. Перші відображають вимоги до функціонування програмного продукту. Для кількісного встановлення критеріїв якості, за якими буде здійснюватися перевірка і підтвердження відповідності ПЗ заданим вимогам, визначаються відповідні зовнішні вимірювані властивості (зовнішні атрибути) ПЗ, метрики (наприклад, час виконання окремих компонентів), діапазони зміни значень і моделі їх оцінки. Метрики використовуються на стадії тестування або функціонування і називаються зовнішніми метриками. Вони являють собою моделі оцінки атрибутів.

Метрика ПЗ (Software metric) – захід, що дозволяє отримати чисельне значення деякої властивості ПЗ або його специфікацій [7].

Оскільки кількісні методи добре зарекомендували себе в інших областях, багато теоретиків намагалися перенести даний підхід і в розробку ПЗ.

Щоб визначити, за якими показниками можна порівнювати метрики кількості дефектів в залежності від розмру ПЗ, розглянемо основні групи метрик докладніше. До основних метрик ПЗ, що застосовують на практиці відносять кількість рядків коду, метрики Холстеда, об'єктно-орієнтовані метрики та деякі інші.

Кількість рядків коду (Source lines of code) – це розмірно орієнтована метрика ПЗ, в якій обсяг ПЗ розраховується, виходячи з кількості рядків в тексті вихідного коду [8].

Серед метрик розміру кількості рядків є дві основні:

– по кількості фізичних рядків (LOC) – визначається як загальне число рядків вихідного коду, включаючи коментарі і порожні рядки;

– по кількості логічних рядків коду (LLOC) – визначається як загальна кількість команд і залежить від використовуваної мови програмування.

Також є похідні від основних методик, які в залежності від завдання можуть містити додаткову інформацію за такими показниками:

– число порожніх рядків;

– число рядків, що містять коментарі;

– відсоток коментарів (відношення рядків коду до рядків коментаря, похідна метрика стилістики);

– середнє число рядків для функцій (класів, файлів);

– середня кількість рядків, що містять вихідний код для функцій (класів, файлів);

– середнє число рядків для модулів та інші.

При цьому у даній метриці є ряд переваг. Наприклад, дані щодо кількості рядків в завершених проектах або модулях програм можуть бути легко зібрані за допомогою інтегрованих середовищ розробки (IDE) або спеціальних програм.

Метрики, засновані на аналізі кількості рядків і синтаксичних елементів вихідного коду програми, були запропоновані М. Холстедом в 1977 р [9]. Метрики Холстеда (Halstead complexity measures) частково дозволяють врахувати можливість реалізації однієї і тієї ж функціональності різним числом рядків і операторів. Найбільш частим сценарієм використання

цих метрик є оцінка складності проміжних продуктів розробки, проте набір метрик також містить оцінки розміру.

Хоча метрики Холстеда дозволяють використовувати додаткові можливості з аналізу вихідного коду, які відсутні в метриці кількості рядків коду, з точки зору оцінювання розміру проекту велика частина завдань залишається невирішеною. Оцінюввання загального числа операторів і їх операндів до завершення етапу розробки ПЗ може виявитися ще більш складним завданням, ніж оцінювання кількості рядків коду.

Аналіз функціональних точок (Function points) – це метод вимірювання розміру ПЗ з точки зору користувачів системи. Метод був розроблений Аланом Альбрехтом в середині 1970-х років, вперше опублікований в 1979 р. Даний метод призначений для оцінювання обсягу програмного продукту по функціональної моделі так як оцінюється обсяг функцій, що розробляється. Основна мета цього методу полягає в декомпозиції системи таким чином, щоб забезпечити прийнятну складність аналізу. Базовою одиницею виміру, на якій ґрунтується даний метод, є функціональна точка [10]. Однак використання функціональних точок в якості одиниць виміру має ряд істотних недоліків. Для обчислення функціональних точок необхідно детально вивчити специфікацію вимог і підрахувати всі вхідні і вихідні елементи, файли, транзакції, що може бути дуже трудомістким.

Метод точок властивостей (Feature points) являє собою розширення методу функціональних точок, яке враховує не тільки вимоги до системи, але і особливості її реалізації. Головна відмінність від методу функціональних точок полягає в тому, що одержувана оцінка використовується з урахуванням алгоритмічної складності реалізації. До вихідних елементів додається ще один – алгоритм. У методі точок властивостей алгоритм визначаються як набір правил, що визначає якусь істотну обчислювальну задачу. Завдяки обліку алгоритмічної складності точки властивостей краще адаптовані до оцінки систем з високою складністю алгоритмів.

В сучасних методологіях використання об'єктно-орієнтованої методології розробки займає особливе місце, фактично є найбільш широко поширеною методологією. Оскільки в початковому варіанті методу функціональних точок не було передбачено застосування об'єктно-орієнтованого підходу, був розроблений адаптований варіант, який оперує термінами ООП. Його принциповою відмінністю від інших варіацій методу функціональних точок є те, що він не продовжує стандартний набір типів елементів, а використовує зовсім інші: форми (Screens definitions), звіти (User reports), модулі (3GL Modules). Фактично в цій метриці кожному унікальному класу або об'єкту призначається одна об'єктна точка (Object points). Дана метрика зручна для оцінювання проектів, які проектуються по об'єктно-орієнтованій методології і мають велике число компонентів з візуальною складовою. Оскільки визначення кількості точок більше орієнтоване на візуальні компоненти, в проектах з високою алгоритмічною складністю підрахунок об'єктів може ускладнений.

Метод UCP являє собою оцінку розміру проектів на основі діаграм UML і методології RUP. Як і багато інших сучасних методів оцінювання, UCP базується приблизно на тих же принципах, що і метод функціональних точок. Головна відмінність полягає в заміні одиниць вимірювання з функціональних точок на варіанти використання (Use Cases) [11].

Оскільки метрики є важливою частиною багатьох моделей оцінювання кількості дефектів ПЗ, розробка повинна починатися саме з вибору метрики.

1.2 Існуючі моделі для оцінювання кількості дефектів програмного забезпечення

Для кількісної оцінки показників надійності ПЗ використовують різноманітні моделі надійності, під якими розуміють математичні моделі, побудовані для оцінювання залежності надійності від заздалегідь відомих або

визначених у ході виконання завдання параметрів. Ці моделі можна розділити на дві основні групи: емпіричні та аналітичні. Емпіричні моделі засновані на аналізі накопиченої інформації про функціонування раніше розробленого ПЗ. Найбільш проста емпірична модель пов'язує число помилок ПЗ з його обсягом [12].

Аналітичні моделі поділяються на статичні і динамічні. Серед динамічних також можна виділити безперервні і дискретні. При використанні безперервної динамічної моделі передбачається, що функціонування ПЗ описується набором послідовних станів, перехід між якими відбувається у випадку виникнення відмови. До безперервної динамічної моделі відносяться модель Джелінского – Моранді і модель перехідних ймовірностей Маркова. У дискретних моделях передбачається, що спочатку проводиться тестування ПЗ (можливо, у кілька етапів). У разі появи відмов шукаються і виправляються всі помилки, через які сталися відмови. Після цього починається період експлуатації ПЗ. Прикладами дискретних моделей можуть послужити модель Шумана і модель Мусса.

Статичні моделі відрізняються від динамічних насамперед тим, що в них не враховується час появи помилок. До статичних моделей відносяться модель Міллса, модель Нельсона, модель Коркорена.

Проблема вибору моделей надійності для використання на етапі експлуатації і супроводу ПЗ практично не досліджена. Причинами цього є велика кількість моделей та недостатній рівень вивчення особливостей їх використання.

Класифікація моделей надійності ПЗ наведена на рис. 1.1.

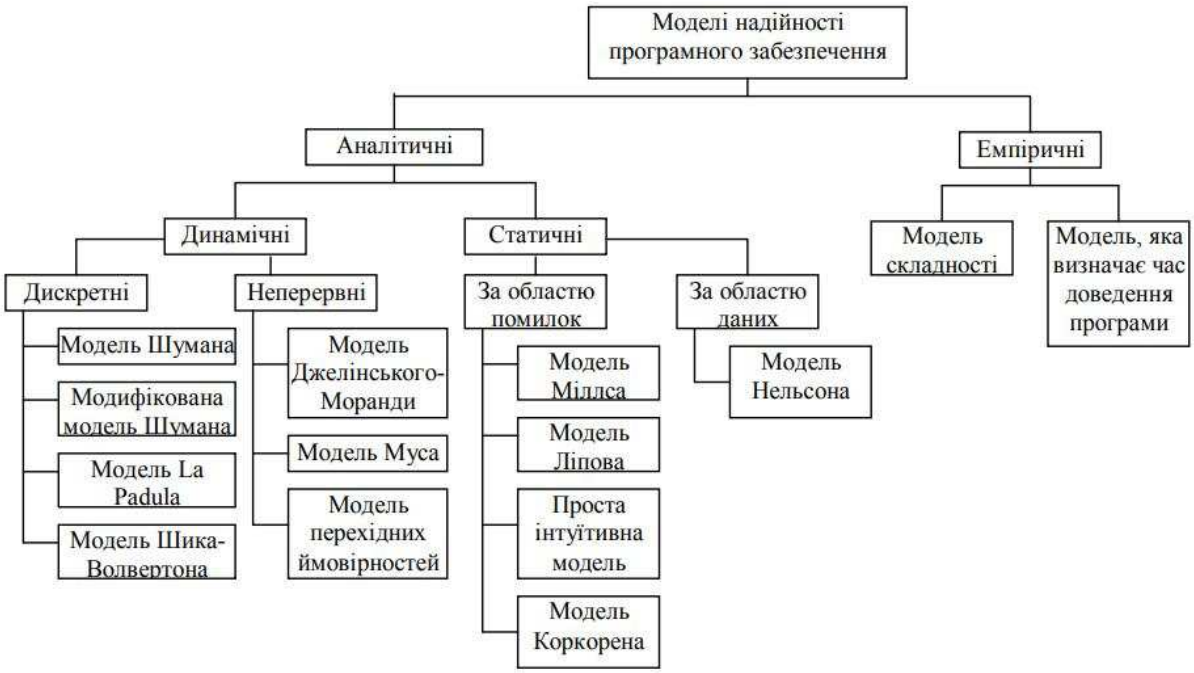

Рисунок 1.1 – Класифікація моделей надійності ПЗ

Так як між надійністю і складністю ПЗ існує тісний зв'язок, то проблем надійності стосується ще одна група моделей – моделі, призначені для оцінювання складності ПЗ. Всі існуючі моделі складності показників визначають тільки окремі, приватні характеристики складного ПЗ. Загальним поняттям для всіх видів складного ПЗ є його структура.

При аналізі структурної складності ПЗ з метою визначення його надійності необхідно проводити багатокрокову процедуру зниження його складності. Поширення складних програмних систем і комплексів вимагає нових підходів до оцінки їх надійності. Складність вирішення цього завдання обумовлена відсутністю універсальних методів і моделей.

Є й інші моделі, які в тому чи іншому вигляді дають оцінки надійності. Вони часто використовують будь-яку додаткову інформацію про процес розробки, наприклад дані про покриття коду тестами або деяку інформацію про організацію процесу розробки. Як наслідок, можливість застосування таких моделей часто обмежена, через що вони знаходять мало застосування на практиці.

1.3 Способи удосконалення однофакторної регресійної моделі для оцінювання кількості дефектів програмного забезпечення

Зазвичай дані для оцінювання кількості дефектів в залежності від розміру ПЗ не відповідають нормального розподілу, що в результаті призводить до помилок у розрахунках та може негативно вплинути на достовірність отриманих результатів, у даному випадку на оцінку кількості дефектів в залежності від розміру ПЗ. Для уникнення цієї проблеми потрібно нормалізувати вихідні дані.

В якості нормалізації негаусівських даних можуть бути використані перетворення на основі десяткового або натурального логарифму, перетворення Бокса – Кокса, перетворення Джонсона та інші. В якості нормалізуючого перетворення будемо використовувати нормалізуюче перетворення на основі десяткового логарифму, яке, з одного боку, достатньо просте для виконання розрахунків, а з іншого – дає непогані показники якості перетворення, наприклад, із використанням стандартизації даних.

Регресійну модель у загальному вигляді можна представити наступним рівнянням:

$$y = \bar{y} + \varepsilon_t = f(x) + \varepsilon_t, \tag{1.1}$$

де y – залежна змінна;

$f(x)$ – функція, яка визначає вид регресійної моделі (лінійна або нелінійна);

x – незалежна змінна;

ε_t – випадкова помилка.

Отримавши нормалізовані дані, можна побудувати лінійну регресійну модель для нормалізованих даних, для неї побудувати довірчий інтервал та інтервал прогнозування і шляхом застосування зворотнього перетворення перейти до нелінійної регресійної моделі та її інтервалів.

Лінійну регресійну модель у загальному вигляді можна представити наступним рівнянням:

$$z_y = b_1 z_x + b_0 + \varepsilon, \tag{1.2}$$

де b_1 та b_0 – коефіцієнти лінійного рівняння регресії, знайти які можна методом найменших квадратів:

$$b_1 = \frac{n\sum_{i=1}^{n} z_{xi} z_{yi} - \sum_{i=1}^{n} z_{xi} \cdot \sum_{i=1}^{n} z_{yi}}{n\sum_{i=1}^{n} z_{xi}^2 - \left(\sum_{i=1}^{n} z_{xi}\right)^2},$$
$$b_0 = \frac{\sum_{i=1}^{n} z_{yi} - b_1 \sum_{i=1}^{n} z_{xi}}{n}, \tag{1.3}$$

ε – ВВ, розподілена за нормальним законом, що визначається як ε~N(0,1).

Побудувавши лінійне рівняння регресії, для більш достовірного оцінювання можна також побудувати довірчий інтервал та інтервал прогнозування лінійного рівняння регресії.

Довірчий інтервал лінійного рівняння регресії будується за допомогою наступної формули:

$$y = \hat{y} \pm t_{(\alpha/2, n-2)} \cdot S \cdot \sqrt{\frac{1}{n} + \frac{(x - \bar{x})^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2}}, \tag{1.4}$$

де $\hat{y}$ – значення y, розраховане за рівнянням регресії;

$t_{(\alpha/2,n-2)}$ – квантіль t-розподілу Стьюдента;

α – рівень значущості;

n – кількість значень випадкових величин у вибірці;

$$S = \sqrt{\frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}\ .$$

Інтервал прогнозування лінійного рівняння регресії будується за допомогою наступної формули:

$$y = \hat{y} \pm t_{(\alpha/2,n-2)} \cdot S \cdot \sqrt{1 + \frac{1}{n} + \frac{(x - \bar{x})^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}}, \qquad (1.5)$$

де $\hat{y}$ – значення у, розраховане за рівнянням регресії;

$t_{(\alpha/2,n-2)}$ – квантіль *t*-розподілу Стьюдента;

α – рівень значущості;

n – кількість значень ВВ у вибірці;

$$S = \sqrt{\frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}\ .$$

Таким чином, використовуючи нормалізуюче перетворення на основі десяткового логарифму та інтервальні оцінки, отримаємо більш достовірну оцінку кількості дефектів ПЗ в порівнянні з існуючими методами.

2 УДОСКОНАЛЕННЯ ОДНОФАКТОРНОЇ РЕГРЕСІЙНОЇ МОДЕЛІ ДЛЯ ОЦІНЮВАННЯ КІЛЬКОСТІ ДЕФЕКТІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Розробка удосконаленої однофакторної регресійної моделі для оцінювання кількості дефектів програмного забезпечення із застосуванням нормалізуючого перетвореня

При наявності вихідних ЕД закон розподілу яких не є нормальним, побудова рівняння регресії без припущення про нормальність ВВ може бути ускладнена, і спотворити результати оцінювання. Тому перед тим, як будувати рівняння регресії, потрібно нормалізувати вихідні ЕД. В результаті нормалізуючого перетворення отримаємо гаусівські ВВ які дозволяють перейти до лінійного рівняння регресії, та для нього побудувати довірчий інтервал та інтервал прогнозування, і шляхом застосування зворотного перетворення перейти до нелінійного рівняння регресії і для нього побудувати довірчий інтервал та інтервал прогнозування [13]. Для нормалізації негаусівських даних використаємо нормалізуюче перетворення на основі десяткового логарифму:

$$z = \log_{10} x; \tag{2.1}$$

де z – нормована нормально розподілена ВВ;

x – ВВ, яка нормалізується.

Перетворення (2.1) має зворотне перетворення:

$$x = 10^{z}. \tag{2.2}$$

Таким чином, можемо побудувати нелінійну регресійну модель, використовуючи лінійну регресійну модель (1.2) та зворотне нормалізуюче перетворення (2.2) за допомогою наступної формули:

$$y = x^{b_1} \cdot 10^{b_0 + \varepsilon}. \tag{2.3}$$

Перевірку на нормальний закон розподілу можна виконати за допомогою критеріїв згоди χ^2 Пірсона або Колмогорова – Смирнова [14]. Використаємо критерій χ^2 Пірсона, тому що він дозволяє зіставляти емпіричний розподіл з теоретичним.

Критерій χ^2 Пірсона можна використовувати для перевірки гіпотез різних розподілів (рівномірного, нормального або іншого), тому що він дозволяє порівнювати розподіл частот незалежно від розподілу.

Для перевірки необхідно порівнювати емпіричні і теоретичні частоти. При повному збігу емпіричних частот з частотами, обчисленими або очікуваними критерій χ^2 буде дорівнює нулю. Якщо ж χ^2 не дорівнює нулю – це вкаже на невідповідність обчислених частот емпіричним частотам ряду. У таких випадках необхідно оцінити значимість критерію χ^2, який теоретично може змінюватися від нуля до нескінченності. Це проводиться шляхом порівняння фактично отриманої величини χ^2 з його критичним значенням. Нульова гіпотеза, тобто припущення, що розбіжність між емпіричними і теоретичними або очікуваними частотами носить випадковий характер, спростовується, якщо χ^2 більше або дорівнює критичному значенню χ^2 для прийнятого рівня значущості *a* і числа ступенів свободи *n*.

Розрахувати критерій χ^2 Пірсона можна за допомогою наступної формули [15]:

$$\chi^2 = \sum_{i=1}^{m} \frac{(n_i - np_i)^2}{np_i}, \tag{2.4}$$

де m – кількість підінтервалів, залежить від обсягу вибірки n;

n_i – кількість значень випадкової величини, що потрапили в i-й підінтервал;

p_i – ймовірність потрапляння випадкової величини в i-й підінтервал відповідно з гіпотетичним законом розподілу ВВ.

Кількість підінтервалів m можна розрахувати в залежності від обсягу вибірки n.

95% довірчий інтервал нелінійного рівняння регресії можна побудувати, використовуючи побудований 95% довірчий інтервал лінійного рівняння регресії за формулою (1.4) та зворотне нормалізуюче перетворення за формулою (2.2).

Аналогічно знайдемо і 95% інтервал прогнозування нелінійного рівняння регресії, використовуючи побудований 95% інтервал прогнозування лінійного рівняння регресії за формулою (1.5) та зворотне нормалізуюче перетворення за формулою (2.2).

Якість отриманого рівняння регресії можна перевірити за допомогою коефіцієнта детермінації R^2, середньої величини відносної похибки *MMRE* та рівня прогнозування *PRED*(0,25).

Коефіцієнт детермінації R^2 можна знайти за допомогою наступної формули [16]:

$$R^2 = 1 - \left(\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \right), \qquad (2.5)$$

де y_i – емпіричне значення y;

$\hat{y}_i$ – розрахункове значення y;

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

Середню величину відносної похибки *MMRE* можна знайти за допомогою наступної формули:

$$MMRE = \frac{1}{N}\sum_{i=1}^{N} MRE_i \,, \tag{2.6}$$

де n – кількість значень у вибірці;

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

Рівень прогнозування *PRED*(0,25) можна знайти за допомогою наступної формули:

$$PRED(0{,}25) = \frac{k}{n}, \tag{2.7}$$

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

Першим кроком при обробці емпіричних даних повинна проводитися перевірка на викиди. Лише після цього можна переходити до наступного кроку аналізу даних [17, 18].

Для перевірки емпіричних даних на викиди можна використовувати квадрат відстані Махаланобіса (Mahalanobis squared distance), який можна знайти за допомогою наступної формули [19]:

$$d_i^2 = (Z_i - \bar{Z})^T S_N{}^{-1} (Z_i - \bar{Z}), \tag{2.8}$$

де $\bar{Z}$ – середній вектор вибірки,

S_N – матриця кореляції вибірки, яку можна знайти за допомогою наступної формули:

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

Після розрахунку d_i^2 можна скористатися критерієм Фишера F, що має апроксимований розподіл F з m та $N–m$ ступенями свободи. Додатково потрібно розрахувати T_S (Test statistic), яку можна знайти за допомогою наступної формули:

$$Ts = \frac{(N-m)Nd_i^2}{(N^2-1)m}. \qquad (2.9)$$

Тестова статистика T_S для квадрату відстані Махаланобіса порівнюється з квантилем розподілу F, який позначається як $F_{m,N-m,\alpha}$, де α – це рівень значущості, який у даній роботи брався рівним 0,05. Точки, для яких значення T_S, розраховане за формулою (2.9), буде перевищувати квантиль розподілу F, вважаються викидами. Ці значення потрібно видалити з набору даних.

Цей алгоритм виконується ітераційно доти, доки всі значення T_S не будуть менше або дорівнюватимуть квантилю розподілу F, тобто після усунення викидів отриманий новий набір значень нормалізується, використовуючи нормалізуючи перетворення (2.1) та для нього знову знаходиться квадрат відстані Махаланобіса за формулою (2.8) та тестова статистика за формулою (2.9).

2.2 Побудова удосконаленої однофакторної регресійної моделі для оцінювання кількості дефектів програмного забезпечення із застосуванням логарифмічного нормалізуючого перетвореня

Для перевірки удосконаленої однофакторної регресійної моделі для оцінювання кількості дефектів ПЗ із застосуванням нормалізуючого перетвореня на основі десяткового логарифму вхідні дані було отримано з досліджень національного космічного відомства США (NASA), результати яких були опубліковані в [20].

Вхідні дані представляють собою такий набір метрик:

– KSLOCs (Kilo-SLOC(Thousand Source/Software Lines of Code)) – тис. вихідних програмних рядків коду;

– Major errors – основні помилки;

– Early detection % – відсоток раннього виявлення;

– DRs (Discrepancy Reports) – звіти про невідповідність;

– Errors Rate – частота помилок;

– Verification detection – виявлення верифікації;

– Process plus product Error Rate – частота помилок в процесі та продукті разом.

Для побудови удосконаленої однофакторної регресійної моделі для оцінювання кількості дефектів ПЗ використовувалися такі метрики: KSLOCs та Major errors. Всього в [20] були представлені дані про 31 проект, однак 3 з них не мали значення метрики Major errors, отже для побудови регресійної моделі використовувалися дані про 28 проектів розробки ПЗ, які наведені в табл. 2.1, де ВВ X – KSLOCs, ВВ Y – Major errors.

Застосуємо до отриманих даних нормалізуюче перетвореня на основі десяткового логарифму згідно з (2.1). Отримані дані також наведені в табл. 2.1. Далі перевіримо отримані дані на викиди, застосовуючи квадрат відстані Махаланобіса d_i^2 та тестову статистику T_S згідно з (2.8) та (2.9) відповідно.

Значення T_S для d_i^2 також наведено в табл. 2.1. Перевірка показала, що в даному наборі даних присутні 2 викиди при F=3,369, це рядки 22 та 28. Вони були видалені з набору даних. Таким чином скоригована вибірка склала 26 наборів даних.

Таблиця 2.1 – Вихідні та нормалізовані дані

№	X, KSLOCs	Y, Major Errors	*Zx*	*Zy*	T_S для d_i^2
1	173	13634	0,6021	1,6021	0,7786
2	696	122011	1,0253	2,1703	1,1470
3	31	1940	0,9031	1,4472	1,1485
4	290	76817	1,0569	2,1673	0,9378
5	49	2496	0,7709	1,8195	0,6137
6	24	6069	1,0864	2,2455	1,3309
7	30	1898	0,9445	1,9294	0,2379
8	37	3774	0,8195	1,6021	0,1442
9	93	27485	0,7993	1,6990	0,1363
10	170	8762	0,4914	1,3979	0,9619
11	169	7371	0,8451	1,4624	0,7274
12	63	10350	1,0828	1,5051	2,1952
13	82	11973	0,2788	1,1761	2,0028
14	131	1920	1,4683	2,1430	1,3755
15	223	6858	1,3284	2,0414	0,7461
16	175	9093	1,5366	2,1239	2,0788
17	202	15929	1,3802	2,0569	1,0303
18	196	15525	1,0170	1,9085	0,0442
19	68	6926	1,1847	1,9494	0,2780
20	41	5114	0,8633	1,7782	0,0909
21	211	16849	1,0414	1,8261	0,0682
22	80	18598	1,0828	1,7853	0,2941
23	963	14809	0,8261	1,6628	0,0794
24	**59**	**5469**	**1,4281**	**2,7050**	**3,5018**
25	72	7159	1,1644	2,0212	0,1880
26	75	8971	0,8451	1,7324	0,0722
27	104	10359	0,9031	1,7853	0,0278
28	**66**	**8068**	**0,0792**	**0,7782**	**3,7959**

На рис. 2.1 та рис. 2.2 приведені гістограми емпіричного розподілу для вибірок *X* та *Y* відповідно.

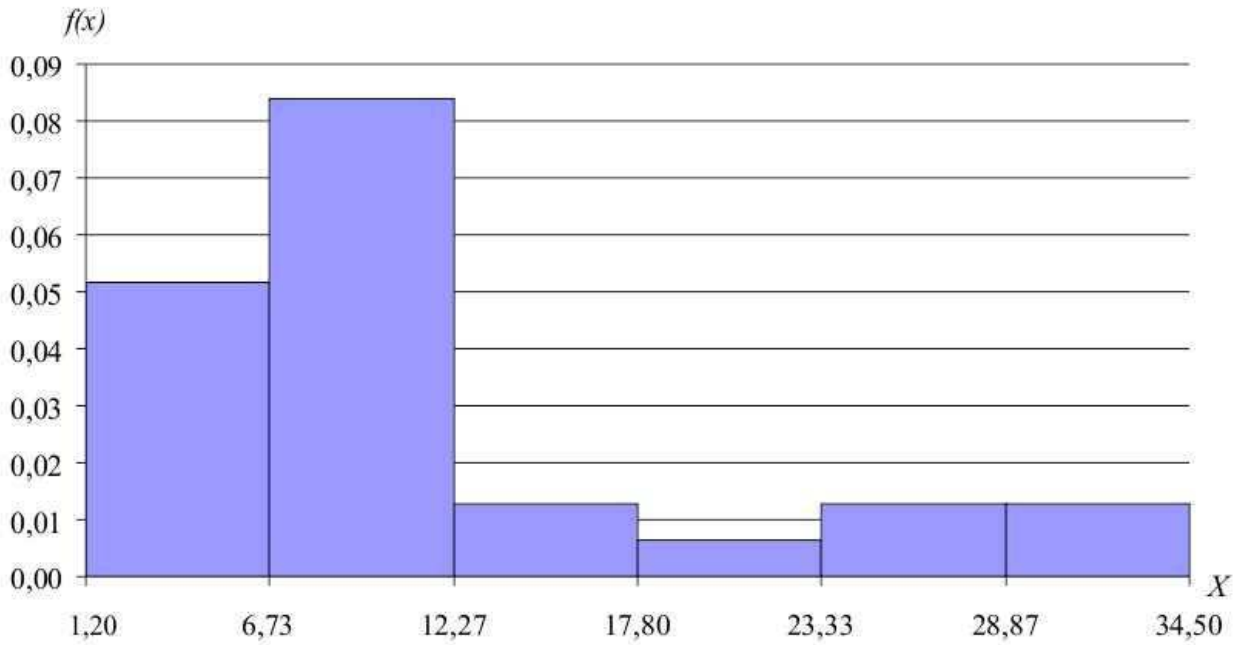

Рисунок 2.1 – Гістограма емпіричного розподілу для вибірки *X*

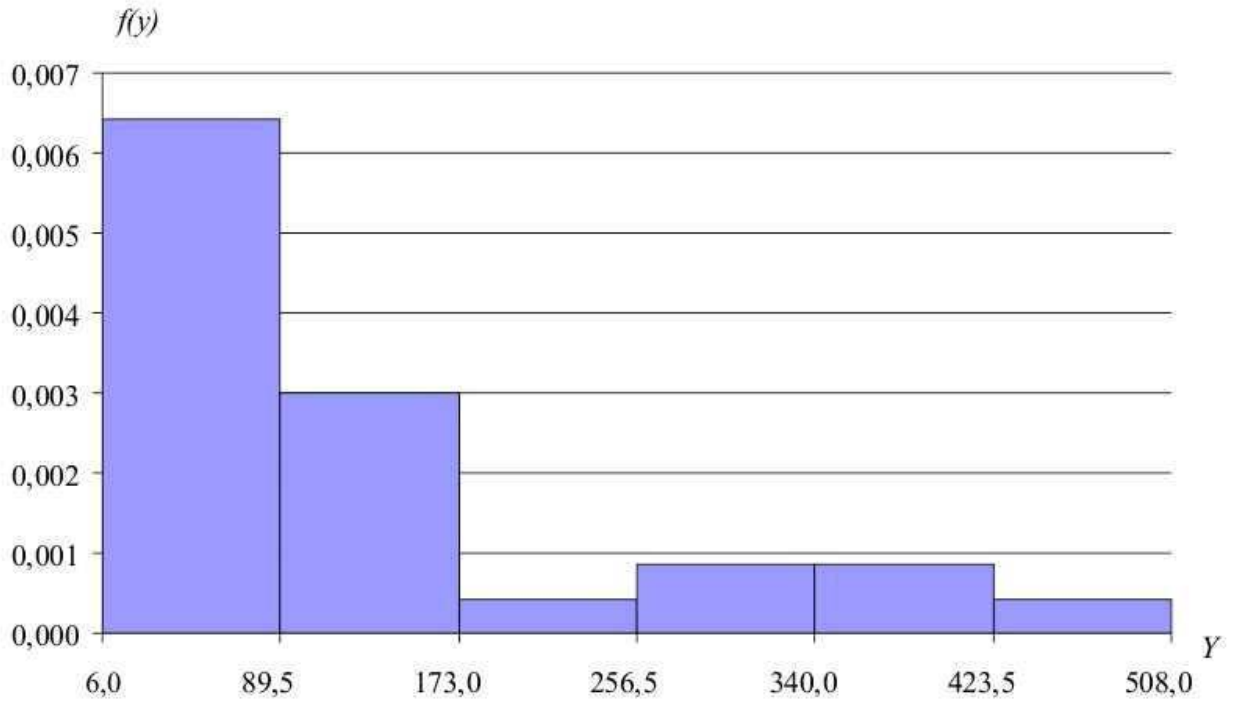

Рисунок 2.2 – Гістограма емпіричного розподілу для вибірки *Y*

З довірчою ймовірністю 0,95 гіпотеза про відповідність вибірок *X* та *Y* нормальному закону розподілу відхиляється. Для вибірки *X*: $\chi^2 = 18{,}05$, при $\chi^2_{кр} = 7{,}81$. Для вибірки *Y*: $\chi^2 = 25{,}76$, при $\chi^2_{кр} = 7{,}81$.

Застосуємо до отриманих даних нормалізуюче перетвореня на основі десяткового логарифму згідно з (2.1). На рис. 2.3 та рис. 2.4 приведені гістограми розподілу для нормалізованих вибірок Z_x та Z_y відповідно.

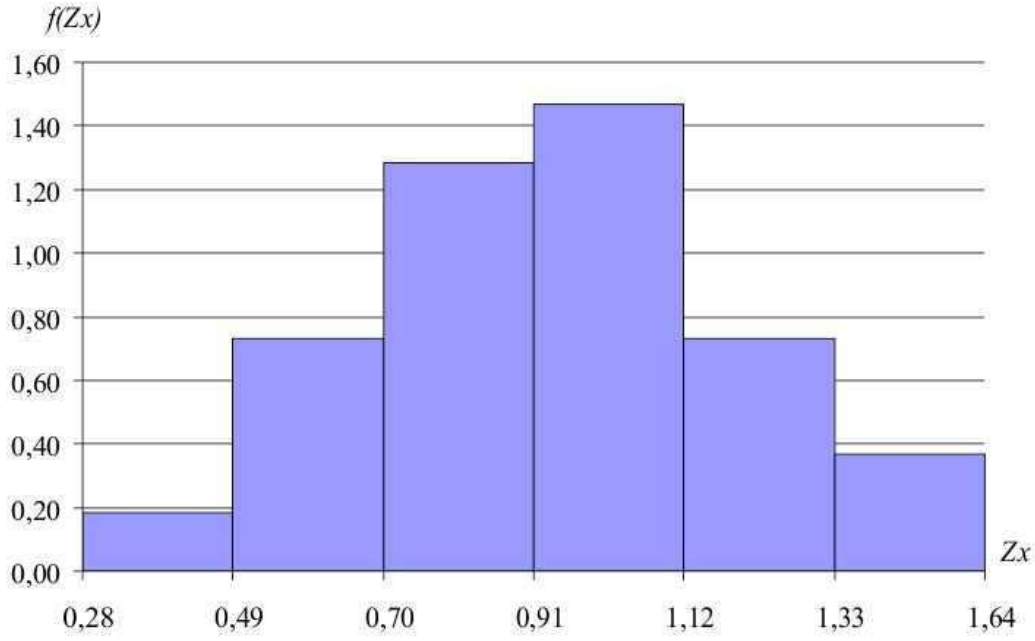

Рисунок 2.3 – Гістограма розподілу для вибірки Z_x

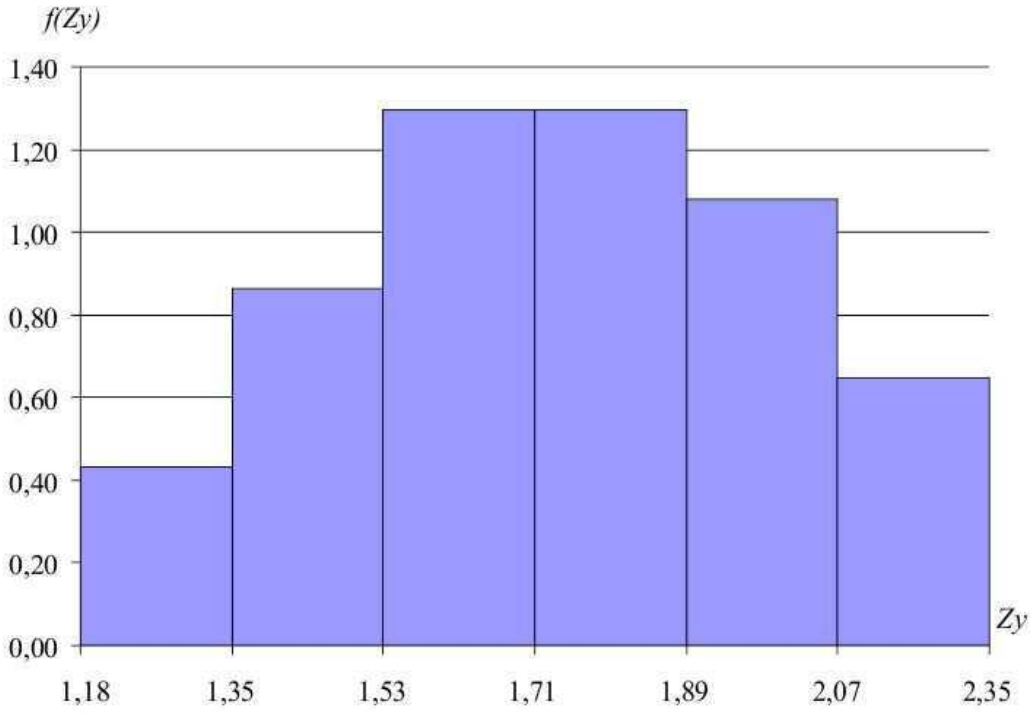

Рисунок 2.4 – Гістограма розподілу для вибірки Z_y

Перевіримо нормалізовані дані на нормальній закон розподілу. З довірчою ймовірністю 0,95 гіпотеза про відповідність нормалізованих вибірок Z_x та Z_x нормальному закону розподілу приймається. Для вибірки Z_x: $\chi^2 = 4{,}47$, при $\chi^2_{кр} = 7{,}81$. Для вибірки Z_x: $\chi^2 = 1{,}23$, при $\chi^2_{кр} = 7{,}81$.

Будуємо лінійну регресійну модель для нормалізованих даних згідно з (1.2) та за допомогою методу найменших квадратів знаходимо його коефіцієнти згідно з (1.3): $b_1 = 0{,}7602$, $b_0 = 1{,}0680$:

$$z_y = 0{,}7602 z_x + 1{,}0680 + \varepsilon .$$

Будуємо довірчий інтервал та інтервал прогнозування лінійного рівняння регресії згідно з (1.4) та (1.5) відповідно, які разом із самим рівнянням та нормалізованими даними зображені на рис. 2.5.

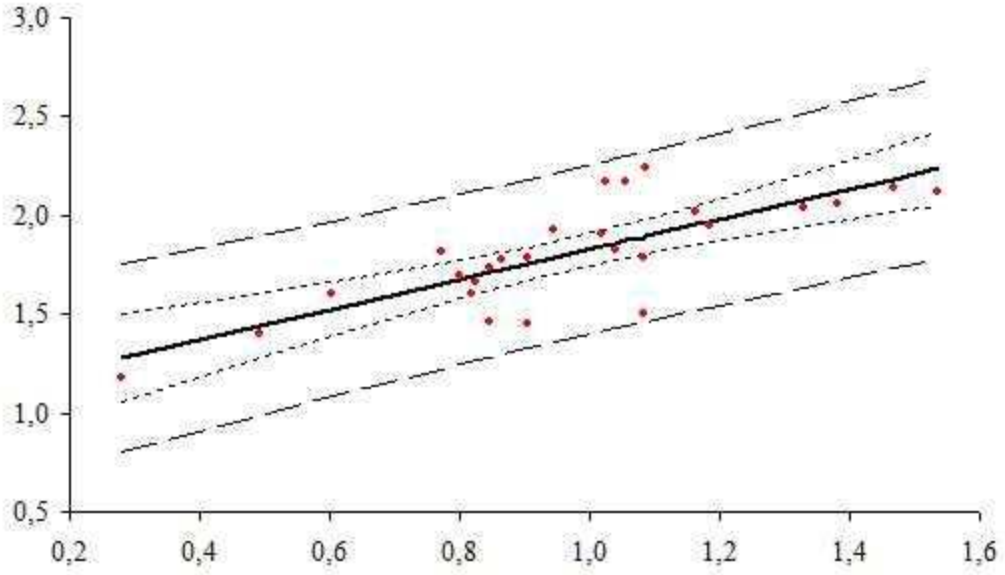

Рисунок 2.5 – Нормалізовані дані, лінійне рівняння регресії, довірчий інтервал та інтервал прогнозування

Як видно з рис. 2.5, у нормалізованому наборі даних, що розглядається, викидів немає, усі значення нормалізованих величин знаходяться всередині інтервалу прогнозування.

Переходимо до ЕД. Будуємо нелінійну регресійну модель для згідно з (2.3):

$$y = x^{0,7602} \cdot 10^{1,0680+\varepsilon}.$$

Будуємо довірчий інтервал та інтервал прогнозування нелінійного рівняння регресії згідно з (1.4) та (1.5) відповідно, та зворотним нормалізуючим перетворенням (2.2), які разом із самим рівнянням та ЕД зображені на рис. 2.6.

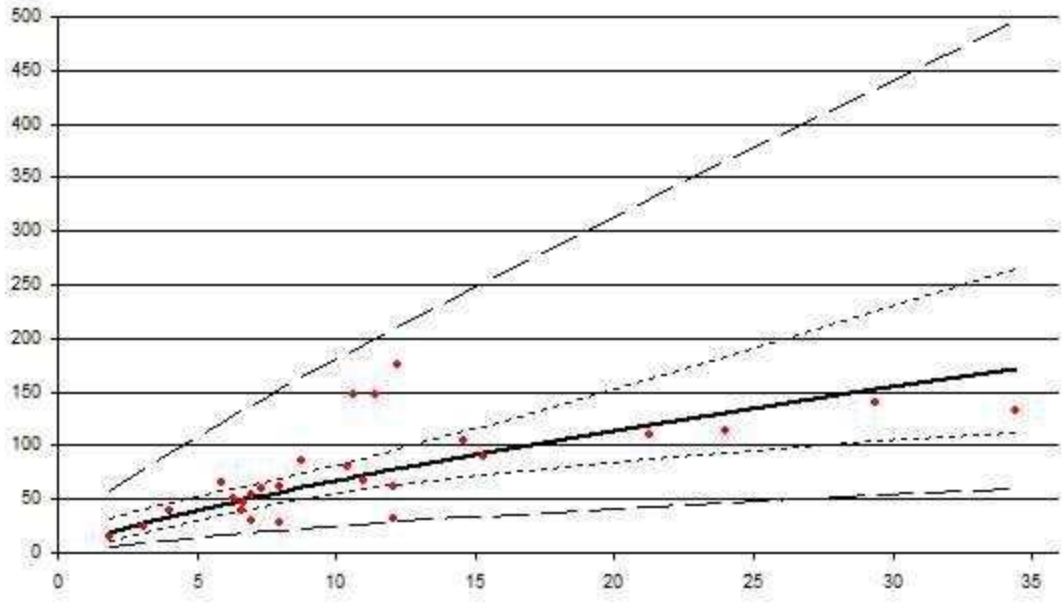

Рисунок 2.6 –ЕД, нелінійне рівняння регресії, довірчий інтервал та інтервал прогнозування

Як видно з рис. 2.6, у ЕД, що розглядаються, також викидів немає, усі значення ЕД знаходяться всередині інтервалу прогнозування.

Будуємо лінійну регресійну модель в припущенні про нормальність вихідних ЕД згідно з (1.2), коефіцієнти рівняння знаходимо за допомогою методу найменших квадратів згідно з (1.3): b1=3,7111, b0=34,2271. Лінійна регресійна модель має вигляд:

$$z_y = 3,7111 z_x + 34,2271 + \varepsilon.$$

Далі будуємо довірчий інтервал та інтервал прогнозування лінійного рівняння регресії згідно з (1.4) та (1.5) відповідно, які разом із самим рівнянням регресії та ЕД зображені на рис. 2.7.

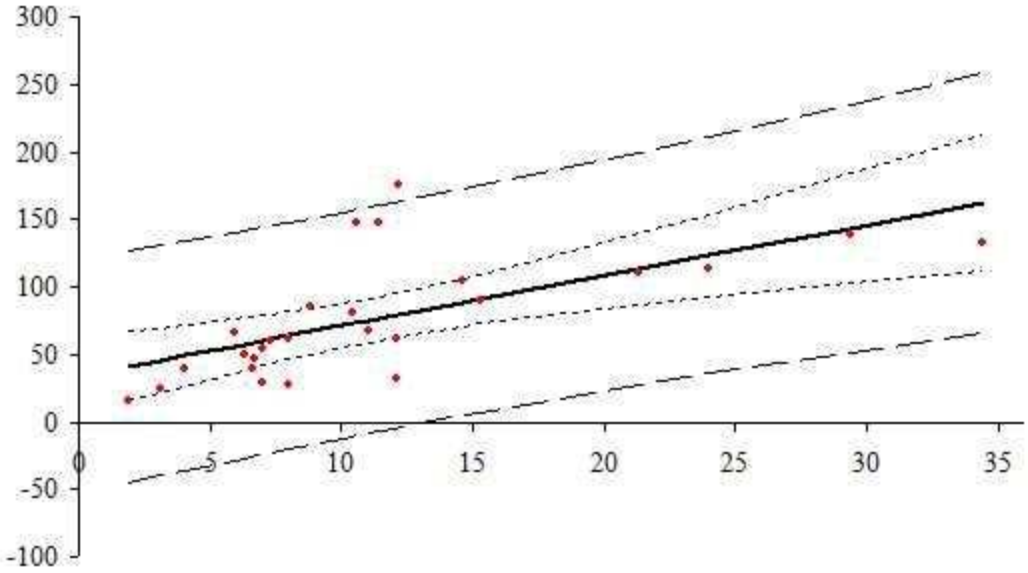

Рисунок 2.7 –ЕД, лінійне рівняння регресії, довірчий інтервал та інтервал прогнозування

Як видно з рис. 2.7, у ЕД, що розглядаються, при використанні лінійної регресії без виконання нормалізації присутні викиди. Також бачимо, що нижня границя інтервалу прогнозування має значення нижче нуля. Тому використання лінійної регресії для цих ЕД не доречно.

Щоб додатково проілюструвати це твердження, розглянемо окремо значення для границь довірчого інтервалу та інтервалу прогнозування. Результати для довірчого інтервалу наведено у табл. 2.2, а для інтервалу прогнозування – у табл. 2.3.

Як видно з табл. 2.2, усі значення нижньої границі довірчого інтервалу більші нуля, як для нелінійного рівняння з використанням логарифмічного нормалізуючого перетворення, так і для лінійного рівняння без виконання нормалізації. Однак довжина довірчого інтервалу у разі використання логарифмічного нормалізуючого перетворення приблизно у 77% значень

менша за відповідну довжину довірчого інтервалу без виконання нормалізації.

Таблиця 2.2 – Границі довірчого інтервалу рівнянь регресії

№	Нелінійне рівняння з використанням логарифмічного нормалізуючого перетворення			Лінійне рівняння без виконання нормалізації		
	Нижня границя	Верхня границя	Довжина	Нижня границя	Верхня границя	Довжина
1	2	3	4	5	6	7
1	24,43	46,09	21,66	26,58	71,56	44,98
2	58,04	85,36	27,33	57,36	89,77	32,40
3	46,71	69,14	22,43	46,23	81,60	35,37
4	61,03	90,67	29,64	60,44	92,62	32,18
5	35,63	57,05	21,42	36,21	76,04	39,83
6	63,84	96,10	32,26	63,35	95,65	32,30
7	50,48	73,96	23,48	49,82	83,94	34,12
8	39,50	61,02	21,52	39,63	77,81	38,18
9	37,86	59,32	21,45	38,17	77,04	38,87
10	18,89	40,45	21,55	21,89	69,57	47,68
11	41,64	63,31	21,67	41,55	78,86	37,31
12	63,50	95,41	31,91	63,00	95,26	32,27
13	11,42	31,77	20,35	15,53	67,03	51,50
14	103,86	224,98	121,12	102,62	184,04	81,42
15	88,02	162,62	74,60	87,25	139,30	52,05
16	112,31	264,16	151,85	111,40	212,37	100,97
17	93,67	183,21	89,54	92,63	153,96	61,33
18	57,25	84,06	26,80	56,57	89,08	32,51
19	73,35	118,01	44,66	73,08	108,94	35,86
20	43,20	65,04	21,83	42,97	79,66	36,69
21	59,56	88,00	28,44	58,93	91,17	32,25
22	63,50	95,41	31,91	63,00	95,26	32,27
23	40,04	61,59	21,55	40,11	78,07	37,96
24	71,36	112,96	41,60	71,08	105,74	34,66
25	41,64	63,31	21,67	41,55	78,86	37,31
26	46,71	69,14	22,43	46,23	81,60	35,37

Як видно з табл. 2.3, у разі побудови лінійного рівняння без виконання нормалізації нижня границя інтервалу прогнозування має від`ємні значення, на відміну від нелінійного рівняння з використанням логарифмічного нормалізуючого перетворення, приблизно у 77% випадків. Довжина інтервалу прогнозування у разі використання логарифмічного

нормалізуючого перетворення приблизно у 58% значень менша за відповідну довжину довірчого інтервалу без виконання нормалізації.

Таблиця 2.3 – Границі інтервалу прогнозування рівнянь регресії

№	Нелінійне рівняння з використанням логарифмічного нормалізуючого перетворення			Лінійне рівняння без виконання нормалізації		
	Нижня границя	Верхня границя	Довжина	Нижня границя	Верхня границя	Довжина
1	2	3	4	5	6	7
1	12,12	92,92	80,81	-35,99	134,13	170,11
2	26,23	188,85	162,62	-10,05	157,18	167,23
3	21,17	152,57	131,41	-20,00	147,83	167,83
4	27,70	199,79	172,09	-7,06	160,13	167,19
5	16,65	122,09	105,44	-28,29	140,54	168,83
6	29,12	210,64	181,52	-4,10	163,11	167,21
7	22,78	163,88	141,10	-16,90	150,67	167,57
8	18,21	132,41	114,20	-25,50	142,94	168,44
9	17,55	128,01	110,46	-26,69	141,91	168,60
10	9,77	78,21	68,44	-39,69	131,16	170,85
11	19,07	138,23	119,16	-23,92	144,33	168,25
12	28,95	209,29	180,35	-4,47	162,73	167,20
13	6,38	56,94	50,56	-44,70	127,25	171,95
14	53,91	433,45	379,54	51,76	234,91	183,15
15	43,34	330,28	286,94	27,21	199,33	172,12
16	59,78	496,27	436,49	65,57	258,21	192,64
17	47,03	364,91	317,88	35,72	210,87	175,15
18	25,86	186,11	160,25	-10,80	156,45	167,25
19	34,34	252,07	217,73	7,04	174,97	167,93
20	19,71	142,56	122,85	-22,74	145,37	168,11
21	26,97	194,33	167,36	-8,55	158,65	167,20
22	28,95	209,29	180,35	-4,47	162,73	167,20
23	18,42	133,87	115,44	-25,11	143,29	168,39
24	33,20	242,79	209,59	4,57	172,25	167,68
25	19,07	138,23	119,16	-23,92	144,33	168,25
26	21,17	152,57	131,41	-20,00	147,83	167,83

Перевіримо якість отриманих рівнянь регресії, для чого використаємо коефіцієнт детермінації R^2 згідно з (2.5), середню величину відносної похибки *MMRE* згідно з (2.6) та рівен прогнозування *PRED*(0,25) згідно з (2.7). Зведені дані представлені в табл. 2.4.

Таблиця 2.4 – Перевірка якості регресійних моделей

Параметр	Нелінійна модель з використанням логарифмічного нормалізуючого перетворення	Лінійна модель без виконання нормалізації
R^2	0,6271	0,4287
MMRE	0,2522	0,4108
Pred(0,25)	0,7369	0,5769

Як видно з табл. 2.4, рівняння, побудоване із використанням логарифмічного нормалізуючого перетворення, має кращі значення наведених параметрів. Однак значення параметрів *MMRE* та *Pred*(0,25) гірші за допустимі, що показує необхідність спробувати інші нормалізуючі перетворення для нормалізації ЕД.

3 ПРОЕКТ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ КІЛЬКОСТІ ДЕФЕКТІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Ескізний проект програмного забезпечення

3.1.1 Вибір мови моделювання

Моделювання часто використовується при розробці складних інженерних систем. Велика кількість програм нині не поступається своєю складністю інженерним спорудам, що дає підставу на моделювання програмних систем.

Тому для відображення процесу створення ПЗ та для того, щоб простежити послідовність трансформації від початкової моделі до завершеного програмного продукту було обрано мову моделювання UML [21].

UML – це мова, призначена для візуалізації, специфікації, конструювання й документування програмних систем. Крім того її можна використати для моделювання різного роду програм. Це уніфікована графічна мова моделювання для опису, візуалізації, проектування та документування об’єктно-орієнтованих систем. UML покликаний підтримувати процес моделювання ПЗ на основі об’єктно-орієнтованого підходу, організовувати взаємозв'язок концептуальних і програмних понять, відображати проблеми масштабування складних систем.

Мова UML призначена для вирішення наступних завдань:

– надати користувачеві мову візуального моделювання, яка легко сприймається, спеціально призначена для розробки і документування моделей складних систем самого різного цільового призначення;

– забезпечити вихідні поняття мови UML можливістю розширення і спеціалізації для більш точного уявлення моделей системи в ООАП (об'єктно-орієнтований аналіз і проектування) конкретної предметної області;

– заохочувати розвиток ринку об'єктних інструментальних засобів;

– інтегрувати в себе новітні і найкращі досягнення практики.

3.1.2 Побудова діаграми варіантів використання

Мета розробки діаграм наступна:

– визначити загальні межі і предметну область;

– сформулювати загальні вимоги до функціональної поведінки проектованої системи;

– розробити початкову концептуальну модель системи її подальшої деталізації у формі логічних і фізичних моделей;

– підготувати початкову документацію для взаємодії розробників системи з замовниками і користувачами.

Для того, щоб більш точно зрозуміти, як повинна працювати система, все частіше використовується опис функціональності системи через варіанти використання – Use Case або прецеденти.

Прецеденти це опис послідовності дій, які може здійснювати система у відповідь на зовнішні дії користувачів або інших програмних систем. Прецеденти відображають функціональність системи.

Обов’язковими елементами є: варіанти використання або прецедент (use case), актор або дійова особа (actor) і відносини між акторами і варіантами використання (relationship).

Актором називається деякий об'єкт, суб'єкт чи система, яка взаємодіє з модельованої бізнес-системою з метою досягнення своїх цілей чи вирішення певних завдань. Це може бути людина, технічний пристрій, програма або будь-яка інша система, яка служить джерелом впливу на модельовану

систему [22].

В UML є декілька типів відносин між акторами й варіантами використання:

– асоціації (association relationship);
– включення (include relationship);
– розширення (extend relationship);
– узагальнення (generalization relationship).

Для графічного відображення майбутньої програми для оцінювання кількості дефектів ПЗ та для фіксації вимог до програми було виконано побудову UML діаграми, а саме – діаграми варіантів використання, яка представлена на рис. 3.1.

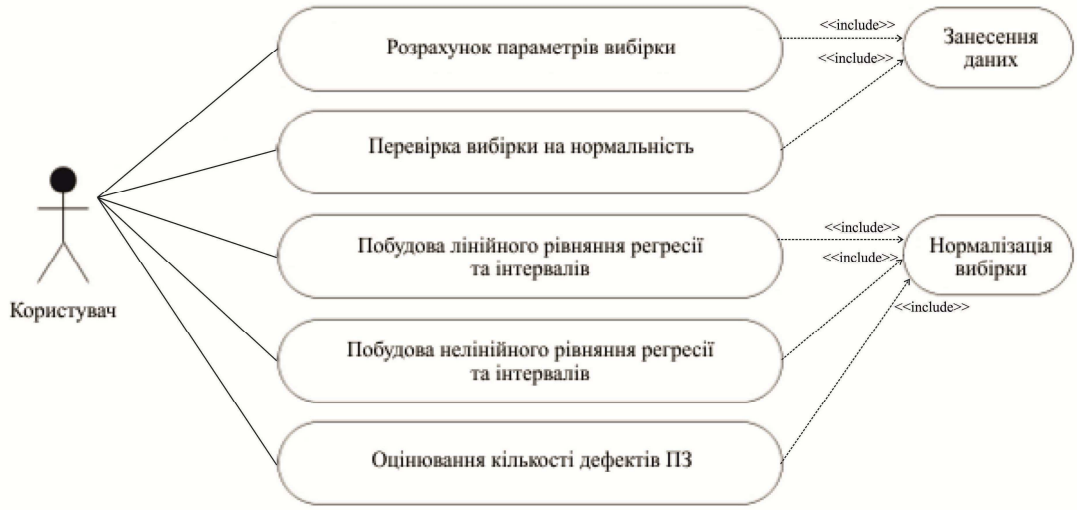

Рисунок 3.1 – Діаграма варіантів використання

Застосування варіантів використання на всіх рівнях діаграми дозволяє не тільки досягти необхідного рівня уніфікації позначень для подання функціональності підсистем і системи в цілому, але і є потужним засобом послідовного уточнення вимог до проектованої системи на основі спуску від пакетів системи до операцій класів. З іншого боку, модифікація окремих операцій класу може надати зворотний вплив на уточнення сервісу відповідного варіанта використання, тобто реалізувати ефект зворотного

зв'язку з метою уточнення специфікацій або вимог на рівні пакетів системи [23].

3.1.3 Розробка специфікації варіантів використання

Варіанти використання, представлені на рис. 3.1, конкретизовані у табл. 3.1.

Таблиця 3.1 – Конкретизовані варіанти використання

Основні актори	Найменування	Формулювання
Користувач	Занесення даних	Заносить інформації про кількість рядків коду та кількість дефектів ПЗ
Користувач	Розрахунок параметрів вибірки	Розраховує параметри вибірки для занесених даних
Користувач	Перевірка вибірки на нормальність	Перевіряє вибірки на відповідність нормальному закону розподілу
Користувач	Нормалізація вибірки	Нормалізує вибірки за допомогою логарифмічного нормалізуючого перетворення
Користувач	Побудова лінійного рівняння регресії та інтервалів	Будує лінійне рівняння регресії для оцінювання кількості дефектів ПЗ, а також довірчий інтервал та інтервал прогнозування лінійного рівняння регресії
Користувач	Побудова нелінійного рівняння регресії та інтервалів	Будує нелінійне рівняння регресії для оцінювання кількості дефектів ПЗ, а також довірчий інтервал та інтервал прогнозування нелінійного рівняння регресії
Користувач	Оцінювання кількості дефектів ПЗ	Виконує оцінювання кількості дефектів ПЗ в залежності від кількості рядків коду

Потік подій варіанта використання має такі складові: короткий опис; передумови; основний потік подій; альтернативний потік подій; постумови.

Ці складові частини для кожного варіанту використання приведені у табл. 3.2 – 3.8.

Таблиця 3.2 – Опис варіанту використання «Занесення даних»

Складова	Опис
Короткий опис	Відображає варіанти операції і дає можливість внести інформацію з файлу у відповідні поля.
Передумови	Запуск програми.
Основний потік подій	Актор вибирає файл з даними вибірки і імпортує їх для подальшої обробки.
Альтернативний потік подій	Якщо файл має не вірний формат, то система повідомить про помилку.
Постумови	Значення кількості рядків коду та кількості помилок повинно завжди бути додатнім.

Таблиця 3.3 – Опис варіанту використання «Розрахунок параметрів вибірки»

Складова	Опис
Короткий опис	Розрахунок параметрів вибірки (емпіричних даних чи нормалізованої).
Передумови	Даний прецедент буде доступний після успішного імпорту вибірки з файлу.
Основний потік подій	Прецедент розраховує основні параметри вибірки.
Альтернативний потік подій	-
Постумови	-

Таблиця 3.4 – Опис варіанту використання «Перевірка вибірки на нормальність»

Складова	Опис
1	2
Короткий опис	Перевірка вибірки (емпіричних даних чи нормалізованої) на відповідність нормальному закону розподілу.
Передумови	Даний варіант використання буде доступний після успішного імпорту вибірки з файлу або нормалізації вибірки.

Продовження табл. 3.4

1	2
Основний потік подій	Прецедент перевіряє вибірку (емпіричних даних чи нормалізовану) на відповідність нормальному закону розподілу.
Альтернативний потік подій	-
Постумови	-

Таблиця 3.5 – Опис варіанту використання «Нормалізація вибірки»

Складова	Опис
Короткий опис	Виконання нормалізації вибірки ЕД
Передумови	Даний варіант використання буде доступний після успішного імпорту вибірки з файлу
Основний потік подій	Прецедент виконує нормалізацію вибірки ЕД за допомогою десяткового логарифму.
Альтернативний потік подій	-
Постумови	-

Таблиця 3.6 – Опис варіанту використання «Побудова лінійного рівняння регресії та інтервалів»

Складова	Опис
Короткий опис	Побудова лінійного рівняння регресії та довірчого інтервалу і інтервалу прогнозування.
Передумови	Даний прецедент буде доступний після успішної нормалізації вибірки ЕД.
Основний потік подій	Прецедент розраховує коефіцієнти лінійного рівняння регресії, довірчий інтервал та інтервал прогнозування лінійного рівняння регресії.
Альтернативний потік подій	-
Постумови	-

Таблиця 3.7 – Опис варіанту використання «Побудова нелінійного рівняння регресії та інтервалів»

Складова	Опис
Короткий опис	Побудова нелінійного рівняння регресії та довірчого інтервалу і інтервалу прогнозування.
Передумови	Даний прецедент буде доступний після успішної нормалізації вибірки ЕД.
Основний потік подій	Прецедент будує нелінійне рівняння регресії, довірчий інтервал та інтервал прогнозування нелінійного рівняння регресії.
Альтернативний потік подій	-
Постумови	-

Таблиця 3.8 – Опис варіанту використання «Оцінювання кількості дефектів ПЗ»

Складова	Опис
Короткий опис	Отримати значення за рівнянням регресії та інтервали по значенню фактора.
Передумови	Даний прецедент буде доступний після побудова нелінійного рівняння регресії та відповідних інтервалів
Основний потік подій	Прецедент дозволяє отримати оцінку кількості дефектів ПЗ за допомогою нелінійної регресійної моделі, яка будується на основі лінійної регресійної моделі та зворотного нормалізую чого перетворення.
Альтернативний потік подій	-
Постумови	-

3.1.4 Побудова діаграми діяльності

Поведінка об'єкта відбивається в його станах, щоб зобразити це, використовується два види діаграм: Statechart diagram (діаграма станів) і Activity diagram (діаграма активності) [24].

На рис. 3.2 – 3.3 зображено діаграми діяльності для варіантів використання «Занесення даних» та «Нормалізація вибірки».

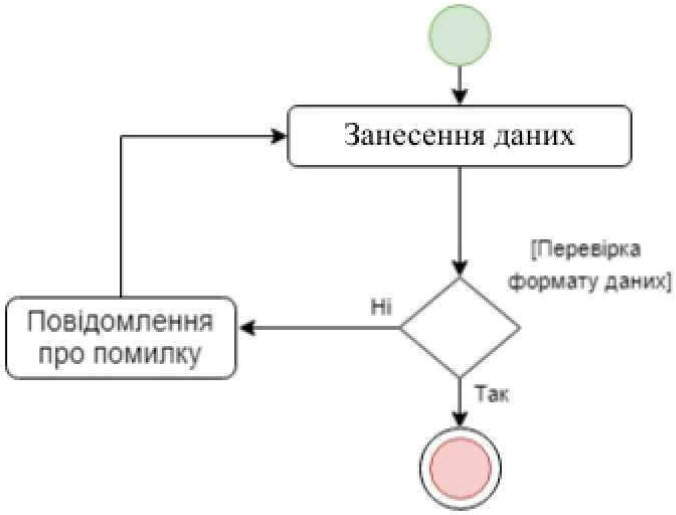

Рисунок 3.2 – Діаграма діяльності для варіанту використання «Занесення даних»

Рисунок 3.3 – Діаграма діяльності для варіанту використання «Нормалізація вибірки»

Дані діаграми діяльності добре відображають послідовність дій та події, що ініціюють дії або є їх кінцевим результатом.

3.1.5 Розробка прототипу інтерфейсу користувача

Користувальницький інтерфейс – це сукупність програмних і апаратних засобів, що забезпечують взаємодію користувача з комп'ютером. Основу такої взаємодії становлять діалоги.

Оскільки якість процесу інтерактивної взаємодії користувача із системою – швидкість, зручність, низький рівень втоми – пов’язана з такими психологічними характеристиками людини як короткострокова та середньострокова пам’ять, час реакції, можливість сприйняття візуальної інформації, то при розробці інтерфейсу користувача необхідно, щоб він задовольняв основні властивості, такі, як адаптованість, юзабіліті або зручність користування, достатність, дружність, гнучкість.

Схема екранних форм програми для оцінювання кількості дефектів програмного забезпечення приведена на рис. 3.4.

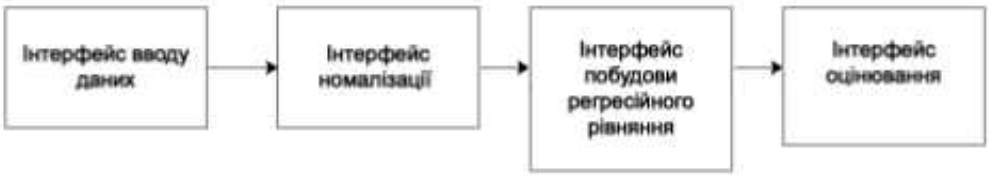

Рисунок 3.4 – Схема екранних форм

З урахуванням всіх властивостей, наведених вище, було спроектовано ескізи інтерфейсу користувача майбутньої програми, які зображені на рисунках 3.5 – 3.6.

Побудовані ескізи форм програми допоможуть задовольнити основні потреби користувача.

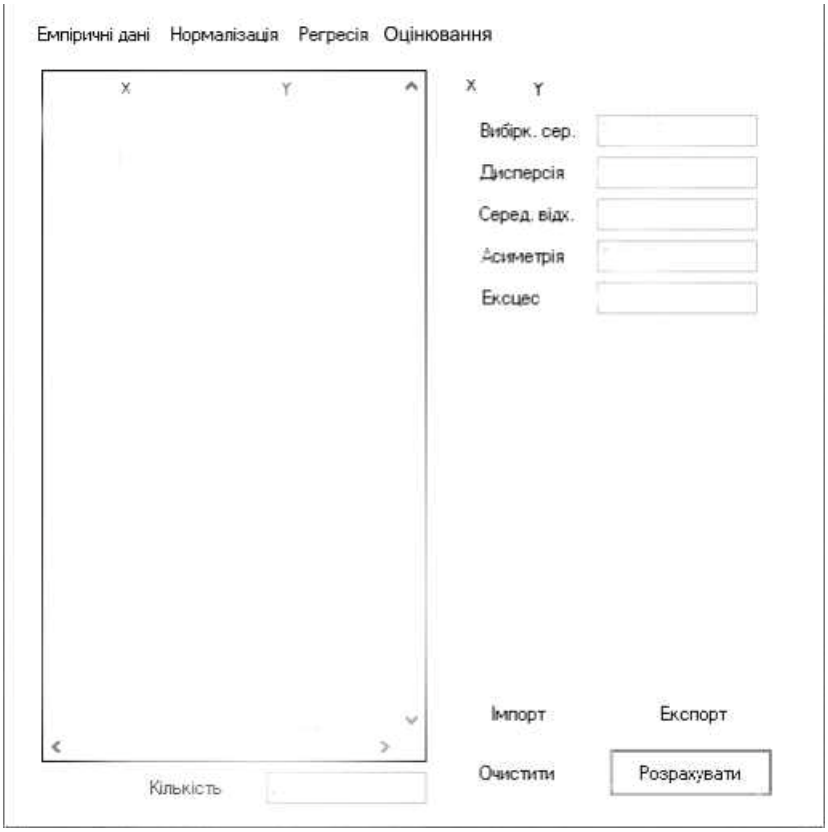

Рисунок 3.5 – Ескіз форми розрахунків вихідних даних

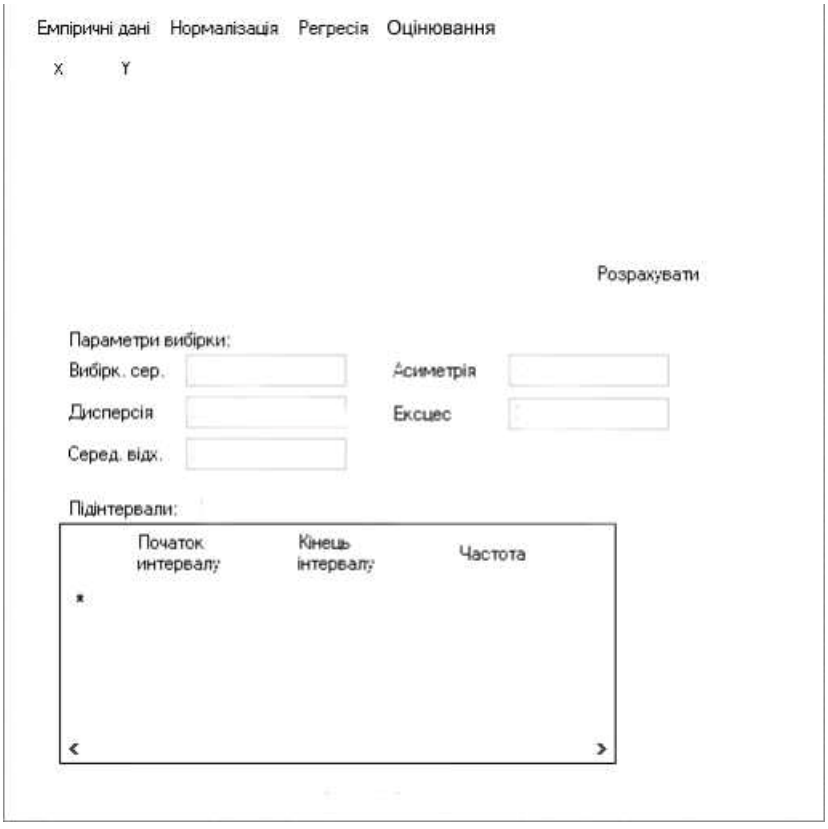

Рисунок 3.6 – Ескіз форми нормалізації даних

3.2 Технічний проект програмного забезпечення

3.2.1 Спосіб зберігання даних

У випадку даної програми відсутня база даних через те що відсутня необхідність у довгостроковому зберіганні результатів розрахунків. Початкові дані вибірки та результати розрахунків зберігаються у файлі формату CSV.

CSV-файли (значення, розділені комами) зазвичай використовуються для обміну табличними даними між системами в звичайному тексті. В основному вони містять рядок заголовка, який надає імена стовпців для даних, але в іншому вони вважаються частково структурованими. Це пов'язано з тим, що CSV-файли з самого початку не можуть представляти ієрархічні або реляційні дані. Зв'язки між даними зазвичай обробляються з використанням декількох CSV-файлів. Зовнішні ключі зберігаються в одному або декількох файлах, однак зв'язки між цими файлами не обробляються самим форматом.

Файли в форматі CSV можуть використовувати інші роздільники, крім ком, наприклад символи табуляції або пробілу.

Незважаючи на обмеження, CSV-файли є популярним вибором для обміну даними, тому що вони підтримуються широкою низкою бізнес-додатків, споживчих та наукових програм.

Наприклад, програми баз даних і електронних таблиць можуть імпортувати і експортувати CSV-файли. Аналогічним чином більшість модулів обробки пакетних і потокових даних (наприклад, Spark і Hadoop) відразу підтримують серіалізацію і десеріалізацію CSV-файлів і пропонують способи застосування схеми при читанні. Вони спрощують роботу з даними, надаючи варіанти виконання запиту і зберігаючи відомості в більш ефективному форматі для швидкої обробки.

Основною перевагою CSV формату є розмір, при передачі великого об’єму інформації CSV формати можуть зберегти близько половини від розміру даних, що зберігаються у інших популярних форматах передачі даних.

Враховуючи переваги і недоліки формату CSV, можна сказати що він відмінно підходить для розроблюваної програми, адже він має просту структуру і значно спростить процес інтеграції у разі включення розроблюваної програми в інший програмний комплекс.

3.2.2 Статична модель програмного забезпечення

Діаграма класів – статичне представлення структури моделі – діаграма, на якій представлена сукупність статичних елементів моделі, таких як класи з атрибутами й операціями, а також відношення, що їх з’єднують.

Діаграма класів призначена для подання статичної структури моделі системи в термінології класів об’єктно-орієнтованого програмування. Діаграма класів відображає різні взаємозв'язки між окремими сутностями предметної області, такими як об'єкти й підсистеми, а також описує їхню внутрішню структуру й типи відносин. На даній діаграмі не вказується інформація про часові аспекти функціонування системи.

Діаграми класів для варіантів використання «Занесення даних» та «Нормалізація вибірки» представлені на рис. 3.7 – 3.8. Ці діаграми класів демонструють певні класи системи, їх атрибути, методи і взаємозв'язки між ними.

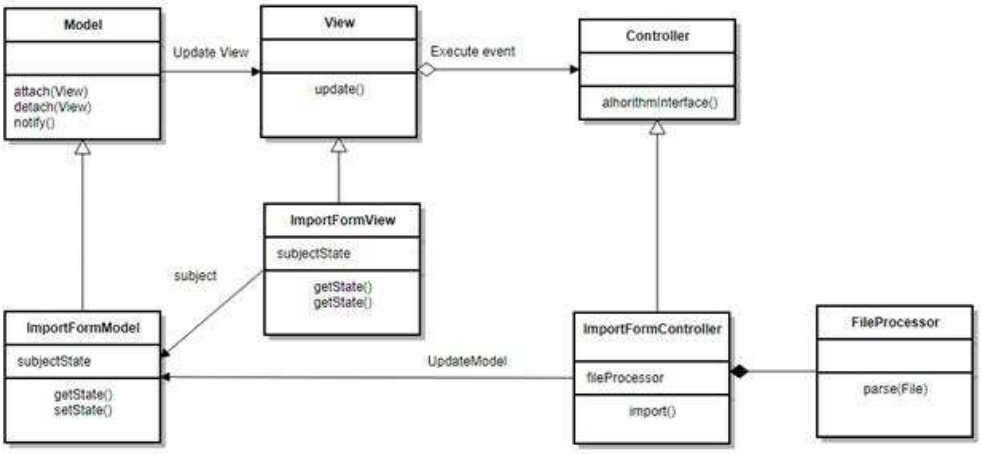

Рисунок 3.7 – Діаграма класів для варіанту використання «Занесення даних»

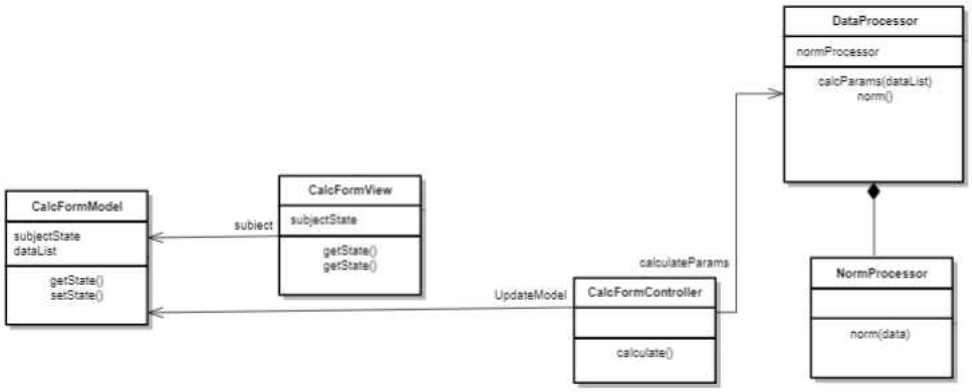

Рисунок 3.8 – Діаграма класів для варіанту використання «Нормалізація вибірки»

3.2.3 Динамічна модель програмного забезпечення

Діаграма послідовності – динамічне представлення структури моделі – діаграма, на якій показані взаємодії об’єктів, упорядковані за часом їхнього прояву. На діаграмі послідовності присутня вісь часу, що дозволяє візуалізувати відношення між переданими повідомленнями.

На діаграмі послідовностей можна побачити, що відбувається в системі поза очима користувача. Коли користувач відкриває вікно, відображення надсилає запит до контролера для отримання даних, контролер в свою чергу

надсилає запит до моделі. Ці дані надходять до відображення пройшовши обробку і відображаються користувачу.

Лінія життя об’єкта – вертикальна лінія на діаграмі послідовності, що представляє існування об’єкта протягом певного періоду часу. Лінія життя об’єкта зображується пунктирною вертикальною лінією. Якщо об’єкт існує в системі постійно, то і його лінія життя триває по всій робочій області діаграми послідовності від самої верхньої її частини до самої нижньої. Окремі об’єкти, закінчивши виконання своїх операцій, можуть бути знищені.

Діаграми послідовностей для варіантів використання «Занесення даних» та «Нормалізація вибірки» представлені на рис. 3.9 – 3.10.

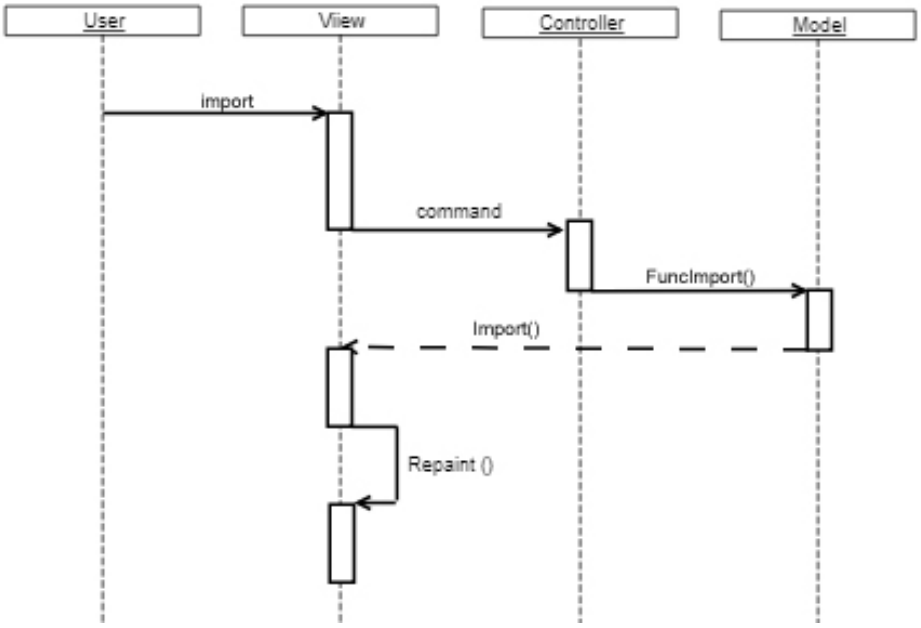

Рисунок 3.9 – Діаграма послідовності для варіанту використання «Занесення даних»

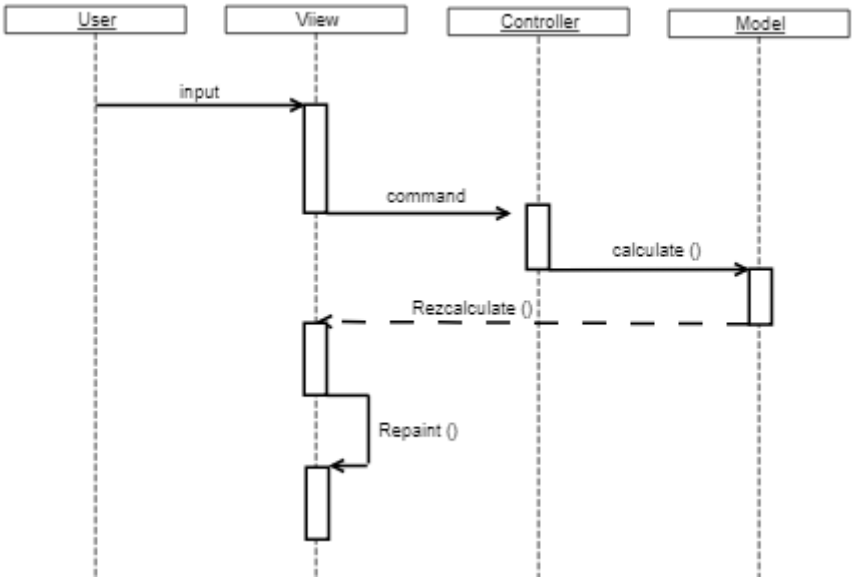

Рисунок 3.10 – Діаграма послідовності для варіанту використання «Нормалізація вибірки»

Як правило, кожна окрема діаграма послідовності описує поведінку тільки в межах одного варіанта використання. На такій діаграмі прийнято відображати екземпляри об'єктів та повідомлення, якими ці об'єкти обмінюються один з одним в рамках даного варіанта використання.

3.3 Робочий проект програмного забезпечення

3.3.1 Обґрунтування вибору мови програмування

Для розробки програмного забезпечення обрано об'єктно-орієнтовану мову програмування C#.

C# – це мова програмування створена спеціально для роботи у середовищі Microsoft .NET Framework [25].

Мова C# була розроблена з урахуванням сильних і слабких особливостей інших мов, зокрема Java і C++. На сьогоднішній момент мова

програмування C# один з найпотужніших, швидко розвиваються і затребуваних мов.

Ключові особливості мови C#:

– компонентна орієнтованість;
– код зібраний воєдино (декларації і реалізації об'єднані разом);
– уніфікована система типів і їх безпечність;
– автоматична і мануальна робота з пам'яттю;
– використання єдиної бібліотеки класів – CLR.

Мова C# розроблена насамперед для платформи .NET, яка є середовищем, що об'єднує програмні технології, для розробки Web і Windows-додатків.

Гнучкість мови C # є величезною перевагою, в порівнянні з деякими мовами програмування. Різноманітність додатків, які можуть бути розроблені за допомогою C #, і Visual Studio практично безмежне:

- додатки для Windows;
- мобільні додатки;
- веб-додатки;
- ігри;
- додатки для Android і iOS, які розробляються за допомогою додаткових фреймворків, таких як Xamarin або Mono.

Звичайно, всі ці речі можливо виконувати і за допомогою інших мов програмування, але зазвичай, в таких випадках, використовуються сторонні інструменти інших розробників. Програмісти, які працюють з C# мають згуртований набір інструментів, які підтримуються Microsoft для розробки будь-якого типу програми.

Microsoft Visual Studio дозволяє створювати додатки, що працюють на платформі .NET. Особливість цієї платформи полягає в широкому наборі сервісів, які доступні в різних мовах програмування. При цьому сервіси реалізуються у вигляді проміжного коду, який не залежить від базової архітектури. Чи не головною метою створення такої платформи було

оснащення розробників спеціальними сервісно-орієнтованими програмами, які могли б працювати на будь-якій платформі, починаючи від персонального комп'ютера і закінчуючи мобільним пристроєм.

3.3.2 Побудова діаграми компонентів

На основі проведеного проектування побудуємо діаграму компонентів програми, яка представлена на рис. 3.11.

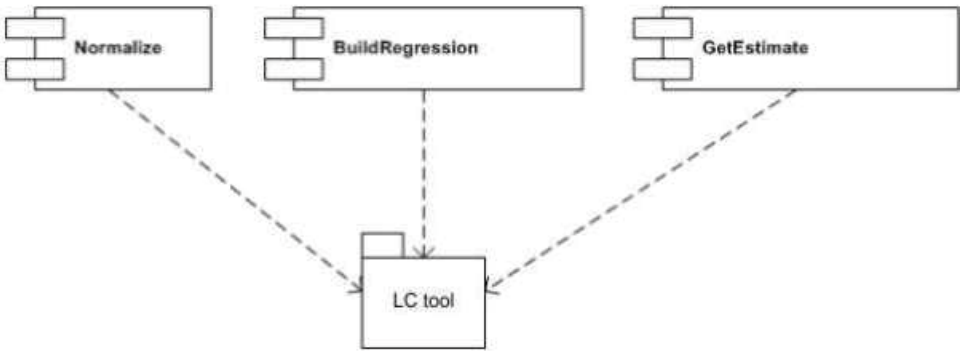

Рисунок 3.11 – Діаграма компонентів програми

Компонент Normalize виконує нормалізацію внесених даних про кількість дефектів ПЗ.

Компонент BuildRegression будує лінійну регресійну модель на основі нормалізованих даних та нелінійну регресійну модель ЕД, а також довірчий інтервал та інтервал передбачення для них.

Компонент GetEstimate виконує оцінювання кількості дефектів програмного забезпечення за допомогою нелінійного рівняння регресії на основі логарифмічного нормалізуючого перетворення.

3.3.3 Випробування програмного забезпечення

Під час випробування було проведено повне функціональне тестування, а також навантажувальне тестування і тестування на відмову всього програмно-апаратного комплексу.

Всі виявлені недоліки програми були зафіксовані в протоколах і усунуті до моменту впровадження програми у дію.

Програму і методику випробувань наведено у Додатку Д.

Для проведення тестування програми було вирішено використати модель «чорної скриньки», якої буде достатньо для покриття всіх функціональних вимог.

Функціональні вимоги до програми

За допомогою програми для оцінювання кількості дефектів програмного забезпечення користувач повинен мати змогу:

1) вносити дані;

2) виконувати нормалізацію внесених даних;

3) будувати лінійне рівняння регресії та довірчий інтервал і інтервал прогнозування для нього на основі нормалізованих даних;

4) будувати нелінійне рівняння регресії та довірчий інтервал і інтервал прогнозування для нього на основі ЕД;

5) розраховувати оцінку кількості дефектів програмного забезпечення.

Розроблений програмний продукт було успішно випробувано згідно розробленого документу «Програма і методика випробувань ПЗ», що представлений у додатку Д. У ході випробувань було перевірено наступні функції:

1) Внесення даних;

2) Виконання нормалізації внесених даних;

3) Побудова лінійного рівняння регресії та довірчого інтервалу і інтервалу прогнозування для нього на основі нормалізованих даних;

4) Побудова нелінійного рівняння регресії та довірчого інтервалу і інтервалу прогнозування для нього на основі ЕД;

5) Розрахунок оцінки кількості дефектів ПЗ.

Було отримано такі результати:

– Внесено ЕД.

– Виконано нормалізацію ЕД за допомогою логарифмічного перетворення на основі десяткового логарифму;

– Побудовано лінійне рівняння регресії та довірчий інтервал і інтервал прогнозування для нього на основі нормалізованих даних;

– Побудовано нелінійне рівняння регресії та довірчий інтервал і інтервал прогнозування для нього на основі ЕД;

– Обчислено значення оцінки кількості дефектів програмного забезпечення в залежності від кількості рядків коду.

2. Проведено перевірку програми на її програмно-апаратну сумісність: виконано тестування на апаратно-програмних платформах різної конфігурації, але не нижче за мінімальні вимоги, наведені в технічному завданні (див. Додаток А), виконано перевірку на наявність і усунення всіх помилок.

3. Проведено візуальний перегляд програми: виконано остаточне налагоджено на предмет виявлення та усунення дрібних помилок.

4 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ КІЛЬКОСТІ ДЕФЕКТІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Практичним результатом кваліфікаційної роботи є програма для оцінювання кількості дефектів ПЗ. Було здійснено постановку задачі на розробку ПЗ, розроблено ескізний, технічний та робочий проекти ПЗ.

Згідно з вимогами до програми, наведеними в технічному завданні (Додаток А), програма надає такі функції:

– імпорт вхідних даних X та У з файлу;

– розрахунок параметрів вхідних даних;

– перевірка даних на викиди та нормальний закон розподілу для значень X та У;

- нормалізація даних за допомогою нормалізуючого претворення на основі десяткового логарифму для значень X та У;

- перевірка нормалізованих даних на нормальній закон розподілу для значень X та У;

- побудова лінійного рівняння регресії, довірчого інтервалу, та інтервалу прогнозування;

- побудова нелінійного рівняння регресії, довірчого інтервалу, та інтервалу прогнозування;

- оцінювання кількості дефектів ПЗ;

- експот результатів у файл.

Програма повністю відповідає вимогам до організації вхідних та вихідних даних, вимогам до надійності та іншим вимогам, зазначеним в технічному завданні (Додаток А).

Також була розроблена наступна програмна документація:

– технічне завдання (Додаток А);

– опис програми (Додаток Б)

– текст програми (Додаток В);

– інструкція користувача (Додаток Г);

– програма та методика випробувань ПЗ (Додаток Д).

Програма була розроблена на мові програмування C# в середовищі «Visual Studio 2019».

Створена програма була протестована на відповідність технічному завданню. Під час тестування програма показала повну працездатність.

Розроблена програма може бути використана користувачем, не здатним до програмування. Інтерфейс є інтуїтивно зрозумілим.

5 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВІД РОЗРОБКИ І ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розрахунок вартості програмного продукту та процес формування витрат є дуже важливим моментом для розробника. Саме тому необхідно провести розрахунок витрат на створення й експлуатацію майбутнього ПЗ.

Витрати на розробку ПЗ охоплюють такі витрати:

– заробітна плата розробника;

– амортизація та експлуатація ПК, на якому виконується розробка;

– засоби розробки, матеріали, комплектуючі і т.д.

Основним показником економічної ефективності функціонування ПЗ є підвищення ефективності керування інформацією у вигляді зниження витрат на керування при одночасному збільшенні швидкості і якості одержання потрібного результату.

5.1 Розрахунок витрат на створення і експлуатацію програмного забезпечення

Витрати на розробку програми складаються з витрат на зарплату розробника, на амортизацію ПК, на якому виконується розробка, на експлуатацію цього ПК, на засоби розробки та витрат на матеріали і комплектуючі.

Розробка ПЗ виконується програмістом, місячний оклад якого складає 10000 грн. Додаткова заробітна плата складає 20% від основної. Виходячи з цього, основна і додаткова заробітна плата розроблювача системи 12000 грн/міс, а вартість сучасного ПК складає 15000 грн. (приведена вартість ПК на базі Intel Core i9). При вартості кіловат-години електроенергії рівної 0,9

грн., розраховується вартість розробки ПЗ. Витрати на допоміжні матеріали приведені в табл. 5.1.

Таблиця 5.1 – Витрати на допоміжні матеріали

Пункти витрат	Сума грн.
Папір	40,00
Заправлення картриджа до принтера	90,00
Диск	10,00
Література	250,00
Непередбачені витрати	100,00
Разом	490,00

Вартість ПЗ розрахуємо по формулі:

$$С_{пр} = (З_{зп} + З_{сз} + З_{зг} + З_{е}) * Т + З_{м} \quad (5.1)$$

де Т – тривалість розробки, міс.,

$З_{зп}$ – основна і додаткова заробітна плата обслуговуючого персоналу, грн.,

$З_{сз}$ – відрахування на соціальні заходи (38% від основної і додаткової заробітної плати), грн.,

$З_{зг}$ – загальногосподарські витрати (10% від основної заробітної плати), грн.,

$З_{е}$ – витрати на електроенергію,

$З_{м}$ – витрати на основні і допоміжні матеріали.

$З_{е}$ при споживанні потужності 0,5 кВт, тривалості роботи на місяць, рівної 22*6 = 132 годин, вартості кіловат-години електроенергії 0,9 грн.,

$$З_{е} = 132 * 0.9 * 0.5 = 59,40 \text{ грн.}$$

Витрати на розробку ПЗ наведені в табл. 5.2.

Таблиця 5.2 – Витрати на розробку ПЗ

Найменування витрат	Одиниця виміру	Кількість
Тривалість розробки	Міс	1
Основна і додаткова заробітна плата	Грн.	12000,00
Відрахування на соціальні заходи	Грн.	4560,00
Загальногосподарські витрати	Грн.	1000,00
Витрати на допоміжні матеріали	Грн.	490,00
Витрати на електроенергію	Грн.	59,40

Відповідно до формули (5.1) вартість ПЗ складає:

$$С_{пр} = (12000 + 4560 + 1000 + 59.4) * 1 + 490 = 18109,40 \text{ грн.}$$

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

$$А_{об} = 15000 * 0,6 = 9000,00 \text{ грн.}$$

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

$$В_{м} = 9000 * 0,05 = 450,00 \text{ грн.}$$

Річний обсяг робіт ПК у годинах визначається в такий спосіб:

$$Ф_{м} = 264,5 * Т_{з} \qquad (5.2)$$

де $Т_{з}$ – це середнє місячне завантаження устаткування (близько 4 годин),

264,5 – середня кількість робочих днів у році.

Отже, річний обсяг роботи ПЕОМ складе:

$$Ф_{м} = 264,5 * 4 = 1058 \text{ годин}$$

Витрати на електроенергію Зе при 1058 годинах роботи устаткування в рік складуть

$$З_{е} = 1058 * 0,9 * 0,5 = 476,10 \text{ грн.}$$

Експлуатаційні витрати для ПК за рік складуть:

$$З_{зр} = 9000 + 450 + 476,1 = 9926,10 \text{ грн.}$$

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

$$З_{се} = 18109,4 + 9926,1 = 28035,50 \text{ грн.}$$

5.2 Економічна ефективність розробки і впровадження програмного забезпечення

Основним показником економічної ефективності функціонування системи є підвищення ефективності керування інформацією у вигляді зниження витрат на керування при одночасному збільшенні швидкості і якості одержання потрібного результату.

Крім багатьох інших негативних ефектів, ручна обробка інформації спричиняє наступні негативні економічні ефекти:

– високі витрати на складування паперових документів (сейфи, шафи, папки й ін.);

– підвищені витрати на канцтовари;

– витрати, пов'язані з роботою виявлення раніше допущених помилок (людський фактор).

До числа основних факторів, що визначають приріст прибутку в зв'язку з упровадженням програми, відносяться:

– підвищення продуктивності праці;

– вивільнення робочого часу.

Крім того, не піддається прямій грошовій оцінці підвищення оперативності керування, якість одержуваних результатів, поліпшення організації праці і т.д.

Обов'язковою умовою визначення економічної ефективності програми є порівнянність усіх показників у часі, за цінами й іншими нормами, використовуваними для визначення показників, за змістом і колом елементів витрат.

Визначимо пряму економічну ефективність, ґрунтуючись на тому, що впровадження програмного продукту вивільняє 0,6 працівника (за експертною оцінкою фахівців підприємства).

Зарплата 0,6 працівника в рік складає:

(8000 * 12)* 0.6 = 57600,00 грн.

Річний економічний ефект розраховується по формулі:

$$Э_{год} = \Delta C_n - E_n \cdot k ,$$

де ΔC_n – вивільнені кошти після впровадження програмного продукту (57600 грн.) мінус експлуатаційні витрати (9926,1 грн.) = 47673,90 грн.,

E_n – коефіцієнт ефективності (дорівнює коефіцієнту амортизації(0,6)),

k – одноразові витрати на впровадження продукту (28035,5).

$Э_{год}$ = 47673,9 - 28035,5 * 0,6 = 30852,60 грн.

Строк окупності системи розраховується по формулі:

$$T = \frac{k}{\Delta C} = \frac{28035,5}{47673,9} \approx 0.6 \text{ (року)}$$

Отже строк окупності системи складає приблизно 7 місяців.

6 ОХОРОНА ПРАЦІ

Охорона праці (ОП) – це система соціально-економічних, правових, організаційно-технічних, санітарно-гігієнічних та лікувально-профілактичних заходів та засобів, які спрямовані на збереження життя, здоров’я та працездатності людини під час трудової діяльності [26].

Ця тема є актуальною, бо у теперішній час Україна знаходиться в небезпечному стані щодо питань ОП, життя та здоров’я суспільства. Багато законів залишаються лише у документі, а насправді не діють. Чимало працівників не мають офіційного оформлення, а тому не можуть законно захищати свої права.

ОП охоплює безпосередньо три головні складові: правові норми трудового законодавства, гігієну та техніку безпеки, виробничу санітарію, а також протипожежний захист і електробезпеку.

Метою ОП є забезпечення безпечних, незагрозливих та придатних для праці умов.

6.1 Аналіз шкідливих та небезпечних факторів у відділі розробки ПЗ

Під час роботи на ПК робітники відділення мають справу з дією таких небезпечних факторів: підвищена температура, погане освітлення робочого місця, електричний струм, інтелектуальне або емоційне перевантаження, перенапруження очей, монотонна праця. Проаналізуємо чинники більш докладно.

Небезпечними є монітори з електронно-променевими трубками (ЕЛТ). Робота дисплея спричиняє електростатичні та електромагнітні випромінювання. Згідно зі стандартами не рекомендується використовувати дисплеї з частотою менше, ніж 75 Гц.

Найбільш прийнятними мікрокліматичними параметрами вважається температура 23о Цельсія і 18о Цельсія у теплу та холодну пору року відповідно за відносної вологості 55% [27].

Також можуть виникнути й інші загрози у процесі трудової діяльності.

Загроза ураження електричним струмом. Прийнятним струмом можна вважати такі, що не становлять загрози організму людини, а саме змінний струм промислової частоти 0,6-1,5 мА та постійний струм 5-7 мА [28].

Погане освітлення спричиняє пониження працездатності та продуктивності, надає негативний психологічний вплив на працюючих.

Прийнятна освітленість для роботи за монітором варіюється у межах 400-750 лк, а яскравість в полі зору працівників – у межах 1:5-1:10.

Небезпека виникнення пожежі. Категорія пожежної небезпеки "В" включає споруди, де розміщені ПЕОМ. У таких приміщеннях як правило чимало джерел спалаху: кабелі, техніка, устаткування, меблі, папір тощо.

6.2 Розрахунок захисного заземлення офісного приміщення

Допустимий опір: R_o = 4 Ом;

Тип грунту: садова земля;

Довжина труби заземлювача: l = 3м;

Діаметр труби заземлювача: d = 0.045м;

Глибина закладення грубі від поверхні: t_o = 0.5м.

Розрахунок:

1) Питомий опір ґрунту за даними таблиці - p = 50 Ом * м

2) Для вибраного типу заземлювача і його розмірів визначаємо опір одного заземлювача. Для трубного заземлювача розташованого в грунті, опір розтікання:

$$R_i = \frac{p}{2\pi * l}\left(ln\frac{2l}{d} + \frac{1}{2} * ln\frac{4t + l}{4t -;}\right), t = \frac{l}{2} + t_0$$

$$t = \frac{3}{2} + 0.5 = 1.5$$

Тоді:

$$R_i = \frac{50}{18.84}\left(ln\frac{6}{0.045} + \frac{1}{2} * ln\frac{4*2+3}{4*2-3}\right) = 2.653 * \left(ln133.33 + 0.5 * ln\frac{8+3}{8-3}\right) = 14.026 \text{ Ом}$$

3) Якщо опір одного заземлювача не перевищує допустимий опір пристрою R_j <R_o. то приймаємо 1 штучний заземлювач. Відповідно якщо R_j> R_o, то необхідно приймати кілька штучних заземлювачів, з'єднаних паралельно. У нас R_i> R_o

4) Визначаємо необхідну кількість паралельно з'єднаних заземлювачів:

$n = \frac{R_i}{\tau_o * R_o} - \tau_o$ – коефіцієнт використання заземли гелю ($\tau_o = 1$)

$n = \frac{14.026}{4} = 3.51$ -> n ≥ n: n = 4 заземлювача необхідно.

5) Для зв'язку вершин заземлювачів приймають сполучну лінію. Опір розтікання сполучної риси визначають за формулою:

l_c=1.05*a*n*l=1.05*2*4*3= 25.2 Ом

$R_{lc} = \frac{\rho}{2\pi * l_c} * ln\frac{l^2}{d*t} = \frac{60}{2*3.14*25.2} * ln\frac{9}{0.045*2} = \frac{60}{158,256} * ln\frac{9}{0.09} = 0.31 * 4.6 = 1.426$ Ом

$R_{черты} = \frac{R_{lc}}{\tau_r} = \frac{1.426}{0.89} = 1.6$ Ом

6) Визначаємо фактичний опір заземлювального пристрою:

$$R_з = \frac{R_\varphi * R_{черты}}{R_\varphi + R_{черты}} = \frac{4.38*1.6}{4.38+1.6} = \frac{7.001}{5.98} = 1.17 \text{ Ом}$$

6.3 Розробка заходів щодо зменшення впливу шкідливих та небезпечних факторів

Розглянемо детальніше чинники, котрі призводять до проблем із зором при роботі за персональним комп’ютером.

Основні чинники, які призводять до виникнення проблем із зоровим апаратом:

- недостатнє освітлення робочого місця;
- високий контраст між монітором та оточуючим середовищем.
- відблиски на моніторі;
- мала дистанція між користувачем на монітором;
- мала частота кліпання повіками.

Для покращення освітлення робочого місця зазвичай використовують одну настільну лампу – лампу розжарювання чи компактну ртутну ЛДС, рідше світлодіодну. Але одне джерело світла забезпечує нерівномірне освітлення робочого місця та часто створює відблиски на екрані монітору. Щоб забезпечити рівномірне освітлення також використовують два когерентні джерела світла, наприклад: дві ртутні ЛДС, частоти випромінювання яких є когерентні і через це, вони посилюють одна одну. Але приведений метод, дозволяє тільки частково зменшити контраст між працюючим екраном та оточуючим середовищем і є не економічним.

Принцип дії запропонованого методу досить простий і полягає в розміщенні 4 напрямлених перпендикулярно до користувача джерел світла на корпусі монітору для тилового підсвічування. В якості джерел світла були використані світлодіодні стрічки. Колір випромінювання був обраний спектрально-оптимальний – білий «теплий» (2700–3200 К). А залежно від моделі LED, мають кут розсіювання від 30 до 270 градусів, що дозволяє використовувати їх як для акцентного підсвічування, так і для загального освітлення. Варто також зауважити, що світлодіоди не несуть ніякої загрози навколишньому середовищу.

Проведено експеримент з фотоапаратом та опитування серед програмістів. За допомогою цифрового фотоапарата було створено серію фотознімків у «ручному» режимі, тобто без автоматичного налаштування яскравості, витримки та балансу білого, з тиловою підсвіткою та без неї. Дане рішення дозволяє помітно зменшити контраст, різкий перепад яскравості; завдяки використання LED-стрічки – отримати мініатюрні габарити, низьку собівартість, високу тривалість роботи та безпечність джерел освітлення.

Вимоги безпеки під час роботи з комп’ютером

Щодня перед початком роботи оператор повинен:

– оглянути своє робоче місце;

– про виявлення ознак пошкодження обладнання інформувати свого безпосереднього керівника;

– відрегулювати освітленість на робочому місці, переконатися в відсутності відблисків на екрані комп’ютера, відсутності зустрічного світла;

– перевірити правильність підключення обладнання ЕОМ до електромережі;

– очистити екран комп’ютера від пилу та інших забруднень;

– перевірити правильність організації робочого місця й за необхідності провести відповідні корегування.

Оператор під час роботи зобов'язаний:

– виконувати тільки ту роботу, яку йому було доручено;

– підтримувати порядок і чистоту на робочому місці;

– тримати відкритими всі вентиляційні отвори обладнання;

– коректно закрити всі активні завдання у разі припинення роботи з комп’ютером;

– негайно відключити комп’ютером від електричної мережі у разі виникнення аварійної ситуації.

У ході виконання робіт оператор комп’ютера повинен:

– витримувати відстань від очей до екрана комп’ютером в межах 60-70 см;

– дотримуватися змінного режиму праці та відпочинку, регламентованих перерв у роботі;

– для розробників програм – тривалістю 15 хвилин через кожну годину роботи;

– для інших категорій працівників – тривалістю 15 хвилин через кожні дві години роботи;

– для операторів комп'ютерного набору – тривалістю 10 хвилин, після кожної години роботи.

Під час регламентованих перерв рекомендується виконувати комплекси вправ для очей, рук, хребта, поліпшення мозкового кровообігу тощо. Про виявлення несправності обладнання або інших факторів, які створюють загрозу для життя або здоров'я працівників, необхідно негайно інформувати свого безпосереднього керівника.

Не допускається:

– виконання ремонту та налагодження комп’ютерної техніки безпосередньо на робочому місці оператора;

– зберігання біля комп’ютера паперу, дискет, інших носіїв інформації, запасних блоків, деталей тощо, якщо вони не використовуються для поточної роботи;

– відключення захисних пристроїв, самочинні зміни в конструкції комп’ютера;

– використання комп’ютерів, на екранах яких під час роботи з'являються нехарактерні сигнали, нестабільне зображення на екрані тощо;

– доторкання до задньої панелі системного блоку при включеному живленні;

– вимикання живлення під час виконання активного завдання;

– попадання вологи на поверхню системного блоку, монітора, клавіатури, дисководів, принтерів та інших пристроїв;

– приймання напоїв та їжі на робочому місці.

Після закінчення роботи з використанням необхідно дотримуватися такої послідовності вимикання обладнання:

– закрити всі активні завдання;

– переконатися у відсутності дискет та дисків у дисководах;

– використавши опцію "Завершення роботи" у меню "Пуск", вимкнути живлення системного блоку;

– вимкнути живлення всіх комп’ютерів;

– вимкнути блок аварійного живлення (за наявності);

– відключити комп’ютер від електромережі, при цьому забороняється тягнути штепсельну вилку за дріт.

У випадку виникнення аварійної ситуації оператор зобов'язаний:

– у всіх випадках виявлення пошкодження проводів електричного живлення, несправності заземлення та інших пошкодженнях електрообладнання, виникненні запаху гарі, диму – негайно вимкнути електричне живлення і повідомити про аварійну ситуацію свого безпосереднього керівника й чергового електрика;

– при попаданні людини під електричну напругу негайно звільнити її від дії струму шляхом вимкнення електричного живлення, до прибуття лікаря надати потерпілому долікарську медичну допомогу;

– при будь-яких випадках порушень роботи технічного обладнання або програмного забезпечення негайно викликати представника технічної служби з питань експлуатації обчислювальної техніки;

– у випадку виникнення різі в очах, різкого погіршення зору, виникнення головного болю, больових відчуттів у пальцях та кистях рук, посилення серцебиття – негайно припинити роботу з використанням ЕОМ, повідомити про те, що сталося, свого безпосереднього керівника й звернутися до медичної установи;

– при загорянні обладнання негайно відключити його від електромережі;

– про загорання повідомити свого безпосереднього керівника, оперативного чергового, пожежну службу; ужити заходів щодо ліквідації вогню за допомогою вуглекислотного або порошкового вогнегасника.

7 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

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

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.

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. Наприклад, нафта може служити сировиною для одержання палива, виробництва синтетичних волокон, пластмас, лакофарбових виробів, побутової хімії тощо. І всі ці альтернативні цілі конкурують за використання сирої нафти, обсяги якої, як відомо, обмежені.

Раціональне природокористування означає розробку та здійснення концепції і конкретних заходів щодо раціонального використання і відтворення природних ресурсів, гармонічну взаємодію суспільства і природи, людини і навколишнього природного середовища.

Завдання організації раціонального природокористування вирішується шляхом:

– оптимального розподілу ресурсів між різними господарськими цілями;

– використання технологій, що зберігають ресурси;

– проведення заходів щодо поповнення природних ресурсів.

Іншим, не менш важливим, завданням охорони навколишнього середовища є забезпечення чистоти природних екологічних систем, тобто водного середовища, повітряного басейну, ґрунтових покровів тощо, з тим, щоб забезпечити населення екологічно чистими продуктами харчування, водою, повітрям і, в остаточному підсумку, зберегти високий рівень здоров'я населення та його активного довголіття.

Економічна діяльність у всіх її проявах здійснює забруднення навколишнього середовища. У процесі цієї діяльності забруднюються і стають дефіцитними ресурси повітря, води, територій, що здавалися нескінченними. Нині рівень забруднення досяг загрозливих розмірів, набувши по суті кризового характеру.

Основними принципами охорони навколишнього природного середовища є:

а) пріоритетність вимог екологічної безпеки, обов'язковість додержання екологічних стандартів, нормативів та лімітів використання природних ресурсів при здійсненні господарської, управлінської та іншої діяльності;

б) гарантування екологічно безпечного середовища для життя і здоров'я людей;

в) запобіжний характер заходів щодо охорони навколишнього природного середовища;

г) екологізація матеріального виробництва на основі комплексності рішень у питаннях охорони навколишнього природного середовища,

використання та відтворення відновлюваних природних ресурсів, широкого впровадження новітніх технологій;

д) збереження просторової та видової різноманітності і цілісності природних об'єктів і комплексів;

е) науково обґрунтоване узгодження екологічних, економічних та соціальних інтересів суспільства на основі поєднання міждисциплінарних знань екологічних, соціальних, природничих і технічних наук та прогнозування стану навколишнього природного середовища;

є) обов'язковість екологічної експертизи;

ж) гласність і демократизм при прийнятті рішень, реалізація яких впливає на стан навколишнього природного середовища, формування у населення екологічного світогляду;

з) науково обґрунтоване нормування впливу господарської та іншої діяльності на навколишнє природне середовище;

и) безоплатність загального та платність спеціального використання природних ресурсів для господарської діяльності;

і) стягнення збору за забруднення навколишнього природного середовища та погіршення якості природних ресурсів, компенсація шкоди, заподіяної порушенням законодавства про охорону навколишнього природного середовища;

ї) вирішення питань охорони навколишнього природного середовища та використання природних ресурсів з урахуванням ступеня антропогенної змінності територій, сукупної дії факторів, що негативно впливають на екологічну обстановку;

й) поєднання заходів стимулювання і відповідальності у справі охорони навколишнього середовища;

к) вирішення проблем охорони навколишнього природного середовища на основі широкого міждержавного співробітництва.

Природні ресурси України є власністю народу України, який має право на володіння, використання та розпорядження природними багатствами [29].

7.1 Забруднення навколишнього середовища

Прискорення темпів технологічного прогресу на нашій планеті зумовлює посилення впливу людей на природу, що призводить до якісної зміни співвідношення сил між суспільством і природою. Водночас природні ресурси є основою життя і розвитку людського суспільства і джерелом задоволення потреб.

Забруднюючі речовини, що потрапляють в природне середовище здатні переміщуватись на досить великі відстані, а закономірність цих процесів вивчена ще недостатньо. Ці речовини мігрують у великих кількостях в контурах окремих складових біосфери. Так в атмосфері вони переносяться повітряними течіями.

Питання про забруднення навколоземного космічного простору «космічним сміттям» виникло понад півстоліття тому, з початком запусків штучних супутників в кінці 50-х років. Космічне сміття (головним чином літальні апарати, які відслужили свій термін), по довільним орбітам обертається навколо планети з величезною швидкістю. Ці об'єкти мають величезний запас кінетичної енергії. Час від часу вони стикаються, утворюючи ще більшу кількість дрібних об'єктів – і чим вони дрібніші, тим небезпечнішими вони стають для діючих супутників і літальних апаратів. При достатньо великій кількості зіткнень, кількість нових уламків, які виникають лавиноподібно, може зробити навколоземний космічний простір абсолютно непридатним для польотів.

В деяких випадках об'єкти «космічного сміття», які містять на борту небезпечні (ядерні, токсичні) матеріали або просто мають великі розміри, можуть представляти пряму небезпеку і для Землі – при їх неконтрольованому сході з орбіти, неповному згорянні при проходженні щільних шарів атмосфери Землі і випаданні уламків на населені пункти, промислові об'єкти, транспортні комунікації.

Як знищити космічний мотлох на висоті понад 600 км від поверхні планети? Поки що вчені мають у своєму розпорядженні лише концепти дорогих методів боротьби з цим важкодоступним сміттям. Ефективними практичними можливостями ми поки не володіємо – і воно може безкарно кружляти над нашими головами впродовж тисячоліть.

На щастя, взаємодія з атмосферою на низьких навколоземних орбітах, які використовуються найчастіше, ліквідує основну частину сміття. А зіткнення літальних апаратів з космічним сміттям на менших висотах вже не настільки небезпечні, оскільки при цьому будь-які тіла втрачають швидкість, а з нею і свою кінетичну енергію, а потім, як правило, згорають в щільних шарах атмосфери. На висотах, де тертя об атмосферу незначне, час життя космічного сміття значно зростає. Слабкий вплив атмосфери, сонячного вітру і тяжіння Місяця можуть поступово привести до зниження його орбіти, але на це може знадобитися не одна тисяча років.

Як зазначає НАСА, кількість зафіксованого космічного сміття на орбіті почала зменшуватись ще в 2011 році, а зараз цей процес проходить все більш інтенсивно. Однак ми не можемо розраховувати виключно на допомогу світила в боротьбі з космічними уламками.

7.2 Розробка заходів з охорони навколишнього середовища

До головних завдань в організації природоохоронної діяльності підприємств відноситься:

- аналіз кількісних і якісних показників діяльності підприємства, які здійснюють вплив на довкілля, ефективності запровадження заходів з охорони довкілля і раціонального використання природних ресурсів за відповідний період;

- розробка перспективних та поточних заходів природоохоронної діяльності з обґрунтуванням потреби щодо обсягів їх фінансування, визначення термінів виконання.

Природоохоронні заходи, що запроваджуються підприємством, повинні повністю компенсувати шкідливий вплив виробництва на навколишнє природне середовище і відповідати за напрямками постанові Кабінету міністрів України від 17 вересня 1996 року № 1147 (зі змінами) «Про затвердження переліку видів діяльності, що належать до природоохоронних заходів».

План підприємств з питань охорони навколишнього природного середовища і раціонального використання природних ресурсів складається з таких розділів:

- охорона і раціональне використання водних ресурсів;
- охорона повітряного басейну;
- охорона і раціональне використання земель;
- охорона і раціональне використання мінеральних ресурсів;
- організаційно-просвітницькі заходи.

У розділі «Охорона і раціональне використання водних ресурсів», передбачається комплекс заходів, що забезпечує скорочення витрат питної води, припинення скидів неочищених стоків в поверхневі водні об'єкти, недопущення в скидах стічних вод перевищення нормативних показників забруднюючих речовин. Реалізація забезпечується розробкою заходів по вдосконаленню технологічних процесів виробництва та обладнання, будівництва споруд для очищення стічних вод, створення оборотних систем виробничого водопостачання, впровадження енерго- та ресурсозберігаючих технологій тощо. Крім того, у цьому розділі визначаються обсяги водоспоживання, водовідведення та скидів стічних вод всіх категорій, що використовуються підприємством.

Розділ «Охорона повітряного басейну», містить природоохоронні заходи, спрямовані на зниження обсягів шкідливих речовин, що викидаються

в атмосферне повітря стаціонарними джерелами забруднення на підприємстві та забезпечення дотримання нормативів гранично допустимих концентрацій викидів в санітарно-захисній зоні підприємства.

Показники даного розділу зазначаються окремо для кожного джерела забруднення з подальшим визначенням зведених даних по підприємству.

У розділі «Охорона і раціональне використання земель», відображаються напрями використання земельних ділянок, які знаходяться у користуванні підприємства під час здійснення господарської діяльності і включають заходи по створенню захисних зелених зон, будівництву та реконструкції протиерозійних, гідротехнічних, протикарстових споруд та інших. Передбачається розробка заходів, спрямованих на попередження (ліквідацію) забруднення грунтів відходами виробництва, проведення своєчасної рекультивації порушених земель та використання родючого шару грунту.

До показника, що характеризує площі порушених земель, відносять землі порушені під час добування корисних копалин, що перебувають під будівельними й іншими роботами, пов'язаними з порушенням грунтового покриву, гідрологічного режиму, зайняті під териконами, смітниками і т.п.

Дані про використання земель відображаються в плані у зведеному вигляді за виключенням земель, що рекультивовані і передані землекористувачам для використання.

У розділі «Охорона і раціональне використання мінеральних ресурсів» згруповані заходи з удосконалення методів розробки родовищ корисних копалин і покращення використання сировини, що добувається. Цей розділ планується для запровадження на підприємствах добувних галузей промисловості.

Розділ «Організаційно-просвітницькі заходи» містить заходи, спрямовані на підвищення кваліфікації фахівців з охорони навколишнього природного середовища, рівня обізнаності працівників підприємств, установ, організацій з вимогами природоохоронного законодавства України, зокрема

в сфері поводження з відходами, збереження ресурсів питної води, забезпечення належного санітарного стану територій населених пунктів.

Всі заходи з охорони природного середовища зводяться в єдиний документ, в якому вказується мета роботи, місце впровадження, головний виконавець і співвиконавці, строки виконання робіт, кошторисна вартість, планові затрати, очікуваний результат ефективності їх впровадження.

ВИСНОВКИ

Кваліфікаційна робота присвячена удосконаленню однофакторної регресійної моделі для оцінювання кількості дефектів ПЗ за рахунок використання нормалізуючого перетворення на основі десяткового логарифму, що дає змогу отримати більш точну оцінку кількості дефектів ПЗ в порівнянні з існуючими моделями.

Всі завдання, поставлені до кваліфікаційної роботи, виконані в повному обсязі. Отримані наступні результати:

– досліджено існуючі моделі оцінювання кількості дефектів ПЗ в залежності від його розміру, які є або не достовірними (модель Акіяма), або не досить точними (моделі Холстеда і багатофакторна модель). Вони показують значні відхилення розрахункової кількості дефектів від дійсних показників;

– обґрунтовано необхідність удосконалення однофакторної регресійної моделі для оцінювання кількості дефектів ПЗ;

– побудовано однофакторну регресійну модель для оцінювання кількості дефектів ПЗ на основі логарифмічного перетворення з використанням десяткового логарифму;

– перевірено емпіричні дані на викиди;

– побудовано лінійне рівняння регресії, довірчий інтервал та інтервал прогнозування для нормалізованих даних;

– побудовано нелінійне рівняння регресії, довірчий інтервал та інтервал прогнозування для ЕД;

– розроблено ескізний, технічний та робочий проект ПЗ;

– за допомогою об’єктно-орієнтованої мови програмування C# розроблена програма для оцінювання кількісті дефектів ПЗ в залежності від його розміру;

– здійснено тестування та випробування програми, яке показало її повну працездатність;

– здійснено розрахунок економічної ефективності від розробки та впровадження ПЗ;

– розглянуто спеціальні питання з охорони праці та охорони навколишнього середовища.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Halstead, M. Elements of software science [Text] / M. Halstead // Moscow: Finances and Statistics, 1981. – 128 p.

2. Akiyama, F. An Example of Software System Debugging [Text] / F. Akiyama // Information Processing, 1971. Vol. 71, pp.353-379.

3. Yaremchuk, S.A. The method of software defects number evaluation using complexity metrics [Text] / S.A. Yaremchuk // Radioelectronic and computer systems. – 2012. – No 5. – P.212–218.

4. Марков, А.С. Модели оценки и планирования испытаний программных средств по требованиям безопасности информации [Текст] / А.С. Марков // Вестник МГТУ им. Н.Э. Баумана. Сер. «Приборостроение», 2011. Специальный выпуск «Технические средства и системы защиты информации». с.90-103. ISSN 0236-3933.

5. Пушкін, О.С. Регресійні моделі для оцінювання кількості дефектів програмного забезпечення / О.С. Пушкін, Л.М. Макарова // Матеріали III Всеукраїнської науково-практичної інтернет-конференції студентів, аспірантів та молодих вчених за тематикою «Сучасні комп’ютерні системи та мережі в управлінні»: збірка наукових праць (30 листопада 2020 року) / Під редакцією Г.О. Райко. – Херсон: Видавництво ФОП Вишемирський В. С., 2020. – с.73-74.

6. ISO/IEC Standard No. 9126: Software engineering – Product quality; Parts 1–4. International Organization for Standardization (ISO) / International Electrotechnical Commission (IEC), Geneva, Switzerland, 2001-2004.

7. Поморова, О.В. Аналіз та опрацювання метрик якості програмного забезпечення на етапі проектування [Текст] / О.В. Поморова, Т.О. Говорущенко, С.Я. Тарасек //Хмельницький національний університет, 2010 – 55 с.

8. Albrecht, J. Software function, source lines of codes, and development effort prediction: a software science validation [Text] / J. Albrecht, J.E. Gaffney // IEEE Trans Software Eng. SE-9. – 1983. – P.639–648.

9. Титов, А.И. Выбор метрики размера проекта в модели оценки трудоемкости разработки программ [Текст] / А. И. Титов // Петербургский государственный университет путей сообщения Императора Александра I Санкт-Петербург, 2016. – 32 с.

10. Bonnie, S. Function Point Counting Practices Manual [Text] / S. Bonnie // Function Point Users Group. – Westerville, Ohio, 1994. – 370 p.

11. Cohn, M. Estimating with Use Case Points [Text] / M. Cohn // Methods Tools. – 2005. – Vol. 13, № 3. – P.3–13.

12. Електронний конспект лекцій з дисципліни «Емпіричні методи програмної інженерії» [Електронний ресурс] / Режим доступу: URL: http://tc.kpi.ua/content/kurs/ – 15.10.2020 р – Загол. з екрану.

13. Приходько, С.Б. Доверительный интервал нелинейной регрессии времени восстановления работоспособности устройств терминальной сети [Текст] / С.Б. Приходько, Л.Н. Макарова // Восточно-Европейский журнал передовых технологий. – 2014. – № 3(4). – С.26-31.

14. Вентцель, Е.С. Теория вероятностей: Учеб. для вузов [Текст] / Е.С. Вентцель. – М.: Высш. шк., 1999. – 576 с.

15. Орлов, А.И. Прикладная статистика. [Текст] / А.И. Орлов. – М.: Экзамен, 2004. – 117 с.

16. Магнус, Я.Р. Эконометрика. Начальный курс: Учеб. – 6-е изд., перераб. и доп. [Текст] / Я.Р. Магнус, П.К. Катышев, А.А. Пересецкий. – М.: Дело, 2004. – 576 с.

17. Transform data to normal distribution [Електронний ресурс] / Режим доступу: URL: http://www.sigmamagic.com/forum/archives/297 – 15.10.2020 р – Загол. з екрану.

18. Аппроксимация закона распределения экспериментальных данных [Електронний ресурс] / Режим доступу: URL:

http://opds.sut.ru/old/electronic_manuals/oed/f05.htm#r054 – 15.10.2020 р – Загол. з екрану.

19. Prykhodko, S. Multivariate outlier detection technique based on normalizing transformations for non-Gaussian data [Text] / S. Prykhodko, N. Prykhodko, L. Makarova, K. Pugachenko // «Інформаційні технології та комп`ютерне моделювання»: матеріали статей Міжнародної науково-практичної конференції, м. Івано-Франківськ, 15-20 травня 2017 року. – Івано-Франківськ: п. Голіней О.М., 2017. – С.170-174.

20. Christopher, J. The Legacy of Space Shuttle Flight Software [Text] /C.J. Hickey, J.B. Loveall, J.K. Orr, A.L. Klausman // NASA Johnson Space Center. – Houston, Texas, 2011. – P.7-24.

21. Уніфікована мова моделювання [Електронний ресурс] / Режим доступу: URL: https://bibliofond.ru/view.aspx?id=446563#1 – 15.10.2020 р – Загол. з екрану.

22. Діаграма варіантів використання [Електронний ресурс] / Режим доступу: \www/ URL: http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?id=28086 – 15.10.2020 р – Загол. з екрану.

23. Крэг, Л. Применение UML 2.0 и шаблонов проектирования / Л. Крэг // 3-е изд. – М.: Вильямс, 2006. – 736 с. ISBN 0-13-148906-2.

24. Діаграма станів [Електронний ресурс] / Режим доступу: URL: http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?id=2808710 – 15.10.2020 р – Загол. з екрану.

25. Котов, О.М. Язык C#: краткое описание и введение в технологии программирования : учебное пособие [Текст] / О. М. Котов. // Екатеринбург : Издательство Урал. ун-та, 2014. – 208 с.

26. Поняття, мета і завдання охорони праці [Електронний ресурс] / Режим доступу: URL: http://ncpn.net.ua/oxrana_truda.html – 15.10.2020 р – Загол. з екрану.

27. ГОСТ 12.1.005-88. Система стандартов безопасности труда. Общие санитарно-гигиенические требования к воздуху рабочей зоны [Текст]. – Введ. 1989-01-01 – М.: Стандартинформ, 2000. – С. 4.

28. ГОСТ 12.1.038-82. Система стандартов безопасности труда. Электробезопасность. Предельно допустимые значения напряжений прикосновения и токов [Текст]. – Введ. 1983-07-03 – М.: Стандартинформ, 1998. – С. 10.

29. Охорона навколишнього середовища [Електронний ресурс] / Режим доступу: \www/ URL: https://pidruchniki.com/13351207/ekonomika/ohorona_navkolishnogo_seredovischa – 15.10.2020 р – Загол. з екрану.

Додаток А Технічне завдання

Вступ

Назва розроблюваного проекту: «Програма для оцінювання кількості дефектів ПЗ». Галузь застосування – підприємства, які спеціалізуються на розробці ПЗ.

1. Підстави для розробки

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

2. Призначення розробки

2.1 Експлуатаційне призначення

Дана розробка призначена для побудови лінійного та нелійного рівняння регресії, довірчих інтервалів та інтервалів прогнозування для оцінювання кількості дефектів ПЗ.

2.2 Функціональне призначення програми

Функціональним призначенням програми є автоматизація розрахунків, яка полегшить побудову лінійного та нелійного рівняння регресії, довірчих інтервалів та інтервалів прогнозування для оцінювання кількості дефектів ПЗ.

3. Вимоги до програмного забезпечення

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу виконуваних функцій

ПЗ повинно виконувати наступні функції:

– імпорт вхідних даних X та У з файлу;

– розрахунок параметрів вхідних даних;

– перевірка даних на викиди та нормальний закон розподілу для значень X та У;

- нормалізація даних за допомогою нормалізуючого претворення на основі десяткового логарифму для значень X та У;

- перевірка нормалізованих даних на нормальній закон розподілу для значень X та У;
- побудова лінійного рівняння регресії, довірчого інтервалу, та інтервалу прогнозування;
- побудова нелінійного рівняння регресії, довірчого інтервалу, та інтервалу прогнозування;
- оцінювання кількості дефектів ПЗ;
- експот результатів у файл.

3.1.2 Вимоги до організаційних вхідних та вихідних даних

Введення даних здійснюється за допомогою пристроїв введення даних (клавіатури та миші). Дані або вводяться вручну, або обираються із представлених списків.

Обмін інформацією між програмним забезпеченням відбувається за допомогою файлів з довільним доступом, з розширенням .xls

Вхідні дані: KLOCs, major Errors.

Вихідні дані: дані, нормалізовані за допомогою нормалізуючого перетворення на основі десяткового логарифму.

3.2 Вимоги до надійності

Програмний продукт повинен функціонувати при безперебійній роботі персонального комп’ютера. При виникненні збоїв в роботі, відновлення нормальної роботи повинне проводитися після перезавантаження програми.

У разі введення користувачем некоректної інформації система повинно повідомити про помилку і надати можливість виправити її.

3.3 Вимоги до умов експлуатації

Необхідний рівень підготовки користувачів: мінімальні навики в користуванні комп’ютером. Комп’ютер призначений для роботи в закритому опалювальному приміщені при наступних умовах навколишнього середовища:

- температура повітря від $+10^0$C до $+30^0$C;
- атмосферний тиск від 630 мм до 800 мм ртутного стовпа;

- відносна вологість повітря не більше 80%;
- запиленість повітря не більше 0,75 мг/м^2.

3.4 Вимоги до складу та параметрів технічних засобів

Система повинна задовольняти мінімальній комплекції:

- Pentium IV – 400 МГц або AMD – 300 МГц;
- ОЗУ – 2 Гб;
- вільної пам’яті на жорсткому диску не менше 5 Гб.

3.5 Вимоги до інформаційної та програмної сумісності

Система також повинна відповідати вимогам до інформаційної та програмної сумісності. Для повноцінного функціонування системи необхідно використовувати операційну систему Windows 7 – 32-bit/64-bit або вище.

Для використання ПЗ потрібно мати встановлену платформу «.NET Framework».

3.6 Вимоги до маркування та пакування

Програмне забезпечення буде упаковано в електронний архів і мати найменування KLOCs.rar.

3.7 Вимоги до транспортування та зберігання

Вимоги до транспортування не висовуються у зв’язку з відсутністю фізичних носів.

4. Вимоги до програмної документації

- технічне завдання;
- текст програми;
- опис програми;
- керівництво користувача;
- програма та методика випробувань ПЗ.

5. Техніко-економічні показники

Не розраховуються в зв’язку з тим, що продукт розробляється в рамках кваліфікаційної роботи.

6. Стадії та етапи розробки

Стадії та етапи розробки представлені нижче в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки проекту

Стадії розробки	Етапи робіт	Термін виконання робіт	
		Початок етапу	Кінець етапу
1	2	3	4
1. Технічне завдання	1.1Обгрунтування необхідності розробки програми	06.09.20	11.09.20
	1.2 Розробка технічного завдання	11.09.20	16.09.20
	1.3 Затвердження технічного завдання	16.09.20	28.09.20
2. Ескізний проект	2.1 Розробка ескізного проекту	28.09.20	08.10.20
	2.2 Затвердження ескізного проекту	08.10.20	15.10.20
3. Технічний проект	3.1 Розробка технічного проекту	15.10.20	21.10.20
	3.2 Затвердження технічного проекту	21.10.20	01.11.20
4. Робочий проект	Розробка робочого проекту	01.11.20	07.11.20
	Затвердження робочого проекту	07.11.20	15.11.20
	Розробка програмної документації	15.11.20	20.11.20

7. Порядок контролю та прийому

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.

Прийом 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 на протязі не більше ніж 2 тижнів виправити зазначені помилки і оповістити замовника про повторне проведення перевірки.

Додаток Б Текст програми

Linearregres.cpp:

```
public static List<double[]> loadPoints(String file,
    int pointsPerFile, int cenVecSize,int labelSize,
    Configuration conf) throws Exception {
    System.out.println("filename: "+file );
    List<double[]> points = new LinkedList<double[]>();
    double[] trainingData =
      new double[pointsPerFile * cenVecSize];
      double[] labelData =
      new double[pointsPerFile * labelSize];
    Path pointFilePath = new Path(file);
    FileSystem fs =
      pointFilePath.getFileSystem(conf);
    FSDataInputStream in = fs.open(pointFilePath);
    int k =0;
    int p = 0;
    try{
      for(int i = 0; i < pointsPerFile;i++){
        String[] line = in.readLine().split(",");
        for(int j = 0; j < cenVecSize; j++){
          trainingData[k] = Double.parseDouble(line[j]);
          k++;
        }
        for(int s = cenVecSize; s < cenVecSize+labelSize; s++){
          labelData[p] = Double.parseDouble(line[s]);
          p++;
        }
      }
    } finally{
      in.close();
      points.add(trainingData);
      points.add(labelData);
    }
    return points;
  }
```

ListLinearregres.cpp:

```
List<List<double[]>> pointArrays = LinRegUtil.loadPoints(trainingDataFiles,
pointsPerFile,
                    vectorSize, nDependentVariables, conf, numThreads);
      List<double[]> featurePoints = new LinkedList<>();
      for(int i = 0; i<pointArrays.size(); i++){
          featurePoints.add(pointArrays.get(i).get(0));
      }
      List<double[]> labelPoints = new LinkedList<>();
      for(int i = 0; i<pointArrays.size(); i++){
          labelPoints.add(pointArrays.get(i).get(1));
      }
for(int k=0;k<featurePoints.size();k++)
            {
                array_data_feature[k] = featurePoints.get(k);
                array_startP_feature[k] = totalLengthFeature;
                totalLengthFeature += featurePoints.get(k).length;
            }

      for(int k=0;k<labelPoints.size();k++)
```

```
      {
          array_data_label[k] = labelPoints.get(k);
          array_startP_label[k] = totalLengthLabel;
          totalLengthLabel += labelPoints.get(k).length;
      }

    long featuretableSize = totalLengthFeature/nFeature;
    long labeltableSize = totalLengthLabel/nLabel;

   //initializing Numeric Table


    NumericTable featureArray_daal = new HomogenNumericTable(daal_Context,
Double.class, nFeature, featuretableSize,
NumericTable.AllocationFlag.DoAllocate);
    NumericTable labelArray_daal = new HomogenNumericTable(daal_Context,
Double.class, nLabel, labeltableSize,
NumericTable.AllocationFlag.DoAllocate);
```

LRegress.h

```
TrainingDistributedStep1Local linearRegressionTraining = new
TrainingDistributedStep1Local(daal_Context, Float.class,
                    TrainingMethod.qrDense);
    linearRegressionTraining.input.set(TrainingInputId.data, trainData);
    linearRegressionTraining.input.set(TrainingInputId.dependentVariable,
trainDependentVariables);

    PartialResult pres = linearRegressionTraining.compute();
    ts2 = System.currentTimeMillis();
    compute_time += (ts2 - ts1);

    ts1 = System.currentTimeMillis();
    partialResultTable.addPartition(new Partition<>(this.getSelfID(),
serializePartialResult(pres)));
    boolean reduceStatus = false;
    reduceStatus = this.reduce("linreg", "sync-partialresult",
partialResultTable, this.getMasterID());
if(this.isMaster()){
      TrainingDistributedStep2Master linearRegressionTrainingMaster = new
TrainingDistributedStep2Master(daal_Context, Float.class,
                TrainingMethod.qrDense);
      ts1 = System.currentTimeMillis();
      int[] pid = partialResultTable.getPartitionIDs().toIntArray();
      for(int j = 0; j< pid.length; j++){
        try {

linearRegressionTrainingMaster.input.add(MasterInputId.partialModels,

deserializePartialResult(partialResultTable.getPartition(pid[j]).get()));
        } catch (Exception e)
          {
            System.out.println("Fail to deserilize partialResultTable" +
e.toString());
            e.printStackTrace();
          }
      }
      ts2 = System.currentTimeMillis();
      comm_time += (ts2 - ts1);

      ts1 = System.currentTimeMillis();
      linearRegressionTrainingMaster.compute();
      trainingResult = linearRegressionTrainingMaster.finalizeCompute();
```

```
      ts2 = System.currentTimeMillis();
      compute_time += (ts2 - ts1);
      model = trainingResult.get(TrainingResultId.model);
    }
```

LRegressResult.cpp

```
PredictionBatch linearRegressionPredict = new PredictionBatch(daal_Context,
Float.class, PredictionMethod.defaultDense);
    NumericTable testData = getNumericTableHDFS(daal_Context, conf,
testFilePath, 10, 250);
    linearRegressionPredict.input.set(PredictionInputId.data, testData);
    linearRegressionPredict.input.set(PredictionInputId.model, model);

    /* Compute the prediction results */
    ts1 = System.currentTimeMillis();
    predictionResult = linearRegressionPredict.compute();
    results = predictionResult.get(PredictionResultId.prediction);
```

Normalize.cpp

```
      inline cInt perp_dot(const IntPoint& a, const IntPoint& b) {
            return (-a.Y * b.X) + (a.X * b.Y);
      }

      bool edge_collision(const IntPoint& a1, const IntPoint& a2, const
IntPoint& b1, const IntPoint& b2, bool relaxed) {
            IntPoint a(a2.X - a1.X, a2.Y - a1.Y);
            IntPoint b(b2.X - b1.X, b2.Y - b1.Y);

            cInt f = perp_dot(a, b);

            // check if lines are parallel
            if (f == 0) {
                  // check if collinear intersecting
                  if (a2.X - a1.X == 0) {
                        cInt u = (b1.Y - a1.Y) / (a2.Y - a1.Y);
                        cInt v = (b2.Y - a1.Y) / (a2.Y - a1.Y);
                        return u >= 0 && u <= 1 && v >= 0 && v <= 1;
                  }
                  else {
                        cInt u = (b1.X - a1.X) / (a2.X - a1.X);
                        cInt v = (b2.X - a1.X) / (a2.X - a1.X);
                        return u >= 0 && u <= 1 && v >= 0 && v <= 1;
                  }
                  // if above algo is bad, then:
                  // return perp_dot(b1 - a1, b2 - a1) == 0

                  return false;
            }

            IntPoint c(a1.X - b1.X, a1.Y - b1.Y);
            cInt s = perp_dot(a, c);
            cInt t = perp_dot(b, c);

            if (relaxed)
                  if (f < 0) {
                        if (s >= 0 || t >= 0 || s <= f || t <= f) return
false;
                  }
                  else {
```

```
                        if (s <= 0 || t <= 0 || s >= f || t >= f) return
false;
                }
            else {
                if (f < 0) {
                    if (s > 0 || t > 0 || s < f || t < f) return false;
                }
                else {
                    if (s < 0 || t < 0 || s > f || t > f) return false;
                }
            }

            return true;
      }

      bool edge_poly_collision(const IntPoint& a1, const IntPoint& a2, const
Path& poly) {
            for (Path::const_iterator b2 = poly.begin(), b1 = poly.end() - 1;
b2 != poly.end(); b1 = b2, ++b2) {
                bool relaxed = (&a1 == &*b1) || (&a1 == &*b2) || (&a2 ==
&*b1) || (&a2 == &*b2);
                if (edge_collision(a1, a2, *b1, *b2, relaxed)) return true;
            }
            return false;
      }

      bool splice_vertices(const Path& path1, const Path& path2, Path& result)
{
            bool combined = false;

            int n = 0;
            for (const IntPoint& b : path1) {
                for (const IntPoint& a : path2) {

                    if (a.X == b.X && a.Y == b.Y) {

                        for (const IntPoint& aa : path2) {
                            result.push_back(aa);

                            if (&aa == &a) {
                                for (int i = (n + 1) % path1.size();
i != n; i = (i + 1) % path1.size()) {
                                    result.push_back(path1[i]);
                                }
                                result.push_back(a);
                            }
                        }

                        combined = true;
                        break;
                    }

                }
                if (combined) break;
                ++n;
            }
            return combined;
      }

      Path keyhole_poly(const Path& poly, const Paths& holes) {
            Path keyedpoly = poly;

            Paths combined_holes = holes;
```

```
            for (int h1 = 0; h1 < combined_holes.size() - 1; ++h1) {
                  const Path& hole1 = combined_holes[h1];
                  for (int h2 = h1 + 1; h2 < combined_holes.size(); ++h2) {
                        const Path& hole2 = combined_holes[h2];
                        Path combined;
                        if (splice_vertices(hole1, hole2, combined)) {
                              // Add combined hole back into vector
                              combined_holes.push_back(combined);

                              combined_holes.erase(combined_holes.begin() +
h2);
                              combined_holes.erase(combined_holes.begin() +
h1);
                              --h2;
                              --h1;
                        }

                  }
            }

            Paths holes_to_key;
            for (const Path& hole : combined_holes) {
                  Path combined;
                  if (splice_vertices(keyedpoly, hole, combined)) {
                        keyedpoly = combined;
                  }
                  else {
                        holes_to_key.push_back(hole);
                  }
            }


            Paths deffered_holes;
            for (const Path& hole : holes_to_key) {
                  Path temp_poly;

                  bool finished_keyhole = false;

                              int n = 0;
                  for (const IntPoint& b : hole) {
                        for (const IntPoint& a : keyedpoly) {


                              if (edge_poly_collision(a, b, keyedpoly))
continue;
                              bool intersect = false;
                              for (const Path& path : holes_to_key) {
                                    if (edge_poly_collision(a, b, path)) {
                                          intersect = true;
                                          break;
                                    }
                              }
                              if (intersect) continue;
                              for (const IntPoint& outer : keyedpoly) {
                                    temp_poly.push_back(outer);

                                    // begin splice
                                    if (&outer == &a) {
                                          temp_poly.push_back(b);
                                          for (int i = (n + 1) % hole.size();
i != n; i = (i + 1) % hole.size()) {
                                                temp_poly.push_back(hole[i]);
                                          }
```

```
                                    temp_poly.push_back(b);
                                    temp_poly.push_back(a);
                              }
                        }
                        finished_keyhole = true;
                        break;
                  }
                  if (finished_keyhole) break;
                  ++n;
            }

            if (finished_keyhole)
                  keyedpoly = temp_poly;
            else {
                  deffered_holes.push_back(hole);
            }
      }

      if (deffered_holes.size() > 0) {
            return keyhole_poly(keyedpoly, deffered_holes);
      }

      return keyedpoly;
}

void keyhole_tree(const PolyNode& node, Paths& out) {
      if (node.IsHole()) {

            for (const PolyNode* child : node.Childs)
                  keyhole_tree(*child, out);
      }
      else {
            if (node.ChildCount() == 0) {
            out.push_back(node.Contour);
            }
            else {

                  for (const PolyNode* hole : node.Childs)
                        holes.push_back(hole->Contour);

                  out.push_back(keyhole_poly(node.Contour, holes));
            }
      }
}

}
```

Додаток В Опис програми

1 Загальні відомості

Дана програма призначена для побудови лінійного та нелійного рівняння регресії, довірчих інтервалів та інтервалів прогнозування для оцінювання кількості дефектів ПЗ.

2 Функціональне призначення

Функціональним призначенням програми є автоматизація розрахунків яка полегшить побудови лінійного та нелійного рівняння регресії, довірчих інтервалів та інтервалів прогнозування для оцінювання кількості дефектів ПЗ.

Програмне забезпечення повинно виконувати наступні функції:

– імпорт вхідних даних X та У з файлу;

– розрахунок параметрів вхідних даних;

– перевірка даних на викиди та нормальний закон розподілу для значень X та У;

- нормалізація даних за допомогою нормалізуючого претворення на основі десяткового логарифму для значень X та У;

- перевірка нормалізованих даних на нормальній закон розподілу для значень X та У;

- побудова лінійного рівняння регресії, довірчого інтервалу, та інтервалу прогнозування;

- побудова нелінійного рівняння регресії, довірчого інтервалу, та інтервалу прогнозування;

- оцінювання кількості дефектів ПЗ;

- експот результатів у файл.

3 Опис логічної структури

Програма розроблена на модульній основі, тобто кожний модуль програми відповідає за конкретну дію: імпорт даних, нормалізація, побудова лінійного та нелінійного рівняння регресії, оцінювання кількості дефектів ПЗ та інші.

4 Технічні та програмні засоби

Система повинна задовольняти мінімальній комплекції:

- Pentium IV – 400 МГц або AMD – 300 МГц;
- ОЗУ – 2 Гб;
- вільної пам’яті на жорсткому диску не менше 5 Гб.

Система також повинна відповідати вимогам до інформаційної та програмної сумісності. Для повноцінного функціонування системи необхідно використовувати операційну систему Windows 7 – 32-bit/64-bit або вище.

Для використання програмного забезпечення потрібно мати встановлену платформу «.NET Framework».

5 Виклик та завантаження

ПЗ для побудови лінійного та нелійного рівняння регресії, довірчих інтервалів та інтервалів прогнозування для оцінювання кількості дефектів ПЗ встановлюється безпосередньо на комп’ютер користувача. ПЗ відкривається або за допомогою ярлика, який знаходиться на робочому столі, або за допомогою вибору відповідного пункту меню «Пуск» – «Програми». Програма має один режим використання.

6 Вхідні та вихідні дані

Введення даних здійснюється за допомогою пристроїв введення даних (клавіатури та миші). Дані або вводяться вручну, або обираються із представлених списків.

Обмін інформацією між програмним забезпеченням відбувається за допомогою файлів з довільним доступом, з розширенням .xls.

Вхідні дані: KLOCs, major Errors.

Вихідні дані: дані, нормалізовані за допомогою нормалізуючого перетворення на основі десяткового логарифму.

Додаток Г Інструкція користувача

Для запуску програмного забезпечення потрібно запустити файл «CalcRegression.exe», після чого з’явиться головне вікно, яке представлено на рисунку Г.1.

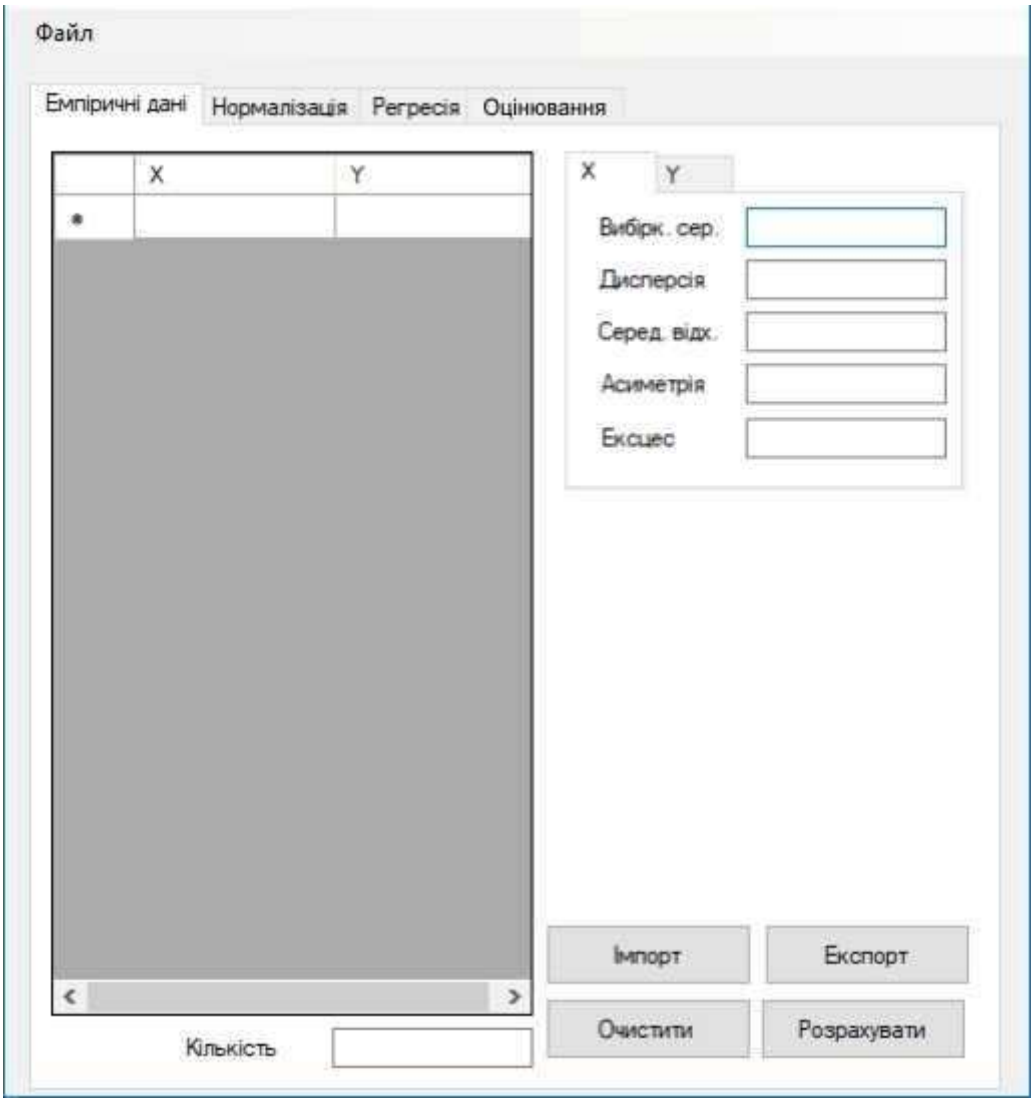

Рисунок Г.1 – Головне вікно

Для завантаження даних з файлу потрібно обрати кнопку «Імпорт». З’явиться вікно, в якому потрібно обрати шлях та файл з розширенням .xls для завантаження. Вікно для імпорту даних зображено на рисунку Г.2.

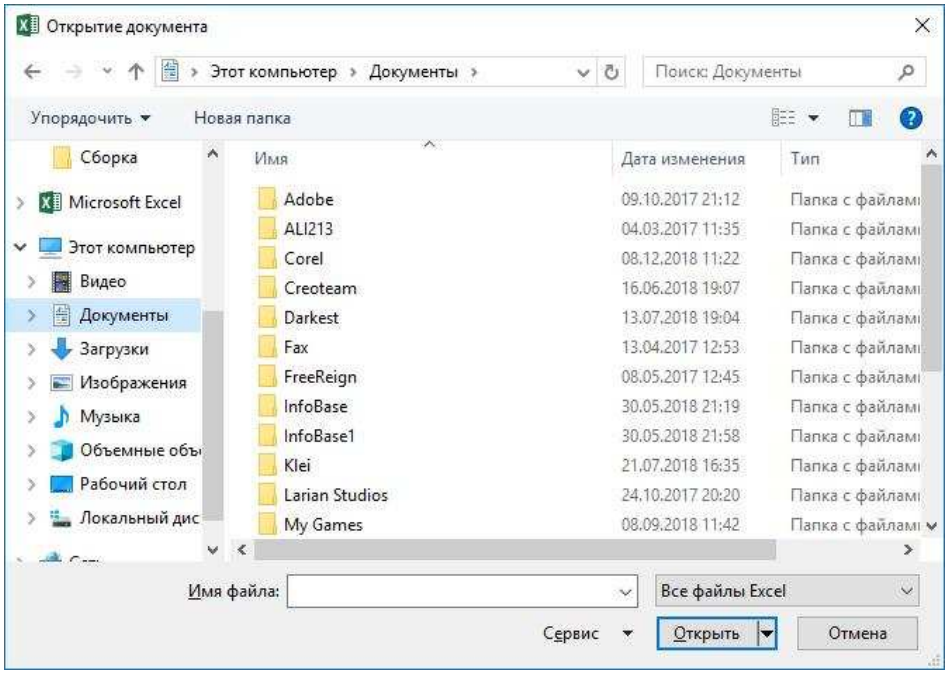

Рисунок Г.2 – Вікно для імпорту даних

Після вибору файлу з даними для завантаження, дані будуть завантажені у програму та пройдуть первісну перевірку. В разі успішної перевірки вони будуть внесені до таблиці вхідних даних. Для здійснення розрахунків потрібно обрати кнопку «Розрахувати» у вкладці «Емпіричні дані». Далі програма розрахує статистичні параметри вибірки та виведе результати розрахунку у відповідня поля. Результат роботи функції розрахунку параметрів вхідних даних представлено на рисунку Г.3.

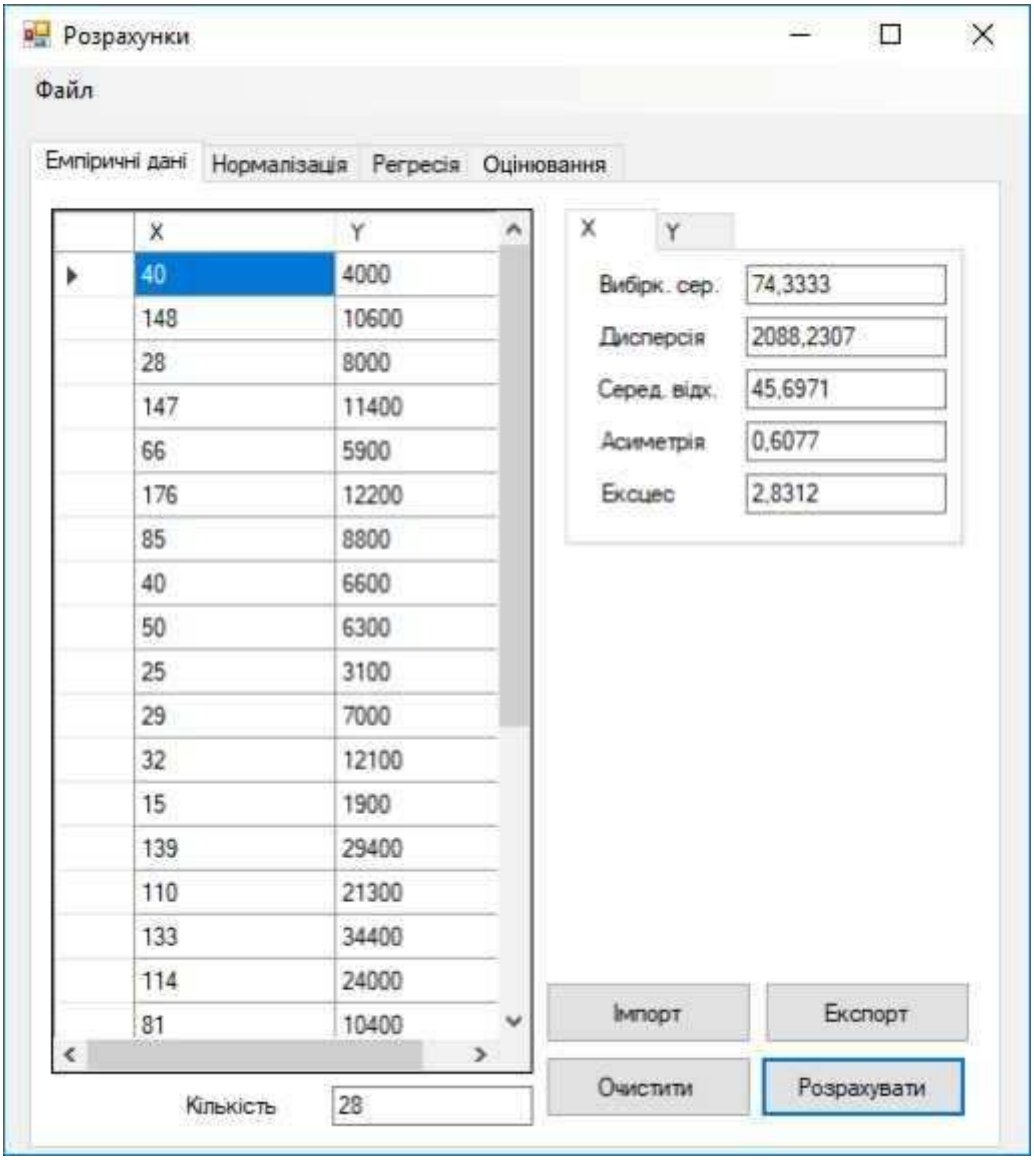

Рисунок Г 3 – Результат роботи функції розрахунку параметрів вхідних даних

Якщо обрано не вірний файл для імпорту, треба обрати кнопку «Очистити». Після цього всі дані, внесені в програму та розраховані програмою, будуть очищені. Розраховані дані можна експортувати. Для цього потрібно обрати кнопку «Експорт». Далі з’явиться вікно, в якому потрібно обрати шлях та вказати ім’я файлу для експорту даних.

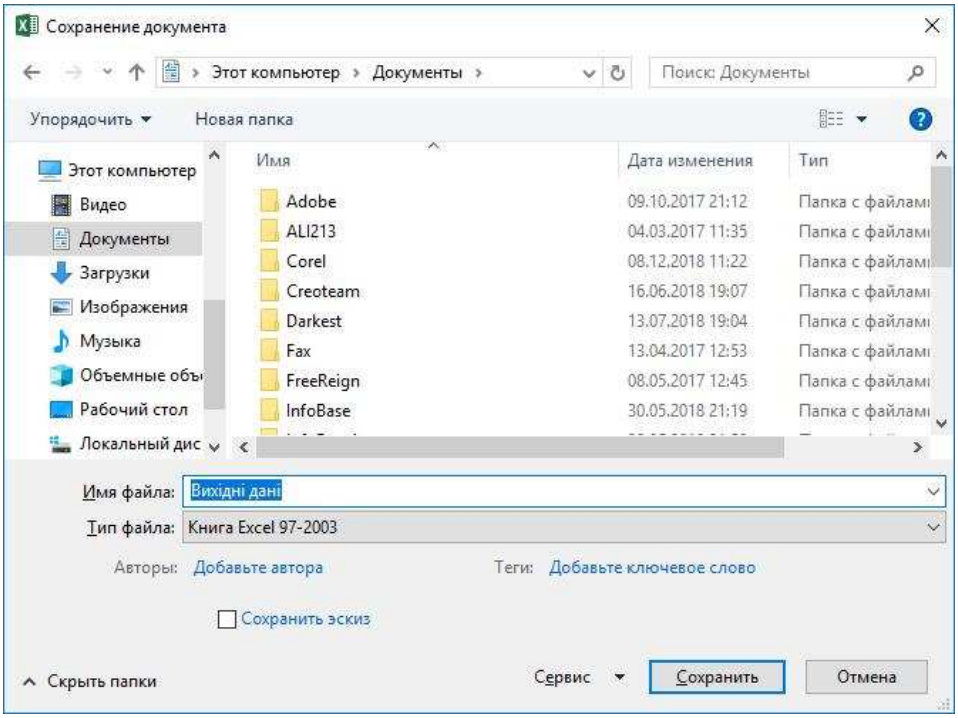

Рисунок Г.4 – Вікно для експорту даних

Для нормалізації даних потрібно перейти на вкладку «Нормалізація» та обрати кнопку «Розрахувати». Далі програма розрахує нормалізовані дані для вхідної вибірки, статистичні параметри нормалізованої вибірки, а також дані, необхідні для перевірки отриманих даних на нормальний закон розподілу. Результати розрахунків будуть виведені у відповідня поля. Результат роботи функції нормалізації вхідних даних представлено на рисунку Г.5.

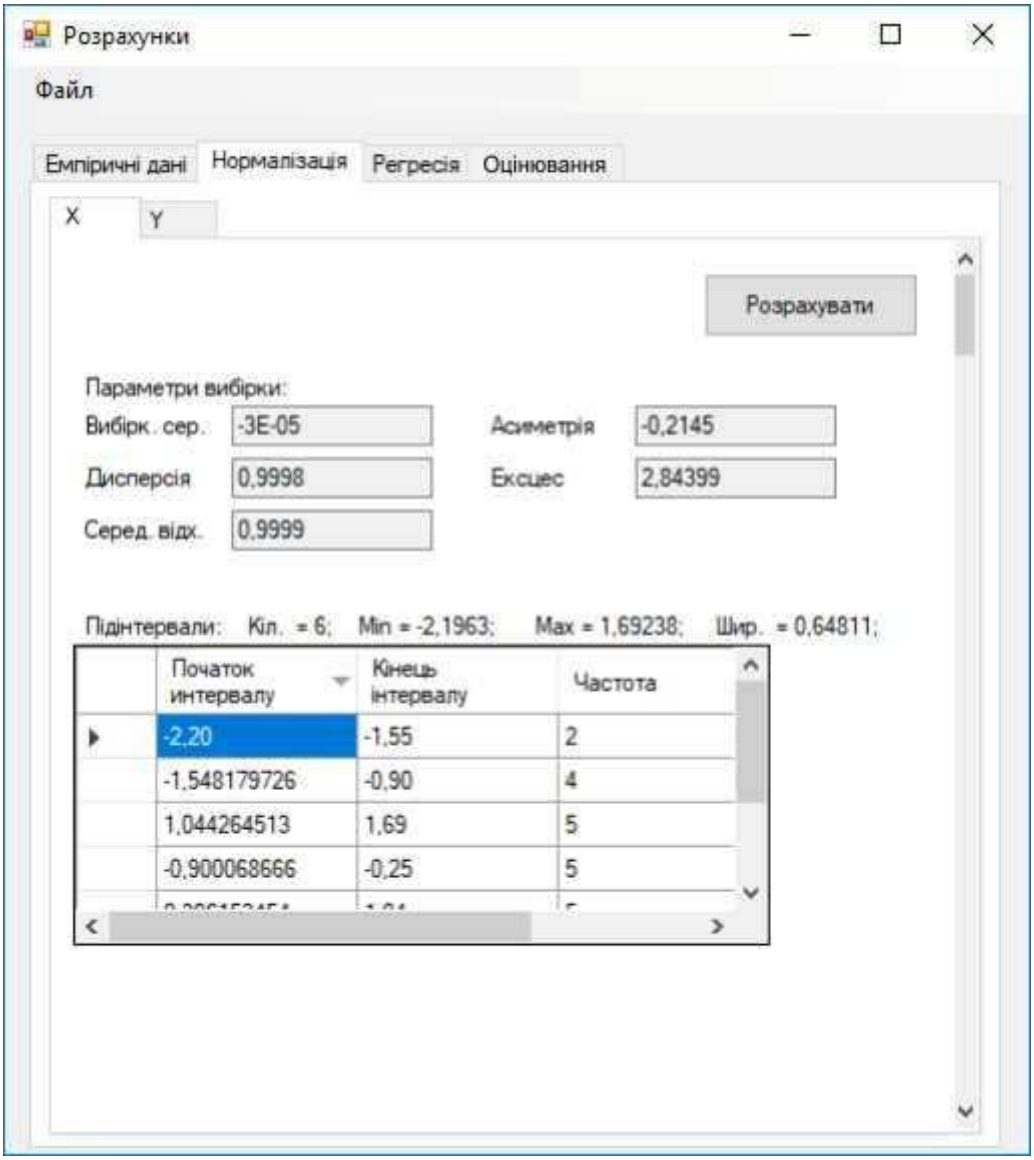

Рисунок Г.5 – Результат роботи функції нормалізації вхідних даних

Для побудови рівняння регресії потрібно перейти на вкладку «Регресія» та обрати кнопку «Побудувати». Далі буде побудовано лінійне рівняння регресії та довірчий інтервал і інтервал прогнозування лінійного рівняння регресії. Далі буде побудовано нелінійне рівняння регресії і довірчий інтервал та інтервал прогнозування нелінійного рівняння регресії.

Для отримання значення оцінки кількості дефектів ПЗ потрібно перейти на вкладку «Оцінювання» та ввести значення, необхідне для розрахунку – кількість вихідних рядків коду у тисячах рядків та обрати кнопку «Розрахунок». Далі буде розраховано значення оцінки кількості

дефектів ПЗ, довірчий інтервал та інтервал прогнозування. Результат роботи функції оцінювання кількості дефектів ПЗ представлено на рисунку Г.6.

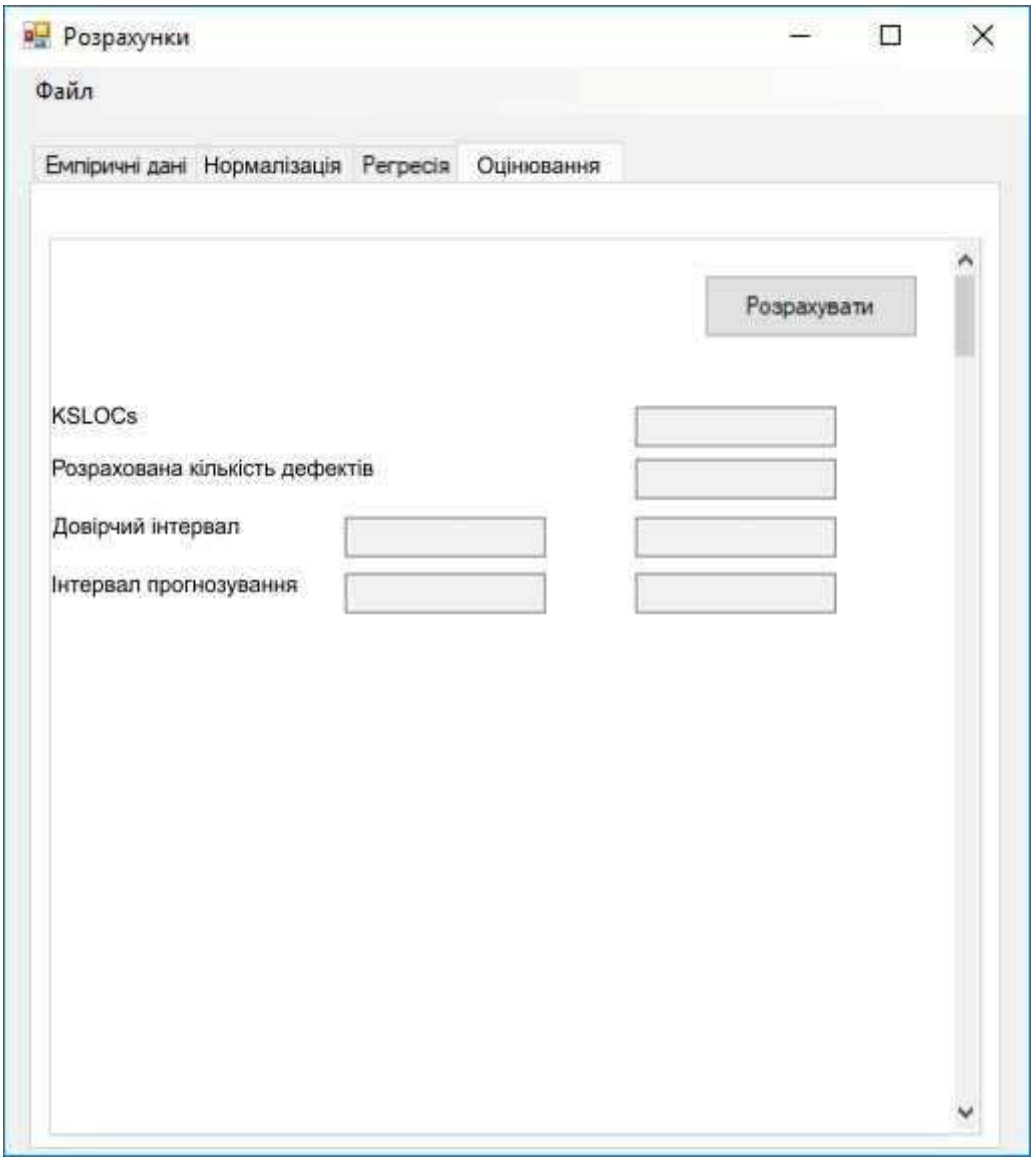

Рисунок Г.6 – Результат роботи функції оцінювання кількості дефектів ПЗ

Додаток Д

Програма та методика випробувань ПЗ

1. Об’єкт випробувань

1.1 Назва об’єкту

Назва розроблюваного проекту: «Програма для оцінювання кількості дефектів ПЗ». Галузь застосування – підприємства, які спеціалізуються на розробці ПЗ.

Коротка назва: програма.

1.2 Область застосування

Функціональним призначенням програми є автоматизація розрахунків, яка полегшить побудову лінійного та нелійного рівняння регресії, довірчих інтервалів та інтервалів прогнозування для оцінювання кількості дефектів ПЗ.

2 Мета випробувань

Мета проведення випробувань – оцінка експлуатаційних характеристик програми, перевірка і підтвердження працездатності програми в умовах, максимально наближених до умов реальної експлуатації.

3 Вимоги до програми

Програма має реалізовувати функції, описані в технічному завданні, яке наведено в додатку А.

4 Вимоги до програмної документації

Для проведення тестування програми повинна бути надана наступна документація:

- технічне завдання на розробку програми;
- текст програми;
- опис програми;
- інструкція користувача;
- програма і методика випробувань ПЗ.

5 Склад, порядок та методи випробувань

5.1 Перевірка програми на відповідність технічному завданню:

– перевірка роботи функції «Імпорт даних»;

– перевірка роботи функції «Нормалізіція даних»;

– перевірка роботи функції «Перевірка на нормальний закон розподілу»;

– перевірка роботи функції «Експорт даних у файл».

5.2 Перевірка програми на її програмно-апаратну сумісність: тестування на апаратних-програмних платформах різної конфігурації, але не нижче за мінімальні вимоги, наведені в технічному завданні; перевірка на наявність і усунення всіх помилок, зазначених у п. 5.1.

5.3 Візуальний перегляд програми: остаточне налагодження на предмет виявлення та усунення дрібних помилок.