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 однофакторного рівняння регресії для оцінювання розміру веб-застосунків, реалізованих мовою Java, та розробка програмного забезпечення для його реалізації»

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

Панькін І.Д.
(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

Панькін Ігор Дмитрович
(прізвище, ім'я, по батькові)

1.Тема магістерської роботи «Удосконалення однофакторного рівняння регресії для оцінювання розміру веб-застосунків, реалізованих мовою Java, та розробка програмного забезпечення для його 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 _____ Панькін І.Д.
(підпис) (прізвище та ініціали)

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

РЕФЕРАТ

Панькін Ігор Дмитрович

«Удосконалення однофакторного рівняння регресії для оцінювання розміру веб-застосунків, реалізованих мовою Java, та розробка програмного забезпечення для його 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: 114 стор., 13 табл., 21 рис., 44 використаних джерела, 5 додатків.

Актуальність теми роботи: підвищення достовірності оцінювання розміру веб-застосунків, реалізованих мовою Java, є важливим завданням, яке потребує удосконалення існуючого рівняння регресії, оскільки ефективність оцінювання розміру програмного забезпечення може стати точкою для успіху чи невдачі проекту на ранньому етапі розробки.

Мета та завдання дослідження: підвищення достовірності оцінювання розміру веб-застосунків, реалізованих мовою Java.

Об’єкт дослідження: процес оцінювання розміру веб-застосунків, реалізованих мовою Java.

Предмет дослідження: однофакторне нелінійне рівняння регресії для оцінювання розміру веб-застосунків, реалізованих мовою Java.

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

Наукова новизна одержаних результатів: удосконалено однофакторне нелінійне рівняння регресії для оцінювання розміру веб-застосунків, реалізованих мовою Java, за рахунок використання одновимірного нормалізуючого перетворення Джонсона сім`ї S_B, що дозволило підвищити достовірність оцінювання розміру веб-застосунків, реалізованих мовою Java, в порівнянні з існуючою моделлю.

Практичне значення одержаних результатів: програмне забезпечення для оцінювання розміру веб-застосунків, реалізованих мовою Java.

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: нелінійна регресія, рівняння регресії, нормалізуюче перетворення Джонсона, розмір веб-застосунків, Java.

ABSTRACT

Pankin Ihor Dmytrovych

«Improving the univariate regression equation for size estimation of web applications implemented in Java, and development the software for it`s 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: 114 pages of typewritten text, 13 tables, 21 figures, a list of references from 44 names, 5 appendices.

Relevance of the theme: Improving the reliability of estimating the size of web applications implemented in Java is an important task that requires improvement of the existing regression equation, because the effectiveness of estimating the size of the software can be a point for success or failure of the project at an early stage of development.

The purpose and objectives of the study: increasing the reliability of estimating the size of web applications implemented in Java.

Object of study: the process of estimating the size of web applications implemented in Java.

Subject of study: a univariate nonlinear regression equation for estimating the size of web applications implemented in Java.

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 non-linear regression equation for estimating of the size of web applications implemented in Java by using the normalizing transformation of the SB Johnson family, which made it possible to increase the accuracy of estimating of size of web applications implemented in Java compared to existing model.

The practical significance of the obtained results: the software for estimating of size of web applications implemented in Java.

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: nonlinear regression, regression equation, Johnson normalizing transformation, size of web applications, Java.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ВВ	випадкова величина
ЕД	емпіричні дані
ПЗ	програмне забезпечення
COCOMO	Constructive Cost Model
FPA	Function Point Analysis
ISBSG	International Software Benchmarking Standard Group
LOC	Lines Of Code
MMRE	Mean of Magnitude of Relative Errors
RUP	Rational Unified Process
UCP	Use Case Points
UML	Unified Modeling Language

ВСТУП

Актуальність теми. На сьогоднішній день все більша кількість компаній переводить свою роботу в мережу інтернет та на віддалений режим роботи. Це пов'язано з бажанням оптимізувати бізнес-процеси та скоротити витрати. Тому постійно збільшується як розробка ПЗ на замовлення компаній взагалі, так і розробка у вигляді веб-застосунків. Крім того, мова програмування Java залишається в трійці лідерів серед використовуваних мов програмування в світі та в Україні, хоча й втратила перше місце в порівнянні з листопадом минулого року [1, 2].

А оскільки у компаній-розробників ПЗ теж є нагальна необхідність зменшити свої витрати, то отримання достовірної оцінки розміру майбутнього ПЗ на ранніх стадіях розробки, і, як наслідок, достовірна оцінка трудомісткості та тривалості розробки, є актуальною задачею для компаній-розробників ПЗ.

Розмір ПЗ є відправною точкою для прогнозування трудомісткості та тривалості розробки у різних моделях, наприклад, COCOMO або ISBSG, а отже, і зусиль, ресурсів та вартості, необхідних для розробки ПЗ [3].

Для оцінювання розміру ПЗ використовують різні методики, які ґрунтуються на технічних, функціональних або інших аспектах розробки. Але не існує єдиного вимірювання або набору метрик та показників для точного оцінювання розміру ПЗ. Самі моделі оцінювання розміру ПЗ бувають на основі експертних оцінок або функціональних точок, варіантів використання, аналогові, параметричні або регресійні [4]. Але найчастіше використовується методика визначення розміру ПЗ – підрахунок кількості рядків вихідного коду програми – в залежності від метрик, отриманих на основі діаграми класів. Для ПЗ, яке розробляється згідно об'єктно-орієнтованої методології розробки, найчастіше для цієї цілі використовують загальну кількість класів ПЗ.

Згідно з [5], із метрик, які отримані на основі діаграми класів, будують нелінійну регресійну модель для негаусівських даних для оцінки розміру ПЗ,

використовуючи нормалізуюче перетворення Джонсона. Це дозволяє отримати кращі результати порівняно з іншими регресійними моделями.

Таким чином, актуальність проблеми підвищення достовірності оцінювання розміру веб-застосунків, реалізованих мовою Java, в даний час є важливим завданням, яке потребує удосконалення існуючого рівняння регресії, оскільки ефективність оцінювання розміру ПЗ може стати точкою для успіху чи навпаки невдачі проекту на ранньому етапі розробки.

Мета і завдання дослідження. Метою роботи є підвищення достовірності оцінювання розміру веб-застосунків, реалізованих мовою Java, та розробка відповідного ПЗ.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

– проаналізувати існуючі методи та моделі для оцінювання розміру веб-застосунків, реалізованих мовою Java, та порівняти їх;

– обґрунтувати необхідність удосконалення рівняння регресії для оцінювання розміру веб-застосунків, реалізованих мовою Java;

– удосконалити однофакторне рівняння регресії для оцінювання розміру веб-застосунків, реалізованих мовою Java;

– дослідити джерела з відкритим вихідним кодом та визначити веб-застосунки, реалізовані мовою Java, які можуть бути використані для перевірки побудованого рівняння регресії;

– отримати необхідні метрики з кожного проекту;

– перевірити емпіричні дані на викиди;

– нормалізувати отримані емпіричні дані, використовуючи нормалізуюче перетворення Джонсона;

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

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

– розробити ПЗ для оцінювання розміру веб-застосунків, реалізованих мовою Java.

Об’єктом дослідження є процес оцінювання розміру веб-застосунків, реалізованих мовою Java.

Предметом дослідження є однофакторне нелінійне рівняння регресії для оцінювання розміру веб-застосунків, реалізованих мовою Java.

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: удосконалено однофакторне нелінійне рівняння регресії для оцінювання розміру веб-застосунків, реалізованих мовою Java, за рахунок використання одновимірного нормалізуючого перетворення Джонсона сім`ї S_B, що дозволило підвищити достовірність оцінювання розміру веб-застосунків, реалізованих мовою Java, в порівнянні з існуючою моделлю.

Практичне значення одержаних результатів. ПЗ для оцінювання розміру веб-застосунків, реалізованих мовою Java.

Особистий внесок здобувача. Кваліфікаційна робота є самостійно виконаною працею. Усі результати, викладені у роботі, отримані автором особисто. У роботі, яка опублікована у співавторстві з керівником кваліфікаційної роботи [6], здобувачеві належить удосконалення однофакторного рівняння регресії для оцінювання розміру веб-застосунків, реалізованих мовою Java.

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 МЕТОДІВ ТА МОДЕЛЕЙ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ВЕБ-ЗАСТОСУНКІВ, РЕАЛІЗОВАНИХ МОВОЮ JAVA

1.1 Метрики для оцінювання розміру програмного забезпечення

Розмір ПЗ являє собою один з найцікавіших, але у той же час найскладніших, внутрішніх атрибутів, який використовується в різних моделях для прогнозування вартості, зусиль, ресурсів, необхідних для розробки та впровадження ПЗ [3].

Розмір ПЗ є одним з найвагоміших факторів в управлінні процесом розробки ПЗ. Доведено, що розмір ПЗ корелює з витратами, зусиллями та ресурсами, необхідними на його розробку. Також інформацію, отриману у результаті оцінювання розміру ПЗ, можна використати для прогнозування зусиль розробки ПЗ за моделями COCOMO або ISBSG, які використовуються для надійного прогнозування різних параметрів, пов’язаних з проектом, та оцінюванням витрат. В якості «розміру» веб-застосунків, реалізованих мовою Java, розуміється кількість рядків вихідного коду цього веб-застосунку.

Існує багато методів для оцінювання розміру ПЗ, деякі з них базуються на кількості строк коду, інші методи обчислюють розмір з функціональних, технічних або інших аспектів. Але більшість з них стають непридатними до використання через відсутність даних, ресурсів або експертних навичок в цій галузі.

Більшість методів для оцінювання розміру ПЗ походять від методу аналізу функціональних точок FPA. Інший підхід полягає в тому, щоб провести функціональне вимірювання, щоб виразити функціональність у кількості, що представляє розмір. Деякі методи визначення розміру ПЗ включають оцінювання на основі варіантів використання. Історично найпоширенішою та найбільш вживаною методологією визначення розміру ПЗ є підрахунок

кількості рядків коду, написаних у вихідному коді програми. Однак не існує єдиного вимірювання, набору метрик та показників для оцінювання розміру ПЗ.

Оцінювання розміру ПЗ відіграє важливу роль у практичних завданнях з управління розробкою та впровадженням ПЗ. Кількість рядків коду ПЗ залежить від ряду показників, які називають метриками ПЗ, що в тому чи іншому випадку впливають на кінцевий результат розробки та впровадження ПЗ.

Метрики ПЗ кількісно визначають різні властивості програмних продуктів та програмних процесів у вигляді чисельного відображення. Мета полягає у виведенні одного або декількох значень особливостей ПЗ, що дає змогу порівнювати ці значення з подібними проектами, зі специфічними стандартами, які притаманні даній компанії. З отриманих результатів можна прийти до висновку щодо якості ПЗ та всього процесу розробки ПЗ, а також, якщо необхідно, вжити подальших заходів.

Часто метрики ПЗ використовують у методах та моделях для оцінювання витрат і зусиль, таких, як Function Point, метод Delphi, COCOMO або ISBSG.

Стандарт IEEE 1061 подає одне з визначень терміну «Метрика ПЗ» як показник якості ПЗ і тлумачить його таким чином: метрика якості ПЗ – це функція, яка відображає програмний блок в числовому значенні [7].

Розрізняють статичні і динамічні метрики ПЗ для зручності використання. Статичні метрики ПЗ враховують показники дизайну, самої програми або документації. Вони також враховують такі аспекти, як складність і простота обслуговування. Динамічні метрики ПЗ використовуються для оцінки продуктивності та надійності ПЗ. Прикладами є необхідний час виконання або кількість помилок, що виникли під час виконання.

Відмінності між процедурною і об'єктно-орієнтованою методиками розробки ПЗ враховуються шляхом подальшої диференціації в звичайні та об'єктно-орієнтовані метрики продукту [7].

Серед основних метрик ПЗ, які використовують для оцінювання розміру ПЗ, виділяють кількість рядків коду. Існує декілька методик підрахунку кількості рядків коду, як основні можна виділити такі [8]:

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

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

Якщо мова підтримує розміщення кількох команд в одному рядку, то один фізичний рядок повинен бути врахований як кілька логічних, якщо він містить більше однієї команди мови.

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 з коментарями, 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 і т.д.

Метрики, засновані на аналізі кількості рядків і синтаксичних елементів вихідного коду програми, були запропоновані багатьма відомими вченими, наприклад М. Холстедом.

Крім того, всесвітньо відома модель COCOMO II заснована на такій метриці ПЗ, як кількість рядків коду LOC. Ця модель часто використовується для надійного прогнозування різних параметрів, пов'язаних з проектом та оцінюванням витрат.

1.2 Існуючі методи та моделі для оцінювання розміру веб-застосунків, реалізованих мовою Java

Основними причинами для оцінювання розміру ПЗ є:

– оцінювання вартості майбутніх проектів;

– оцінювання продуктивності застосування нових засобів і методів;

– визначення якості існуючого ПЗ або його розробки / експлуатації;

– прогнозування якості ПЗ або його розробки / експлуатації у майбутньому;

– поліпшення якості ПЗ;

– прогноз майбутніх потреб у персоналі;

– передбачення і скорочення майбутніх потреб у технічному обслуговуванні.

Основні моделі оцінювання розміру ПЗ поділяють на п'ять категорій: аналогові, регресійні, моделі на основі експертних оцінок, моделі, які базуються на функціональних точках, параметричні моделі [9]. Найбільш перспективними є комбіновані методи, оскільки при суміщенні підходів може виникнути принципово нове рішення, яке дозволить підвищити достовірність оцінювання або розширити сферу застосування.

Аналіз функціональних точок FPA – це метод вимірювання розміру ПЗ з точки зору користувачів системи. Метод був розроблений Аланом Альбрехтом ще в середині 1970-х років, вперше опублікований в 1979 році.

Метод UCP являє собою оцінку розміру проектів на основі діаграм UML і методології RUP. Як і багато інших сучасні методів оцінки, 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.

Регресійні моделі для оцінювання кількості рядків коду можна будувати як без урахування мови програмування (наприклад, по даним розроблених проектів однією фірмою-розробником), так і окремо для різних мов; як взагалі

по всім типам програм, так і розділяючи програми за призначенням. Зазвичай будують регресійні моделі з урахуванням мови програмування, такі моделі виявляються більш достовірними тому, що враховують конкретні особливості різних мов. Побудовані такі моделі для мов програмування C++, JavaScript, PHP, VB, Java.

Якщо розглядати мову програмування Java, можна привести такі регресійні моделі, як наведені, наприклад у [10 – 13].

Методи побудови рівнянь регресії

В загальному вигляді рівняння регресії може бути представлено у наступному вигляді:

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

де y – залежна змінна;

$f(x)$ – функція, яка визначає вид рівняння регресії;

x – незалежна змінна.

Однак при побудові рівнянь регресії виникає ряд особливостей:

– отримуємо нелінійне рівняння регресії, якщо ВВ, що входять до нього, не підпорядковуються гаусівському закону розподілу;

– отримуємо множинну регресію, якщо залежна ВВ залежить одночасно від двох і більше факторів, що зазвичай і буває при вирішенні практичних завдань.

Щоб побудувати нелінійне рівняння регресії, можна скористатися одним з трьох наведених методів:

– метод простого перебору вимагає завдання різних видів рівняння регресії і вибору найкращого наближення із заданих за певним критерієм. Це вимагає досить великої кількості обчислень і не завжди призводить до отримання найкращого рішення. Ці недоліки роблять даний метод неефективним [14].

– метод лінеаризуючих перетворень передбачає перехід від нелінійної регресії до лінійної шляхом заміни вхідних змінних і коефіцієнтів, проте не завжди можливо підібрати таку заміну [15 – 17]. Крім того, така заміна призводить до спрощення рівняння регресії та втрати інформації, пов'язаної з нелінійністю.

– метод нормалізуючих перетворень передбачає пошук таких перетворень, за допомогою яких можна здійснити перехід від вхідних негаусівських ВВ до гаусівських ВВ [16].

Суть даного метода можна представити трьома кроками. На першому кроці обирають нормалізуюче перетворення, за допомогою якого здійснюють перехід від вхідних негаусівских ВВ до гаусівських ВВ. На другому кроці для отриманих гаусівських ВВ будують лінійне рівняння регресії, довірчий інтервал та інтервал прогнозування. На третьому кроці за допомогою зворотного нормалізуючого перетворення переходять до нелінійного рівняння регресії вхідних негаусівских ВВ та відповідних інтервалів [18].

1.3 Способи удосконалення рівняння регресії для оцінювання розміру веб-застосунків, реалізованих мовою Java

ЕД для оцінювання розміру розміру веб-застосунків, реалізованих мовою Java, зазвичай, не підпорядковуються гаусівському розподілу. Це може призвести до помилок у розрахунках та негативно впливає на достовірність отриманих результатів з оцінювання розміру веб-застосунків, реалізованих мовою Java. Щоб уникнути цих помилок перед тим, як будувати рівняння регресії, потрібно скористатися методом нормалізуючих перетворень та нормалізувати ЕД.

Використання методу нормалізуючих перетворень дозволить перейти до гаусівських ВВ, побудувати лінійне рівняння регресії, далі побудувати довірчий інтервал та інтервал прогнозування лінійного рівняння регресії, далі

застосовуючи зворотнє нормалізуюче перетворення побудувати нелінійне рівняння регресії та довірчий інтервал і інтервал прогнозування нелінійного рівняння регресії.

У разі гаусівських ВВ можна побудувати лінійне рівняння регресії у наступному вигляді:

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

де b_1, b_0 – коефіцієнти лінійної регресії, які можна знайти, використовуючи методом найменших квадратів:

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

У разі гаусівських ВВ довірчий інтервал лінійного рівняння регресії можна представити у наступному вигляді [19]:

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

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

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

α – рівень значущості;

n – кількість значень ВВ у вибірці;

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

Інтервал прогнозування лінійного рівняння регресії можна представити у наступному вигляді:

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

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

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

α – рівень значущості;

n – кількість значень ВВ у вибірці;

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

Для нормалізації негаусівських ВВ можуть використовуватися різні нормалізуючи перетворення: перетворення на основі логарифмів – десяткового або натурального, перетворення Бокса-Кокса, перетворення Джонсона та інші. У даній роботі в якості нормалізуючого перетворення буде використовуватися одновимірне чотирьохпараметричне перетворення Джонсона тому, що воно дає кращі результати у порівнянні з іншими відомими перетвореннями.

Тим самим, використовуючи метод нормалізуючих перетворень для удосконалення рівняння регресії для оцінювання розміру веб-застосунків, реалізованих мовою Java, отримаємо більш достовірну оцінку розміру веб-застосунків, реалізованих мовою Java.

2 УДОСКОНАЛЕННЯ РІВНЯННЯ РЕГРЕСІЇ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ВЕБ-ЗАСТОСУНКІВ, РЕАЛІЗОВАНИХ МОВОЮ JAVA

2.1 Удосконалення рівняння регресії для оцінювання розміру веб-застосунків, реалізованих мовою Java, із застосуванням нормалізуючих перетворень

Застосування методу нормалізуючих перетворень та перехід до гаусівської ВВ дозволяє перейти до побудови лінійного рівняння регресії, а потім, застосувавши зворотне перетворення перейти до побудови нелінійного рівняння регресії [20]. Використаємо одновимірне перетворення Джонсона для нормалізації негаусівських ЕД.

Нормалізація на основі сімей розподілу Джонсона – це універсальний вид нормалізації, заснований на такому перетворенні вхідної ВВ *X*, яке дозволить розглядати результат перетворення як ВВ, розподілену за гаусівським законом. Одним з позитивних аспектів даного підходу є те, що значення емпіричної функції розподілу ВВ *X* обчислюються як значення функції нормального розподілу [21].

У загальному випадку одновимірне нормалізуюче перетворення Джонсона має вигляд [22]:

$$z = \gamma + \eta h(x, \varphi, \lambda), \quad \eta > 0, -\infty < \gamma < \infty, \lambda > 0, -\infty < \varphi < \infty, \tag{2.1}$$

де z – нормована нормально розподілена ВВ;

γ, η, φ, λ – параметри перетворення;

x – ВВ, яка нормалізується;

h – функція певної сім'ї розподілу Джонсона:

$$h_1(x,\varphi,\lambda)=\ln(\tilde{x}),x>\varphi,$$
$$h_2(x,\varphi,\lambda)=\ln\left(\frac{\tilde{x}}{1-\tilde{x}}\right),\varphi<x<\varphi+\lambda, \quad (2.2)$$
$$h_3(x,\varphi,\lambda)=Arsh(\tilde{x}),-\infty\le x\le+\infty.$$

Сім'ї функцій h_1 відповідає логарифмічно нормальний розподіл S_L Джонсона, сім'ї функцій h_2 відповідає сім'я розподілів S_B Джонсона, сім'ї функцій h_3 відповідає сім'я розподілів S_U Джонсона, $\tilde{x}=\frac{x-\varphi}{\lambda}$.

Перетворення (2.1) має зворотне перетворення [23]:

$$x=\varphi+\lambda h^{-1}(z,\gamma,\eta),$$
$$\eta>0,-\infty<\gamma<\infty,\lambda>0,-\infty<\varphi<\infty, \quad (2.3)$$

де x – ВВ з розподілом Джонсона;

γ, η, φ, λ – параметри перетворення Джонсона;

z – нормально розподілена ВВ з математичним сподіванням нуль і дисперсією одиниця;

h^{-1} – функції певної сім'ї розподілу Джонсона:

$$h_1^{-1}(z,\gamma,\eta)=e^{\zeta},$$
$$h_2^{-1}(z,\gamma,\eta)=\frac{1}{1+e^{-\zeta}}, \quad (2.4)$$
$$h_3^{-1}(z,\gamma,\eta)=\frac{e^{\zeta}-e^{-\zeta}}{2}.$$

Функція h_1^{-1} – для сім'ї S_L Джонсона, функція h_2^{-1} – для сім'ї S_B Джонсона, функція h_3^{-1} – для сім'ї S_U Джонсона, $\zeta=\frac{z-\gamma}{\eta}$.

Існує три сім'ї розподілів Джонсона – S_L, S_B, S_U – отриманих шляхом різних нелінійних перетворень гаусівської нормованої щільності розподілу ймовірностей. Конкретна сім'я розподілу Джонсона вибирається виходячи із значень квадрата асиметрії A^2 і ексцесу ε вихідної вибірки [24]. Для автоматизованого вибору сім'ї розподілу Джонсона можна також скористатися формулою, наведеною у [25]:

$$\varepsilon(A^2) = 3{,}59 \cdot 10^{-6} A^8 - 4{,}8805 \cdot 10^{-4} A^6 + 4{,}1655 \cdot 10^{-2} A^4 + 1{,}8203 A^2 + 2{,}9658. \quad (2.5)$$

Якщо точка з координатами (A^2, ε) знаходиться біля лінії S_L, то можна використовувати розподіл з сім'ї S_L. Якщо точка з координатами (A^2, ε) знаходиться вище лінії S_L, то можна використовувати розподіл з сім'ї S_U, а якщо нижче лінії S_L до лінії критичної області – то розподіл з сім'ї S_B. Якщо ж точка потрапляє у критичну область, то використовувати для нормалізації сім'ї розподілів Джонсона не можна.

Значення невідомих параметрів розподілу Джонсона можна знайти за допомогою методу максимальної правдоподібності.

Перевірку відповідності як ЕД, так і нормалізованих вибірок гаусівському розподілу можна виконати за допомогою критеріїв згоди, наприклад, χ^2 Пірсона або Колмогорова-Смирнова [26]. Будемо використовувати критерій χ^2 Пірсона. Він застосовується для зіставлення емпіричного розподілу з теоретичним та підходить для перевірки гіпотези про будь-який закон розподілу ВВ. Розрахункова формула критерію χ^2 Пірсона наступна [27]:

$$\chi^2 = \sum_{i=1}^{m} \frac{(n_i - np_i)^2}{np_i}, \quad (2.5)$$

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

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

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

Кількість підінтервалів m можна розрахувати за формулами Старджеса $m = 3{,}31 \cdot \lg n + 1$ або Брукса і Карузера $m = 5 \cdot \lg n$ [28].

Отримавши гаусівський набір даних за допомогою нормалізуючого перетворення Джонсона та перевіривши нормальність отриманого розподілу за допомогою критерія χ^2 Пірсона, можна переходити до наступного кроку – побудови лінійного рівняння регресії (1.2).

Далі для лінійного рівняння регресії будуємо довірчий інтервал згідно з (1.4) та інтервал прогнозування згідно з (1.5).

Для побудови нелінійного рівняння регресії використаємо вже побудоване лінійне рівняння регресії (1.2) та зворотне нормалізуюче перетворення Джонсона (2.3) з відповідною сім'єю S_B (2.4):

$$y = \frac{e^c (\lambda_y + \varphi_y) + \varphi_y}{1 + e^c}, \tag{2.6}$$

де: $c = \frac{1}{\eta_y} \cdot \left(b_1 \left[\gamma_x + \eta_x \ln \left(\frac{x - \varphi_x}{\lambda_x + \varphi_x - x} \right) \right] + b_0 - \gamma_y \right)$.

95% довірчий інтервал нелінійного рівняння регресії можна побудувати, використовуючи лінійне рівняння регресії (1.2), t-розподіл Стьюдента та зворотне нормалізуюче перетворення Джонсона (2.3) з відповідною сім'єю S_B (2.4):

$$y = \frac{e^{k_1} (\lambda_y + \varphi_y) + \varphi_y}{1 + e^{k_1}}, \tag{2.7}$$

де: $k_1 = \frac{1}{\eta_y} \cdot \left(b_1 \cdot z_x + b_0 - \gamma_y \pm t(\alpha/2, n-2) \cdot S_{z_y} \cdot \sqrt{\frac{1}{n} + \frac{(z_x - \bar{z}_x)^2}{\sum_{i=1}^{n}(z_{xi} - \bar{z}_x)^2}} \right)$,

$z_x = \gamma_x + \eta_x \ln\left(\frac{x - \phi_x}{\lambda_x + \phi_x - x} \right)$.

Аналогічно знайдемо і інтервал прогнозування нелінійного рівняння регресії. Перехід до 95% інтервалу прогнозування нелінійного рівняння регресії можна здійснити, використовуючи побудований 95% інтервал прогнозування лінійного рівняння регресії за формулою (1.4) та зворотне перетворення Джонсона (2.3) з відповідною сім'єю S_B (2.4).

2.2 Перевірка якості рівняння регресії для оцінювання розміру веб-застосунків, реалізованих мовою Java

Для перевірки якості однофакторного рівняння регресії для оцінювання розміру веб-застосунків, реалізованих мовою Java, використаємо наступні показники:

– коефіцієнт детермінації R^2 [29]:

$$R^2 = 1 - \left(\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \right), \qquad (2.8)$$

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 ВВ y;

n – кількість значень у вибірці.

Коефіцієнт детермінації R^2 характеризує частку дисперсії, яка обумовлена регресією, в загальній дисперсії показника y та може приймати значення від 0 до 1. Чим ближче значення коефіцієнта R^2 до 1, тим тісніше зв'язок результативної ознаки з досліджуваними факторами.

– середню величину відносної похибки *MMRE*:

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} MRE_i\,, \tag{2.9}$$

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):

$$PRED(0{,}25) = \frac{k}{n}, \tag{2.10}$$

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

Ці параметри також використовуються для порівняння різних рівнянь регресії.

2.3 Перевірка емпіричних даних на викиди

Перше, що потрібно зробити при аналізі отриманих ЕД – перевірити їх на викиди. Гаусівські дані не повинні мати викидів, отже, якщо викиди є, то це може свідчити про те, що дані не розподіляються за гаусівських законом. Якщо викиди будуть знайдені, їх потрібно видалити з вибірки та провести повторну перевірку. Лише після того, як усі знайдені викиди будуть видалені з вибірки, можна переходити до наступного кроку аналізу даних [30, 31].

Для знаходження викидів у ЕД використаємо квадрат відстані Махаланобіса:

$$d_i^2 = (Z_i - \bar{Z})^T S_N{}^{-1} (Z_i - \bar{Z}), \tag{2.11}$$

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

S_N – матриця кореляції вибірки, яка визначається за формулою:

$$S_N = \frac{1}{N} \sum_{i=1}^{N} (Z_i - \bar{Z})(Z_i - \bar{Z})^T . \tag{2.12}$$

Точка, що має найбільшу відстань Махаланобіса до решти безлічі заданих точок, вважається що має найбільшу значимість, тому що вона має найбільший вплив на кривизну і на коефіцієнти рівняння регресії.

Розрахувавши d_i^2 можна використати або критерій χ^2 Пірсона, або критерій F Фишера. При використанні критерію F Фишера потрібно додатково розрахувати T_S (Test statistic). T_S для d_i^2 може бути розрахована за формулою:

$$Ts = \frac{(N-m)Nd_i^2}{(N^2-1)m}, \tag{2.13}$$

та має апроксимований розподіл F з m та $N-m$ ступенями свободи.

У данній роботі використаємо критерій F Фишера. Тестова статистика для квадрату відстані Махаланобіса порівнюється з квантилем розподілу $F_{m,N-m,\alpha}$, де α –рівень значущості, який у даному випадку дорівнюватиме 0,05. Точки, для яких значення T_S, розраховане за формулою (2.13), буде більше, ніж квантиль розподілу F вважаються викидами, і ці значення видаляються з набору даних.

Після усунення викидів отриманий скоригований набір даних знову нормалізується та для нього знаходиться квадрат відстані Махаланобіса за формулою (2.11) та тестова статистика за формулою (2.13). Процедура ітераційно повторюється до тих пір, поки всі значення T_S не будуть менші або дорівнюватимуть квантилю розподілу F.

2.4 Побудова удосконаленого однофакторного рівняння регресії для оцінювання розміру веб-застосунків, реалізованих мовою Java, із застосуванням нормалізуючого перетворення Джонсона

Для оцінювання розміру веб-застосунків, реалізованих мовою Java, однією з основних задач є побудова відповідного рівняння регресії , яке буде нелінійним. Також побудувавши довірчий інтервал та інтервал прогнозування нелінійного рівняння регресії, можна підвищити достовірність оцінювання розміру веб-застосунків, реалізованих мовою Java.

Під час наукового стажування були виконані задачі з підготовки кваліфікаційної роботи, а саме: було знайдено інформацію про 45 веб-застосунків, реалізованих мовою Java, із яких з роботи [13] було взято 30 проектів, на інформаційному ресурсі GitHub [32] було знайдено 15 проектів.

Згідно із [5], із метрик, отриманих на основі діаграм класів можна побудувати нелінійне рівняння регресії для багатовимірних негаусівських даних для оцінювання розміру ПЗ, використовуючи багатовимірне нормалізуюче перетворення Джонсона та отримати непогані результати у порівнянні з іншими регресійними моделями, як лінійними, так і нелінійними.

Але побудова рівняння регресії на основі декількох метрик достатньо складна для людей, які до цього не мали досвіду у цій галузі. Тому, на основі отриманих ЕД та аналізу існуючих регресійних моделей оцінювання розміру ПЗ, було вирішено будувати рівняння регресії для оцінювання розміру веб-застосунків, реалізованих мовою Java, на основі двох метрик: X – загальної

кількості класів (NC) та Y – кількості рядків коду (LOC) та використовуючи одновимірне чотирьохпараметричне нормалізуюче перетворення Джонсона.

Метрики, потрібні для побудови нелінійного рівняння регресії були зняті з проектів, інформація про які наведена у таблиці 2.1, за допомогою плагіну SonarQube [33], який дозволяє отримати, між іншим, такі дані на сторінці «Розмір»: кількість рядків коду, рядків нового коду, виразів, функцій, класів, файлів, директорій.

Таблиця 2.1 – Метрики веб-застосунків, реалізованих мовою Java (фрагмент даних)

№ п/п	Найменування проекта	NC	LOC
31	easybuggy	103	4609
32	GDAV-webapp	10	3286
33	TravelWebApplication-Virtugo	12	3808
34	Web Test	18	1256
35	WebBased-TemperatureLogger	25	2057
36	WebAplication	52	6278
37	simpleorganizer	10	1028
38	SimpleJavaWebApp	28	3459
39	retroclinic-java-aplication-web	11	769
40	JavaWebAplication	14	938
41	HelloJSP	68	9471
42	greenhouse	5	349
43	epam-webapp	48	8746
44	CRUD_JavaWebAplication	13	1148
45	basejava	16	2089

Перевіримо отримані ЕД на викиди, використовуючи квадрат відстані Махаланобіса, який знаходиться за (2.11). Проведемо перевірку серед 45 наборів даних. Розрахунок показав, що в даному наборі даних присутні три викиди: точки 31, 32 та 33, значення T_S для яких, розраховане за (2.13), перевищує квантіль розподілу F, який дорівнює 3,2145 при m=2, α=0,05. Видалимо ці точки з набору даних. У скоригованій вибірці, яка містить 42 набори даних, викиди відсутні. У таблиці 2.2 наведено, значення ЕД X та Y, нормалізовані значення Z_X та Z_Y, а також значення T_S для d_i^2 (фрагмент даних).

Таблиця 2.2 – Метрики веб-застосунків, реалізованих мовою Java, нормалізовані дані та значення T_S для d_i^2 (фрагмент даних)

№ п/п	X	Y	Z_X	Z_Y	T_S для d_i^2
31	**103**	**4609**	**2,3706**	**0,9661**	**4,2481**
32	**10**	**3286**	**-1,0694**	**0,6375**	**4,0955**
33	**12**	**3808**	**-0,7980**	**0,7787**	**3,4730**
34	18	1256	-0,1969	-0,2648	0,0350
35	25	2057	0,2883	0,1993	0,0429
36	52	6278	1,3666	1,2868	0,9520
37	10	1028	-1,0694	-0,4604	0,8320
38	28	3459	0,4555	0,6864	0,2490
39	11	769	-0,9275	-0,7586	0,4209
40	14	938	-0,5692	-0,5523	0,1689
41	68	9471	1,7609	1,7751	1,6750
42	5	349	-2,1108	-1,8048	2,1853
43	48	8746	1,2490	1,6722	1,3929
44	13	1148	-0,6791	-0,3518	0,2922
45	16	2089	-0,3713	0,2137	0,4813

Емпіричний розподіл вибірки X зображено на рис. 2.1, а емпіричний розподіл вибірки Y зображено на рис. 2.2.

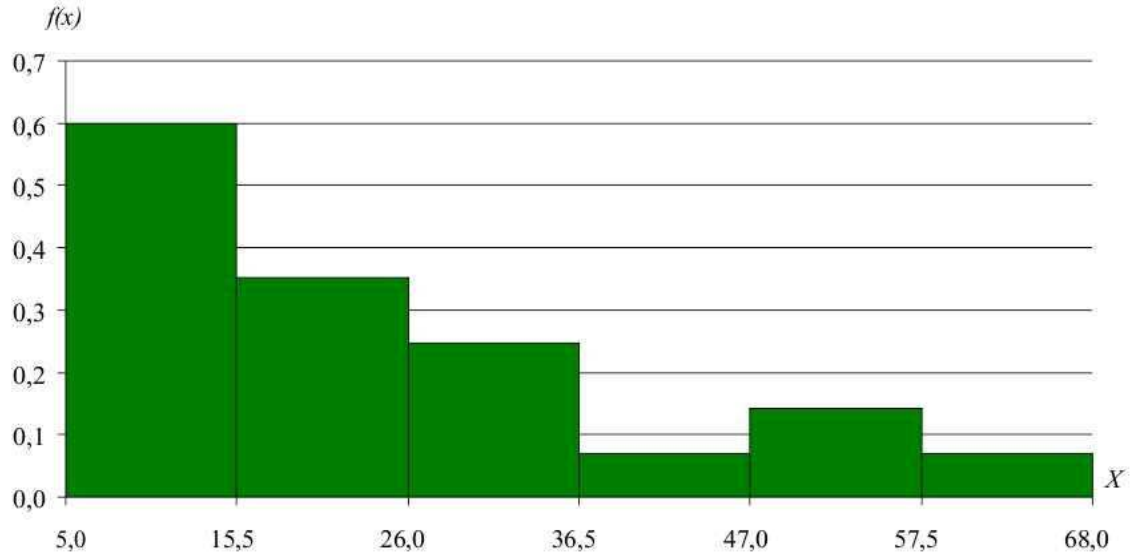

Рисунок 2.1 – Емпіричний розподіл вибірки X

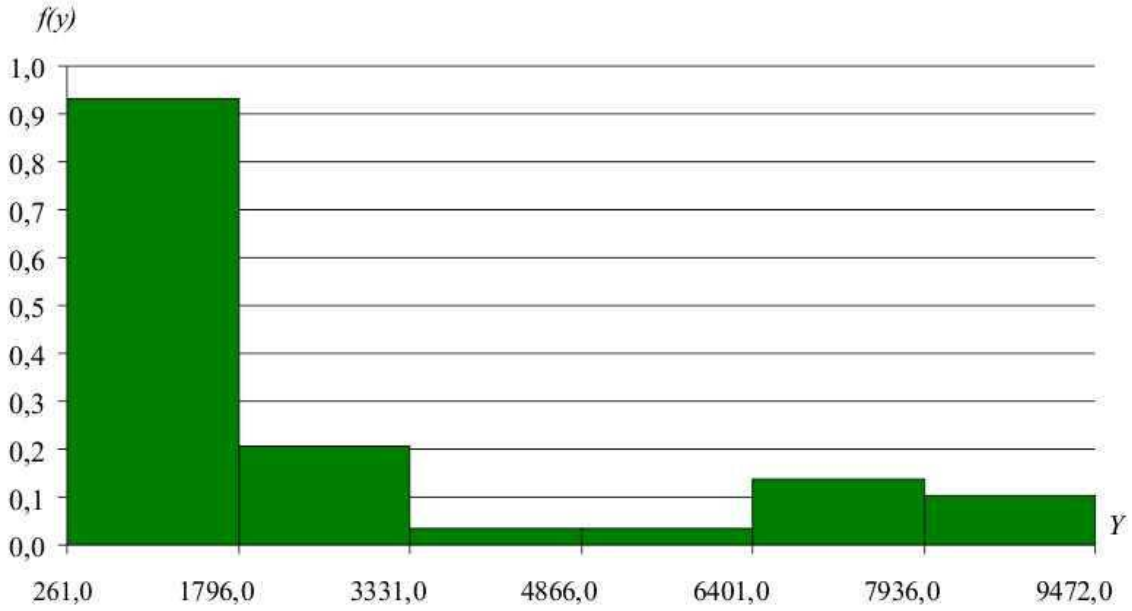

Рисунок 2.2 – Емпіричний розподіл вибірки *Y*

Отримані вибірки *X* та *Y* не відповідають гаусівському закону розподілу: χ^2_X=20,20 та χ^2_Y=63,67 при критичному значенні χ^2=7,81 для довірчої ймовірності 0,95.

Для нормалізації ЕД виходячи із значень квадрата асиметрії A^2 і ексцесу ε вибірок була обрана сім'я розподілів S_B Джонсона згідно з (2.5). Саму нормалізацію даних для вибірок *X* та *Y* проводимо з використанням нормалізуючого перетворення Джонсона (2.1). Значення невідомих параметрів γ, η, φ, λ для перетворення знаходимо за допомогою методу максимальної правдоподібності. Параметри перетворення Джонсона для ВВ *X* та *Y* наведені в табл. 2.3.

Таблиця 2.3 – Параметри перетворення Джонсона для ВВ *X* та *Y*

Параметр	ВВ *X*	ВВ *Y*
γ	8,9300	7,3257
η	1,7027	0,8761
φ	-1,5648	178,09
λ	4256,32	5678397

Отримані нормалізовані вибірки Z_X та Z_Y відповідають нормальному закону розподілу: χ^2_{Zx}=5,28 та χ^2_{Zy}=5,24 при критичному значенні χ^2=7,81 для довірчої ймовірності 0,95.

Гістограми розподілу нормалізованих значень для вибірок Z_X та Z_Y зображені на рис. 2.3 та 2.4 відповідно.

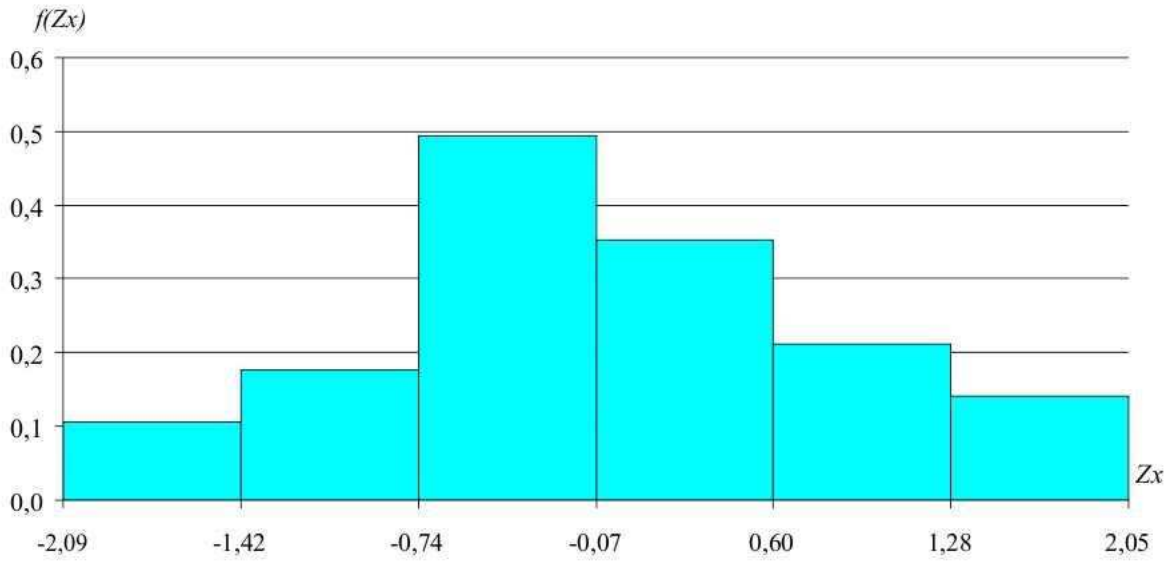

Рисунок 2.3 – Гістограма розподілу нормалізованих значень для вибірки Z_X

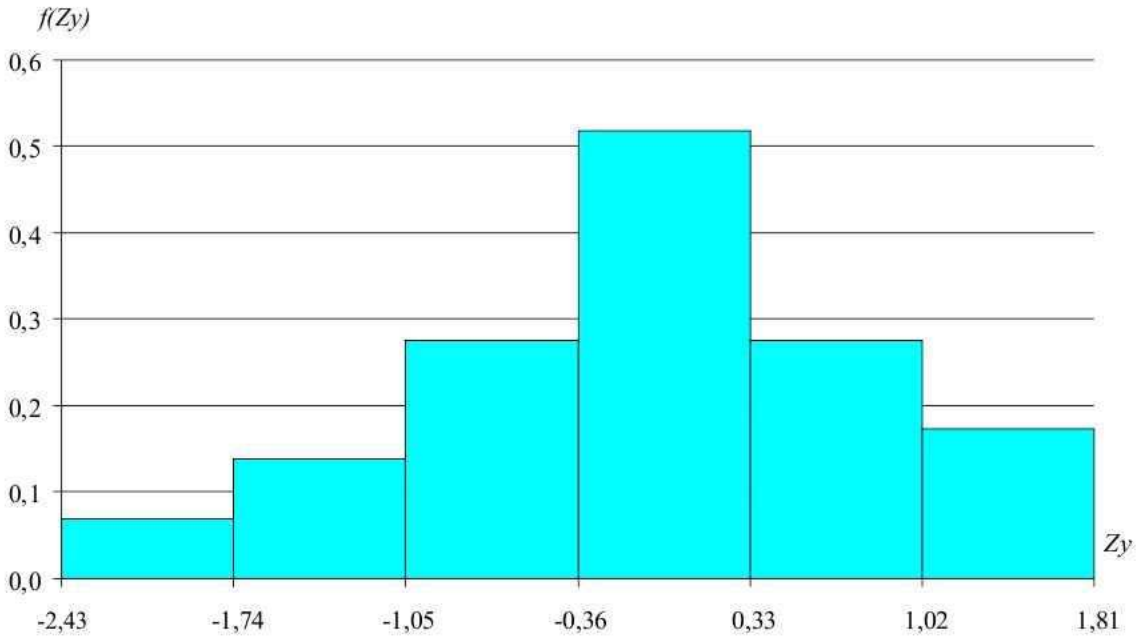

Рисунок 2.4 – Гістограма розподілу нормалізованих значень для вибірки Z_Y

Далі будуємо лінійне рівняння регресії для нормалізованих даних згідно з (1.2), коефіцієнти знаходимо за допомогою методу найменших квадратів (1.3): $z_y = 0{,}9034 z_x - 0{,}000002$. Також будуємо довірчий інтервал лінійного рівняння регресії згідно з (1.4) та інтервал прогнозування лінійного рівняння регресії згідно з (1.5), які разом із самим рівнянням регресії та нормалізованими даними зображені на рис. 2.5.

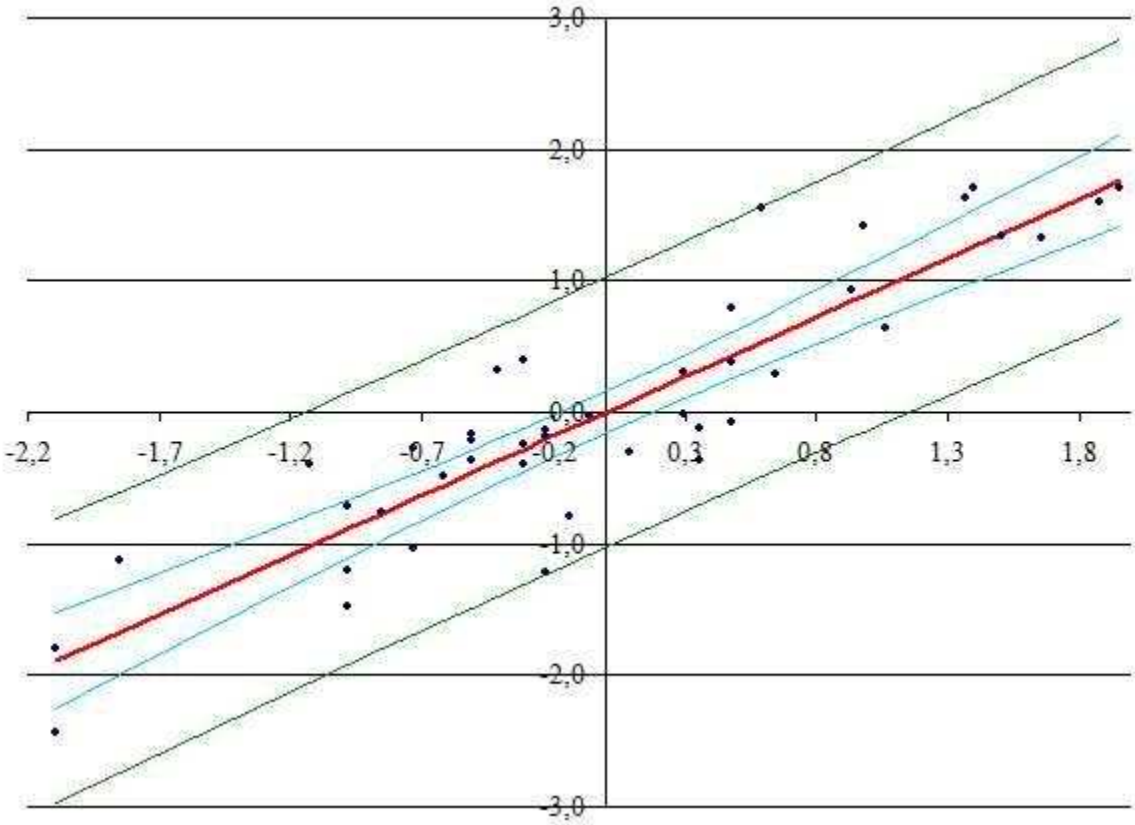

Рисунок 2.5 – Нормалізовані дані, лінійне рівняння регресії, довірчий інтервал та інтервал прогнозування лінійного рівняння регресії

Як видно із рис. 2.5, усі нормалізовані дані знаходяться всередині інтервалу прогнозування. Тобто, нормалізовані дані підпорядковуються гаусівському закону та усі викиди були видалені вірно на етапі первинної обробки даних.

Переходимо до ЕД. На основі лінійного рівняння регресії (1.2) та зворотнього нормалізуючого перетворення Джонсона (2.3) будуємо нелінійне рівняння регресії згідно з (2.6): $y = \frac{5678404 \cdot e^{c} + 7{,}33}{1 + e^{c}}$, де $c = 0{,}9287 + 1{,}7565 \cdot \ln\left(\frac{x - 9{,}0096}{4459{,}02 - x}\right)$.

Довірчий інтервал нелінійного рівняння регресії будуємо згідно з (2.7). Аналогічно будуємо інтервал прогнозування нелінійного рівняння регресії, які разом із нелінійним рівнянням регресії та ЕД зображені на рис. 2.6.

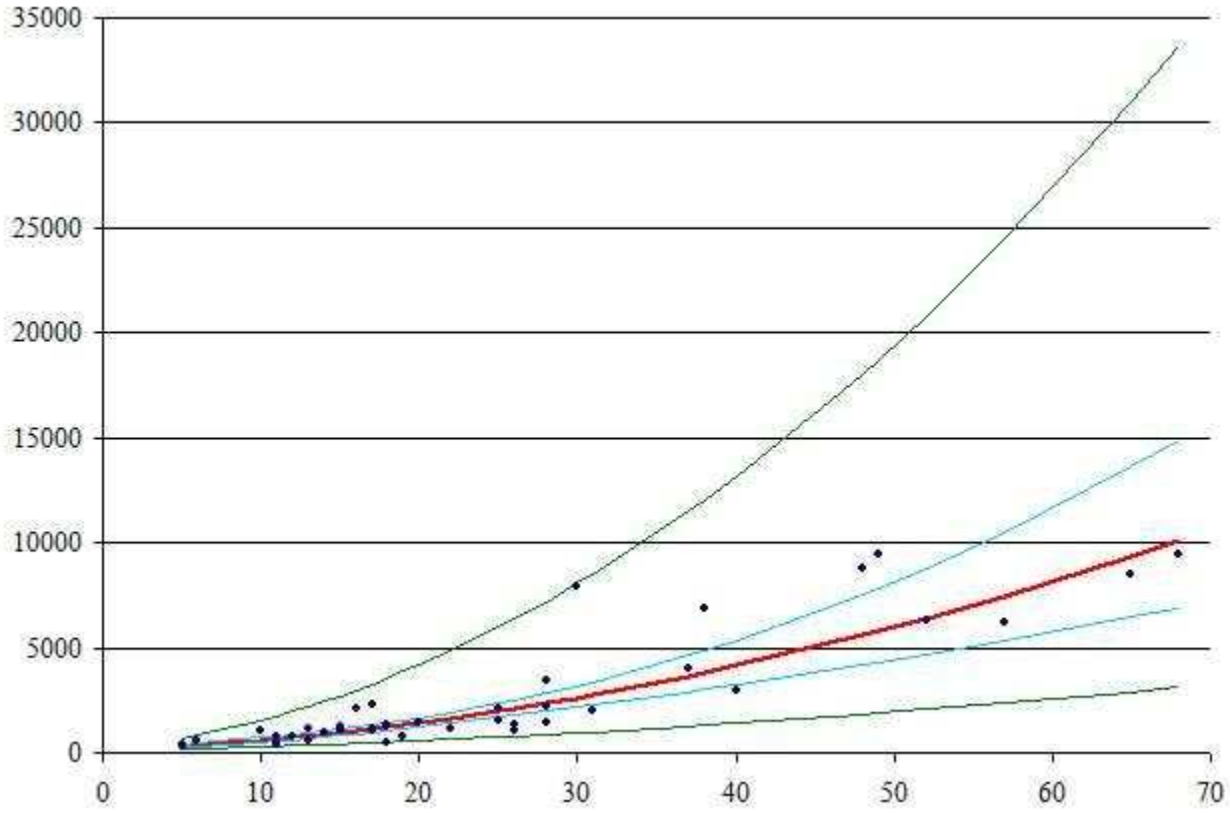

Рисунок 2.6 – ЕД, нелінійне рівняння регресії, довірчий інтервал та інтервал прогнозування нелінійного рівняння регресії

Побудуємо лінійне рівняння регресії в припущенні про нормальність ЕД згідно з (1.2), коефіцієнти рівняння також знайдемо згідно з (1.3) для

порівняння з побудованим удосконаленим нелінійним рівнянням регресії. Лінійне рівняння регресії має вигляд $z_y = 155{,}56 z_x - 1250{,}33$.

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

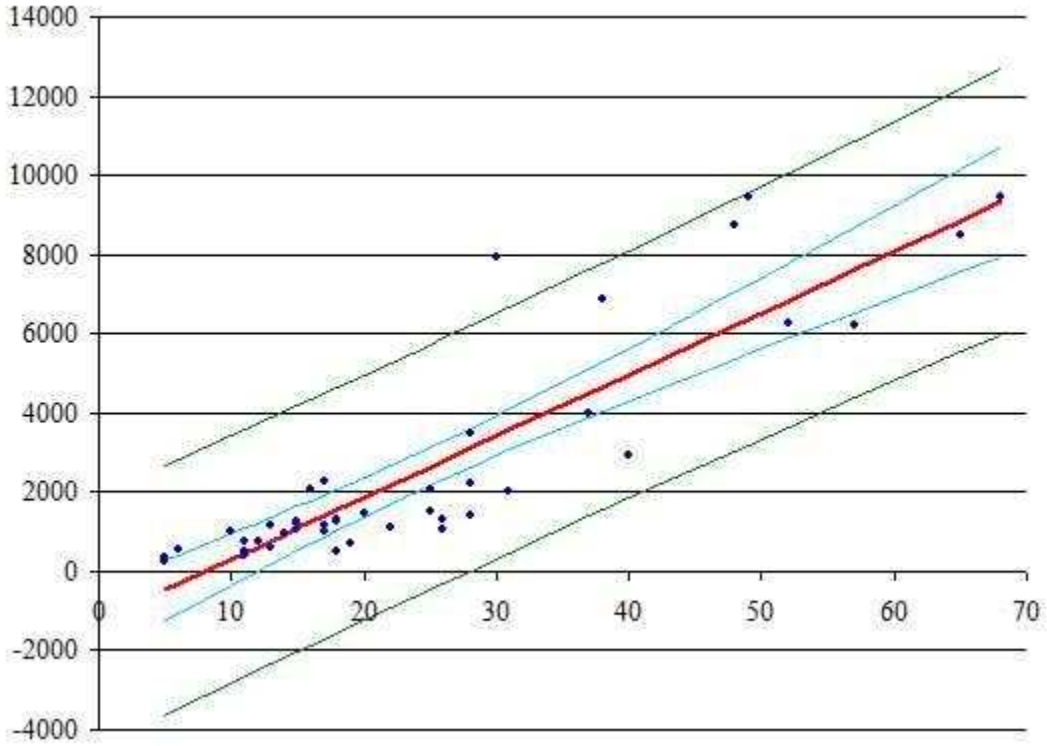

Рисунок 2.7 –ЕД, лінійне рівняння регресії, довірчий інтервал та інтервал прогнозування лінійного рівняння регресії

Як видно із рис. 2.7, не всі ЕД знаходяться всередині інтервалу прогнозування, тобто, ці ЕД не підпорядковуються гаусівському закону розподілу та містять викиди, на відміну від удосконаленого нелінійного рівняння регресії, зображеного рис. 2.6. Також треба відмітити, що лінійне рівняння регресії, нижня границя довірчого інтервалу та нижня границя інтервалу прогнозування мають значення, нижчі нуля. Використовувати для таких ЕД лінійне рівняння регресії в припущенні про нормальність ЕД без застосування методу нормалізуючих перетворень не бажано, оскільки це

призводить до спотворених результатів при оцінюванні розміру веб-застосунків, реалізованих мовою Java.

Перевіримо якість побудованого удосконаленого нелінійного рівняння регресії та порівняємо значення з існуючим лінійним рівнянням. Для перевірки якості використаємо коефіцієнт детермінації R^2, знайдений за формулою (2.8), значення *MMRE*, знайдене за формулою (2.9), та значення *Pred*(0,25), знайдене за формулою (2.10). Ці дані наведено в табл. 2.4.

Таблиця 2.4 – Перевірка якості рівнянь регресії

Параметр	Нелінійне рівняння регресії	Лінійне рівняння регресії
R^2	0,8162	0,7679
MMRE	0,2199	0,5850
Pred(0,25)	0,7524	0,3810

Як видно із табл. 2.4, удосконалене нелінійне рівняння регресії, яке побудовано з використанням нормалізуючого перетворення сім'ї S_B Джонсона, має кращі параметри, які повністю задовольняють вимогам до використаних показників.

Порівняємо отримані значення для границь довірчого інтервалу та інтервалу прогнозування удосконаленого нелінійного рівняння регресії із результатами для існуючого лінійного рівняння регресії. Отримані результати для границь довірчого інтервалу наведено у табл. 2.5, для границь інтервалу прогнозування – у табл. 2.6. У таблицях наведено фрагмент даних.

Як видно із табл. 2.5, нижня границя довірчого інтервалу нелінійного рівняння регресії, яке побудовано з використанням нормалізуючого перетворення сім'ї S_B Джонсона, більша нуля. Нижня границя довірчого інтервалу лінійного рівняння регресії має значення, менші нуля. Довжина довірчого інтервалу нелінійного рівняння регресії менша за відповідну довжину лінійного рівняння регресії для більшості наборів даних.

Таблиця 2.5 – Границі довірчого інтервалу рівнянь регресії (фрагмент даних)

№ про-екта	Нелінійне рівняння регресії			Лінійне рівняння регресії		
	нижня границя	верхня границя	довжина	нижня границя	верхня границя	довжина
31	1052,84	1438,57	385,73	2167,07	3110,51	943,45
32	1674,08	2347,57	673,49	5891,24	7786,86	1895,62
33	4689,92	8796,17	4106,25	-342,75	953,39	1296,14
34	495,55	722,71	227,15	2623,68	3587,29	963,61
35	1963,76	2830,76	867,00	-166,89	1088,65	1255,55
36	552,37	796,71	244,34	355,46	1499,70	1144,24
37	746,82	1043,65	296,84	7940,82	10715,35	2774,53
38	6872,56	14891,39	8018,83	-1231,96	286,94	1518,89
39	279,30	411,05	131,75	5371,47	7062,10	1690,63
40	4187,83	7540,05	3352,21	182,30	1361,72	1179,43
41	678,23	957,01	278,78	698,43	1778,98	1080,55
42	894,04	1230,98	336,93	2167,07	3110,51	943,45

Таблиця 2.6 – Границі інтервалу прогнозування рівнянь регресії (фрагмент даних)

№ про-екта	Нелінійне рівняння регресії			Лінійне рівняння регресії		
	нижня границя	верхня границя	довжина	нижня границя	верхня границя	довжина
1	2	3	4	5	6	7
31	504,59	3554,19	3049,60	-454,09	5731,67	6185,75
32	738,02	5971,61	5233,59	3638,78	10039,31	6400,53
33	2060,55	20799,53	18738,98	-2819,32	3429,95	6249,28
34	305,22	1537,92	1232,69	11,05	6199,92	6188,86
35	853,24	7190,02	6336,79	-2659,61	3581,37	6240,98
36	325,81	1745,23	1419,41	-2182,20	4037,35	6219,55
37	395,25	2444,60	2049,35	5971,32	12684,85	6713,53
38	3113,69	33642,04	30528,35	-3622,13	2677,11	6299,25
39	223,11	701,76	478,65	3045,36	9388,21	6342,85
40	1826,68	18058,27	16231,59	-2341,05	3885,07	6226,12
41	370,86	2198,68	1827,82	-1865,37	4342,78	6208,14
42	447,61	2974,36	2526,75	-454,09	5731,67	6185,75

Як видно із табл. 2.6, нижня границя інтервалу прогнозування нелінійного рівняння регресії, яке побудовано з використанням нормалізуючого перетворення сім'ї S_B Джонсона, більша нуля, тоді як нижня границя інтервалу прогнозування лінійного рівняння регресії має значення, менші нуля, для більшості наборів даних. Довжина інтервалу прогнозування нелінійного рівняння регресії менша за відповідну довжину лінійного рівняння регресії для більшості наборів даних.

3 ПРОЕКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ВЕБ-ЗАСТОСУНКІВ, РЕАЛІЗОВАНИХ МОВОЮ JAVA

3.1 Ескізний проект програмного забезпечення

3.1.1 Вибір мови моделювання

UML – 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.

Тому для відображення процесу створення ПЗ та для того, щоб простежити послідовність трансформації від початкової моделі до завершеної програмної системи було обрано мову моделювання UML [34].

Мова UML призначена для вирішення наступних завдань:

– надати користувачеві мову візуального моделювання, яка легко сприймається, спеціально призначену 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 об'єктно-орієнтованому аналізі і проектуванні конкретної предметної області;

– заохочувати розвиток ринку об'єктних інструментальних засобів.

Жодна з конструкцій мови UML не повинна залежати від особливостей її реалізації в основних мовах програмування та інтегрувати в собі новітні і найкращі досягнення практики та здатність удосконалюватися в часі.

3.1.2 Модель варіантів використання

Для наглядного відображення ПЗ для оцінювання розміру веб-застосунків, реалізованих мовою Java, яке розробляється, було побудовано одну з UML-діаграм – діаграму варіантів використання. Було виявлено варіанти використання та побудовано їх сценарії, розроблено зовнішню специфікацію ПЗ для оцінювання розміру веб-застосунків, реалізованих мовою Java та стани цієї програми при виконанні [35].

Дане ПЗ передбачає наявність лише одного актора – користувача ПЗ, який буде повноцінно використовувати всі функції ПЗ. Діаграма варіантів використання ПЗ для оцінювання розміру веб-застосунків, реалізованих мовою Java, представлена на рис. 3.1.

Рисунок 3.1 – Діаграма варіантів використання ПЗ для оцінювання розміру веб-застосунків, реалізованих мовою Java

Основні функції користувача ПЗ можна бачити на представленій діаграмі варіантів використання.

3.1.3 Специфікації варіантів використання

Виявлені варіанти використання для ПЗ, що розробляється, представлені в табл. 3.1.

Таблиця 3.1 – Виявлені варіанти використання

Основні актори	Найменування	Формулювання
Користувач	Занесення даних	Занесення даних про фактичний розмір веб-застосунків, реалізованих мовою Java, загальну кількість класів та кількість рядків коду
Користувач	Розрахунок параметрів для вихідних вибірок	Розраховує параметри вибірок для занесених даних
Користувач	Перевірка вихідних вибірок на нормальність	Перевіряє вихідні вибірки на відповідність гаусівському розподілу
Користувач	Нормалізація вихідних вибірок	Нормалізує вихідні вибірки за допомогою нормалізую чого перетворення Джонсона
Користувач	Розрахунок параметрів для нормалізованих вибірок	Розраховує параметри вибірок для нормалізованих даних
Користувач	Перевірка нормалізованих вибірок на нормальність	Перевіряє нормалізовані вибірки на відповідність гаусівському розподілу
Користувач	Побудова лінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування	Будує лінійне рівняння регресії для оцінювання розміру веб-застосунків, реалізованих мовою Java, а також довірчий інтервал та інтервал прогнозування лінійного рівняння регресії
Користувач	Побудова нелінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування	Будує нелінійне рівняння регресії для оцінювання розміру веб-застосунків, реалізованих мовою Java, а також довірчий інтервал та інтервал прогнозування нелінійного рівняння регресії

3.1.4 Сценарії варіантів використання

Для відображення поведінки об’єктів було використано діаграму діяльності. Діаграми діяльності були побудовані для всіх основних варіантів використання, наведених на рис. 3.1 та в табл. 3.1. Декілька прикладів побудованих діаграм діяльності для варіантів використання «Завантаження файлу» та «Нормалізація вихідних вибірок» представлені на рис. 3.2 – 3.3 [36].

Рисунок 3.2 – Діаграма діяльності для варіанту використання «Завантаження файлу»

Рисунок 3.3 – Діаграма діяльності для варіанту використання «Нормалізація вихідних вибірок»

Наведені діаграми діяльності для варіантів використання добре показують послідовність подій , що ініціюють дії або є їх кінцевим результатом.

3.1.5 Прототип інтерфейсу користувача

Інтерфейс користувача – це сукупність програмно-апаратних засобів, що забезпечують взаємодію користувача з комп’ютером та ПЗ. Основу такої взаємодії становлять діалогові вікна, у цьому випадку розуміють регламентований обмін інформацією між людиною та ПЗ, здійснюваний в режимі реального часу та спрямований на рішення конкретної задачі – обмін інформацією та координацію дій. Кожний діалог складається з окремих

процесів введення – виведення, які фізично забезпечують зв’язок користувача і ПЗ [37].

В рамках ескізного проекту ПЗ було розроблено прототип інтерфейсу користувача. На рис. 3.4 зображено прототип початкового вікна ПЗ.

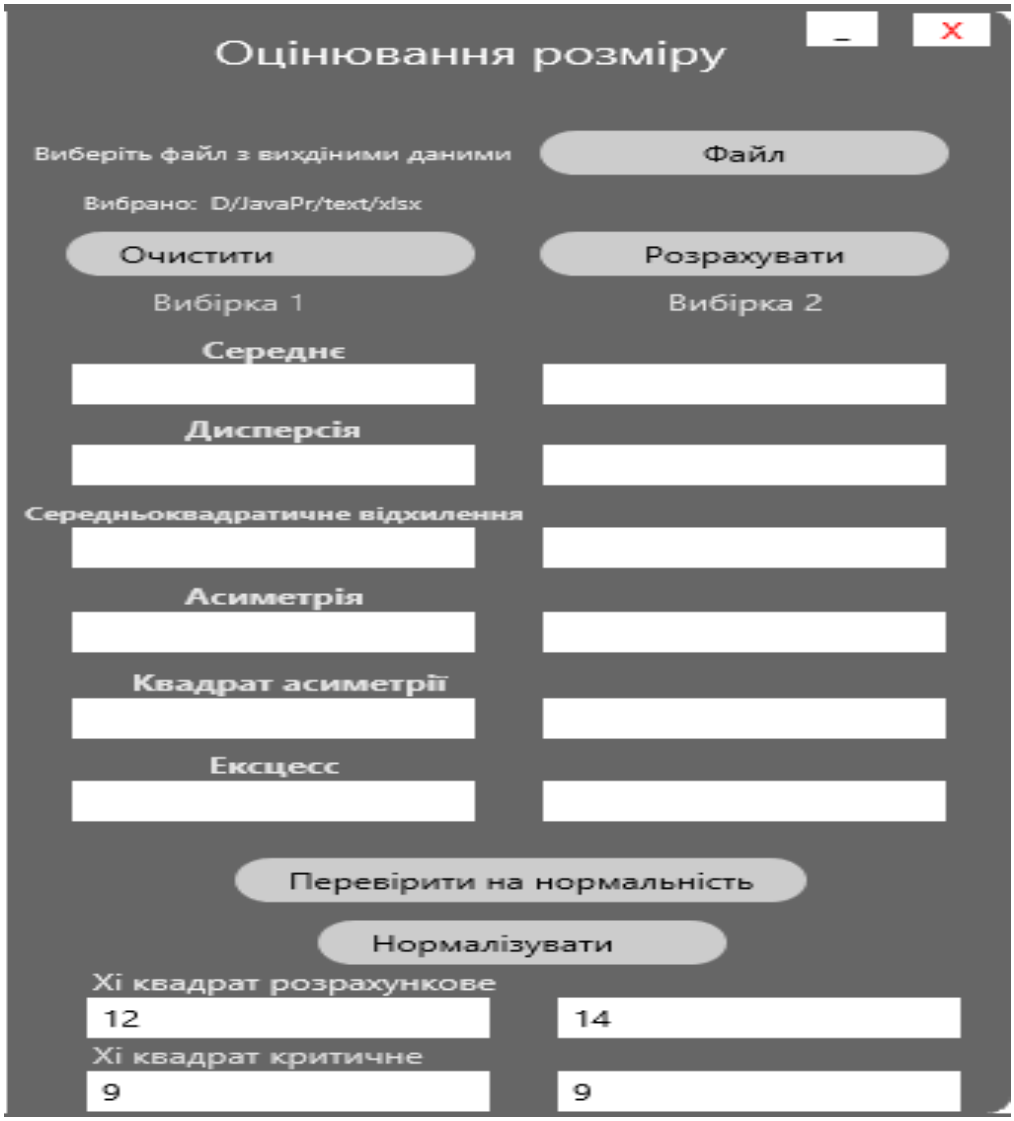

Рисунок 3.4 – Прототип початкового вікна ПЗ

Після розрахунку параметрів вибірок для занесених даних з`являється можливість перевірки на нормальність розподілу та перейти далі до нормалізації вихідних вибірок. На рис. 3.5 зображено прототип вікна «Нормалізація».

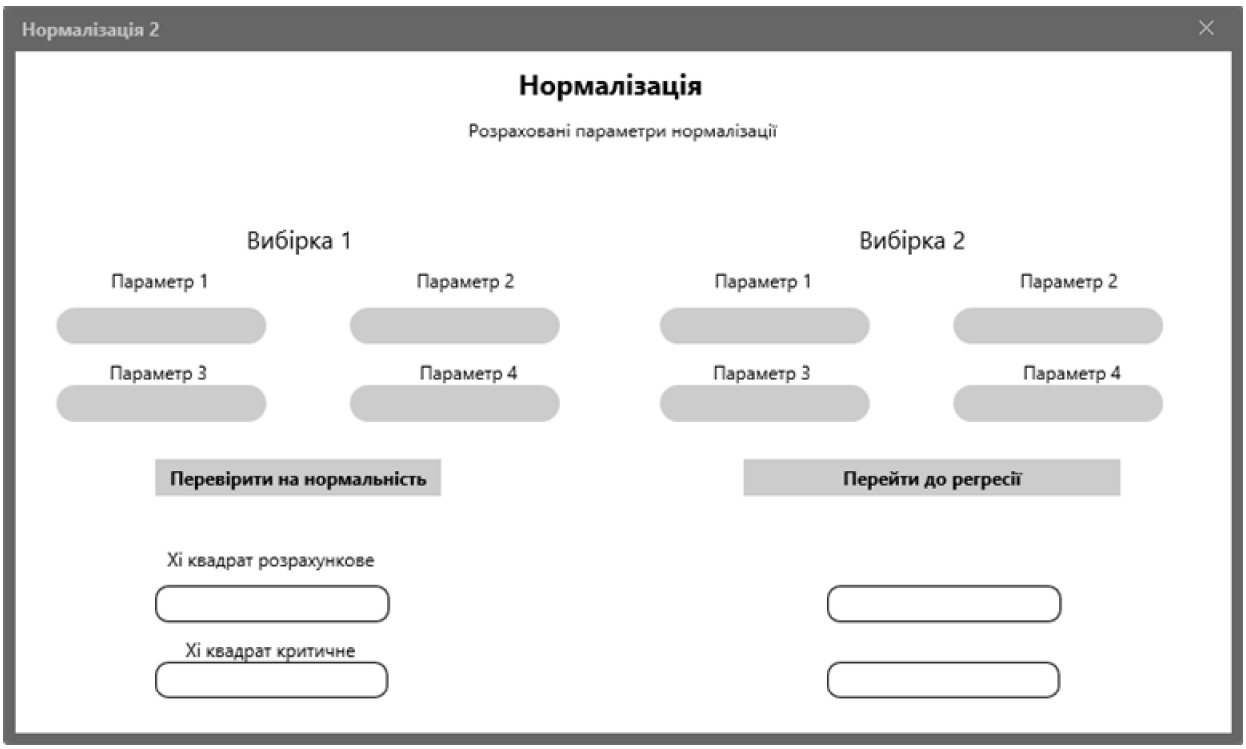

Рисунок 3.5 – Прототип вікна «Нормалізація»

Пісня нормалізації з`являється можливість перейти до побудови рівнянь регресії, відповідне вікно має дві вкладками: для лінійного та нелінійного рівняння регресії. На рис. 3.6 зображено прототип вікна «Рівняння регресії», вкладка «Лінійне рівняння».

Рисунок 3.6 – Прототип вікна «Рівняння регресії», вкладка «Лінійне рівняння»

3.2 Технічний проект програмного забезпечення

3.2.1 Статична модель програмного забезпечення

Діаграма класів служить для представлення статичної структури моделі ПЗ в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти і кооперації, а також їхні відносини. Діаграма класів також може містити позначення для пакетів або для вкладених пакетів, деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм [38].

Діаграма класів ПЗ «JavaWebAnalytics» зображена на рисунку3.7.

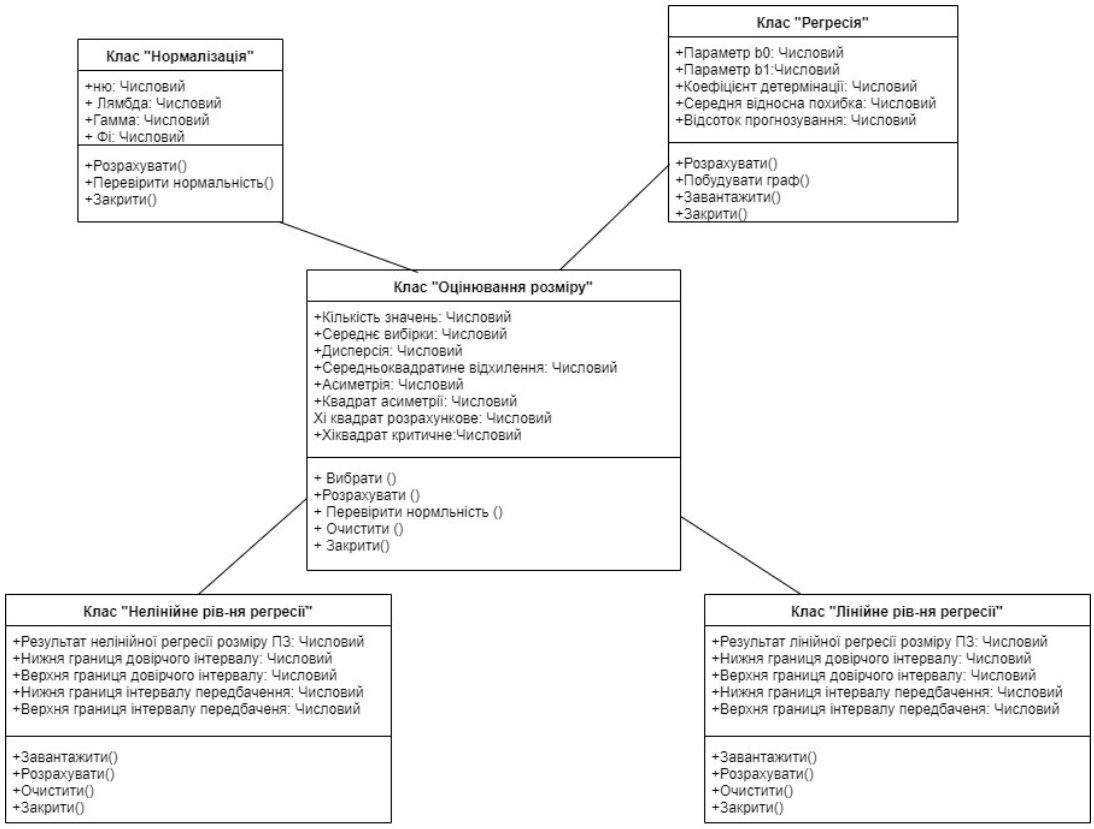

Рисунок 3.7 – Діаграма класів програмного забезпечення «JavaWebAnalytics»

На цій діаграмі класів відображені такі елементи, як класи розроблюваного ПЗ, вказано типи та їх атрибути, а також методи.

3.2.2 Специфікації класів програмного забезпечення

Специфікація класів ПЗ – це визначення властивостей, які характеризують стан об'єкта, і методів, які є способом реалізації його поведінки. Специфікація класів ПЗ «JavaWebAnalytics» представлена в табл. 3.2.

Таблиця 3.2 – Специфікація класів ПЗ «JavaWebAnalytics»

Опис	Складова
1	2
Оцінювання розміру	
Відповідальність	Даний клас відповідає за оцінювання розміру вибірок.
Атрибут	Кількість значень, середнє вибірки, дисперсія, середньоквадратичне відхилення, асиметрія, квадрат асиметрії, хі квадрат розрахункове, хі квадрат критичне
Операції зі специфікаціями нетривіальних операцій	Вибрати(): вибрати файл Розрахувати(): розрахунок параметрів вибірки Перевірити нормальність(): перевірка на нормальність розподілу Закрити(): закрити Очистити(): очистити дані
Нормалізація	
Відповідальність	Даний клас відповідає за нормалізацію вихідної вибірки.
Атрибут	ню, гамма, фі, лямбда.
Операції зі специфікаціями нетривіальних операцій	Розрахувати(): розрахунок параметрів Перевірити нормальність(): перевірка на нормальність розподілу. Закрити(): закрити.
Регресія	
Відповідальність	Даний клас відповідає за лінійне та нелінійне рівняння регресії.
Атрибут	Параметр b_0, параметр b_1, коефіцієнт детермінації, середня відносна похибка, відсоток прогнозування.

Продовження табл. 3.2

1	2
Операції зі специфікаціями нетривіальних операцій	Розрахувати(): розрахунок параметрів. Побудувати графік(): побудова графіку рівняння регресії, довірчого інтервалу та інтервалу прогнозування. Завантажити(): завантажити вихідний файл.
Лінійне рівняння регресії	
Відповідальність	Даний клас відповідає за інформацію про параметри лінійного рівняння
Атрибут	Результат лінійної регресії,нижня границя довірчого інтервалу, верхня границя довірчого інтервалу, нижня границя інтервалу передбачення, верхня границя інтервалу передбаченя
Операції зі специфікаціями нетривіальних операцій	Завантажити, розрахувати, очистити, закрити
Нелінійне рівняння регресії	
Відповідальність	Даний клас відповідає за інформацію про параметри нелінійного рівняння
Атрибут	Результат нелінійної регресії,нижня границя довірчого інтервалу, верхня границя довірчого інтервалу, нижня границя інтервалу передбачення, верхня границя інтервалу передбаченя
Операції зі специфікаціями нетривіальних операцій	Завантажити, розрахувати, очистити, закрити
Операції зі специфікаціями нетривіальних операцій	Розрахувати(): розрахунок параметрів Перевірити нормальність(): перевірка на нормальність розподілу. Закрити(): закрити.

3.2.3 Динамічна модель програмного забезпечення

Динамічна модель ПЗ може бути представлена у вигляді діаграми діяльності, діаграми послідовності, діаграми взаємодії або діаграми станів.

Динамічна модель ПЗ, що розроблюється, представлена у вигляді діаграми послідовності. Діаграма послідовності – це діаграма на якій показані взаємодії об’єктів, упорядковані за часом їхнього прояву [39].

У технічному проекті ПЗ представлені діаграми послідовності для деяких варіантів використання: на рис. 3.8 представлено діаграму послідовності для варіанту використання «Завантаження файлу», на рис. 3.9 представлено діаграму послідовності для варіанту використання «Нормалізація вихідних вибірок».

Рисунок 3.8 – Діаграма послідовності для варіанту використання «Завантаження файлу»

У варіанті використання «Завантаження файлу» користувач спочатку вибирає файл. Система перевіряє розширення файлу на допустимість та повертає результат.

Рисунок 3.9 – Діаграма послідовності для варіанту використання «Нормалізація вихідних вибірок»

У варіанті використання «Нормалізація вихідних вибірок» користувач спочатку проходить етап розрахунку параметрів вихідних вибірок. Потім вводить параметри початкового наближення для нормалізації. Система перевіряє наявність та допустимість введених параметрів. Якщо дані коректні, то повертає розраховані оптимальні параметри перетворення Джонсона.

3.3 Робочий проект програмного забезпечення

3.3.1 Обгрунтування вибору мови програмування та системи розробки

Для реалізації ПЗ для оцінювання розміру веб-застосунків, реалізованих мовою Java, необхідно розглянути проектні рішення з вибору мови програмування. Тобто це повинно бути ПЗ, яке буде надавати можливість вводити вхідні дані, а також в якому будуть реалізовані алгоритми обробки

даних за допомогою обраних методів. Тому для розробки ПЗ необхідно визначитись з мовою програмування, якою можливо реалізувати це ПЗ.

Для вирішення цього завдання можна використати будь-які мови програмування високого рівня, але оскільки ПЗ виконує складні математичні обчислення, бажано щоб мова програмування та середовище розробки дозволяли легко працювати з ними. Тому було обрано мову програмування Java та середовище розробки Eclipse .

Java – високорівнева мова програмування, розроблена компанією Sun Microsystems і випущена в 1995 році. Працює на різних платформах, таких як Windows, Mac OS, а також різних версій UNIX. До переваг даної мови програмування можна віднести такі фактори [40]:

1) Об'єктно-орієнтована: в Java все є об'єктом. Доповнення може бути легко розширено, тому що воно заснований на об'єктній моделі.

2) Платформонезалежна: на відміну від багатьох інших мов, включаючи C і C++, Java, коли була створена, не компілювалася в платформі конкретної машини, а в незалежний від платформи байт-коді. Цей байт код інтерпретується в Java Virtual Machine (JVM), на якій він в даний час працює.

3) Проста: процеси вивчення та введення в мову програмування Java залишаються простими.

4) Безпечна: методи перевірки автентичності засновані на шифруванні з відкритим ключем.

5) Архітектурно-нейтральна: компілятор генерує архітектурно-нейтральні об'єкти формату файлу, що робить скомпільований код виконуваним на багатьох процесорах, з наявністю системі Java Runtime.

6) Портативна: архітектурно-нейтральний і не має залежності від реалізації аспектів специфікацій – все це робить Java портативним. Компілятор в Java написаний на ANSI C з чистою переносимістю, яка є підмножиною POSIX.

7) Міцна: докладає зусиль, щоб усунути помилки в різних ситуаціях, спираючись в основному на час компіляції, перевірку помилок і перевірку під час виконання.

8) Багатопотокова: є функції багатопоточності, можна писати програми, які можуть виконувати безліч завдань одночасно. Введення в мову Java цієї конструктивної особливості дозволяє розробникам створювати налагоджені інтерактивні додатки.

9) Інтерпретована: Java байт-код переводиться на льоту в машинні інструкції та ніде не зберігається. Роблячи процес більш швидким і аналітичним, оскільки зв'язування відбувається як додаткове з невеликою вагою процесу.

10) Високопродуктивна: введення Just-In-Time компілятора, дозволило отримати високу продуктивність.

11) Поширена: призначена для розподіленого середовища інтернет.

12) Динамічна: програмування на Java вважається більш динамічним, ніж на C або C++, тому що призначено для адаптації до мінливих умов [40].

Eclipse є безкоштовною програмною платформою з відкритим вихідним кодом, контролюється організацією Eclipse Foundation. Написана на мові програмування Java і основною метою її створення є підвищення продуктивності процесу розробки ПЗ. Претендує на статус найбільш популярного Java IDE і є єдиним конкурентом такої потужної платформи як NetBeans.

Але на відміну від NetBeans, який для створення елементів призначеного для користувача інтерфейсу використовує від платформи незалежну бібліотеку Swing, в Eclipse використовується платформо-залежна бібліотека SWT – Standard Widget Toolkit.

IDE, розроблені на базі платформи Eclipse, застосовуються для створення ПЗ на різних мовах програмування, тому що Eclipse є платформою для розробки будь-яких інтегрованих середовищ програмування і розширень для себе ж, за принципом "Додатки для Eclipse розробляються в самій Eclipse.

3.3.2 Реалізація та тестування основних класів програмного забезпечення

Реалізація ПЗ була виконана на високорівневій мові програмування Java.

Для тестування ПЗ розглядалися такі основні групи методів:

– методи тестування по формальним специфікаціям (методи чорного ящика);

– методи структурного тестування (методи білого ящика).

При тестуванні методами чорного ящика розглядаються системні характеристики ПЗ, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе. Наприклад, якщо в ПЗ 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^{10}. Тестування методами чорного ящика не реагує на багато особливостей програмних помилок.

Тестування методами чорного ящика дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до ПЗ. Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів.

Тестування методами чорного ящика забезпечує пошук наступних категорій помилок:

– некоректних чи відсутніх функцій;

– помилок інтерфейсу;

– помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних;

– помилок характеристик (необхідна ємність пам'яті і т.д.);

– помилок ініціалізації та завершення.

Подібні категорії помилок методами білого ящика не виявляються.

Оскільки методи білого ящика вимагають більших затрат ресурсів, то в даній роботі виконувалося тестування по специфікаціям, а саме розбиття на класи еквівалентності.

ПЗ розглядалось як чорний ящик. Випробування зводились до послідовного вводу тестових наборів даних та аналізу отриманих результатів. У розроблюваному ПЗ користувач вводить дані вручну лише на одному етапі – етапі розрахунку параметрів Джонсона. Він повинен ввести чотири параметри для початкового наближення. В табл. 3.3 представлені класи еквівалентності та граничні значення таких вхідних даних: гамма, лямбда, фі, ню.

Таблиця 3.3 – Класи еквівалентності вхідних даних для класу «Нормалізація»

Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
гамма	числовий	Всі окрім числового типу (логічні, символьні, графічні зображення)
лямбда	числовий	Всі окрім числового типу (логічні, символьні, графічні зображення)
фі	числовий	Всі окрім числового типу (логічні, символьні, графічні зображення)
ню	числовий	Всі окрім числового типу (логічні, символьні, графічні зображення)

Почнемо тестування з правильних класів еквівалентності класу «Нормалізація», представлених в табл. 3.4.

Таблиця 3.4 – Тести з правильних класів еквівалентності для класу «Нормалізація»

№	Вхідні умови	Правильні класи еквівалентності	Вихідні дані
1	Гамма	В поле «Гамма» введено число «84156.04»	Результат вірний
2	Лябда	В поле «Лямбда» введено число «999»	Результат вірний
3	фі	В поле «Фі» введено число «5,12»	Результат вірний
4	ню	В поле «Ню» введено число «0,02532»	Результат вірний

Тести з неправильних класів еквівалентності класу «Нормалізація» представлені в табл. 3.5.

Таблиця 3.5 – Тести з неправильних класів еквівалентності класу «Нормалізація»

№	Вхідні умови	Неправильні класи еквівалентності	Вихідні дані
1	Гамма	В поле «Гамма» введено графічний елемент – смайл (скопійовано з інтернету)	Після введення смайла виводиться повідомлення про помилку. Програма далі працює правильно. Результат підтверджено.
2	Лябда	В поле «Лямбда» введено будь-який символ ASCII	Програма виводить повідомлення про помилку. Результат підтверджено.
3	фі	В поле «Фі» не вводимо нічого	Після введення цифр виводиться повідомлення про помилку. Програма далі працює правильно. Результат підтверджено.
4	ню	В поле «Ню» введено будь-який символ ASCII	Програма виводить повідомлення про помилку. Результат підтверджено.

Отже, за результатами тестування правильниз та неправильних класів еквівалентності класу «Нормалізація», ПЗ працює правильно, не допускаючи до введення будь-яких інших елементів, окрім стандартних символів вводу. Тестування інших класів виконується аналогічно.

3.3.3 Випробування програмного забезпечення

Випробування ПЗ було виконано згідно з програмою та методикою випробувань ПЗ, яка представлена у додатку Д.

Функція «Завантаження файлу з вихідними даними».

На головному вікні натиснули кнопку «Вибрати файл». Поруч було наведено допустимі параметри. Було вибрано картинку з недопустимим розширенням .png. У результаті отримали повідомлення системи про недопустимий формат файлу та можливість вибрати інший файл (рис. 3.10).

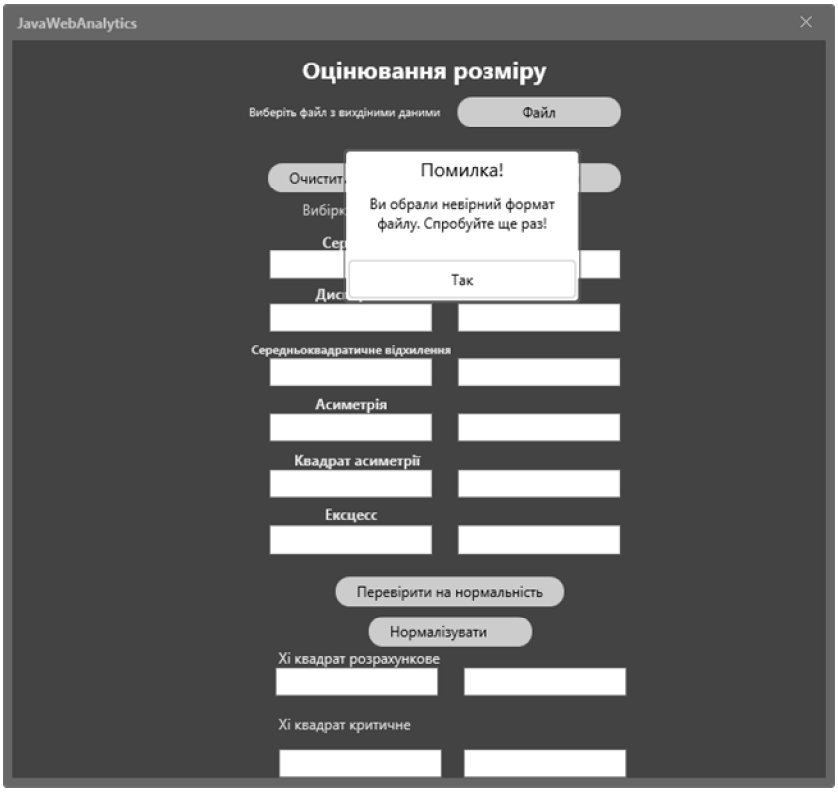

Рисунок 3.10 – Головне вікно ПЗ з повідомленням,
що обрано некоректний формат файлу

Випробування на завантаження коректного формату файлу.

На головному вікні програми натиснули кнопку «Файл». Було обрано файл з розширенням .xlsx. У результаті файл успішно завантажився та нижче було виведено його назву. Випробування пройдено успішно. Результат зображено на рис. 3.11.

Випробування на розрахунок параметрів без завантаження файлу.

На головному вікні програми не вибрали жодного файлу. Натиснули кнопку «Розрахувати». У результаті отримали повідомлення програми про необхідність вибрати файл. Випробування пройдено успішно. Результат зображено на рисунку 3.12.

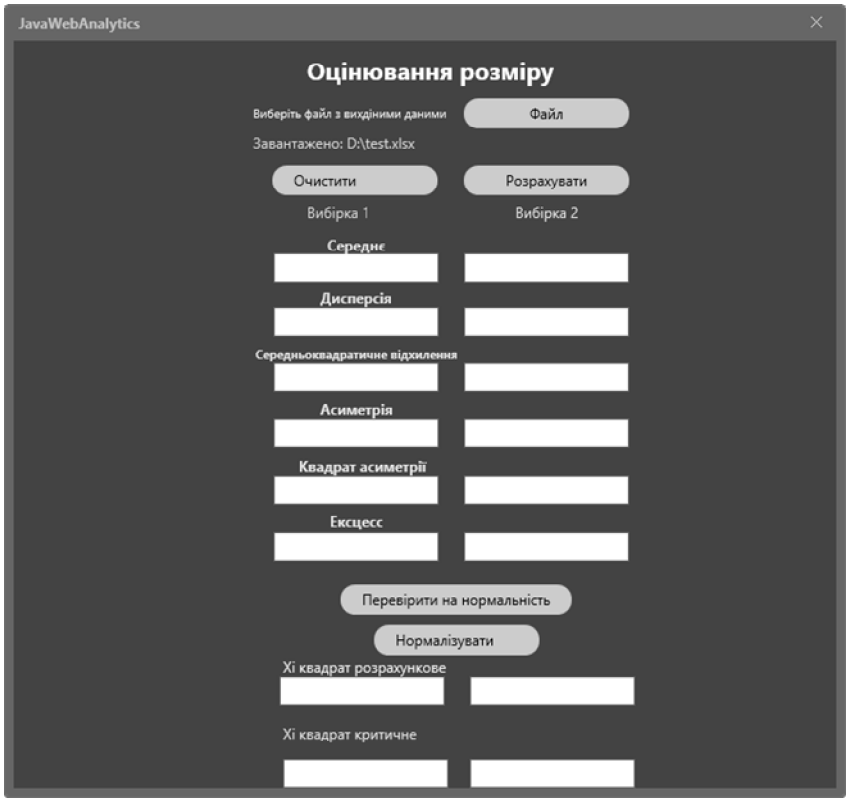

Рисунок 3.11 – Результат завантаження коректного формату файлу

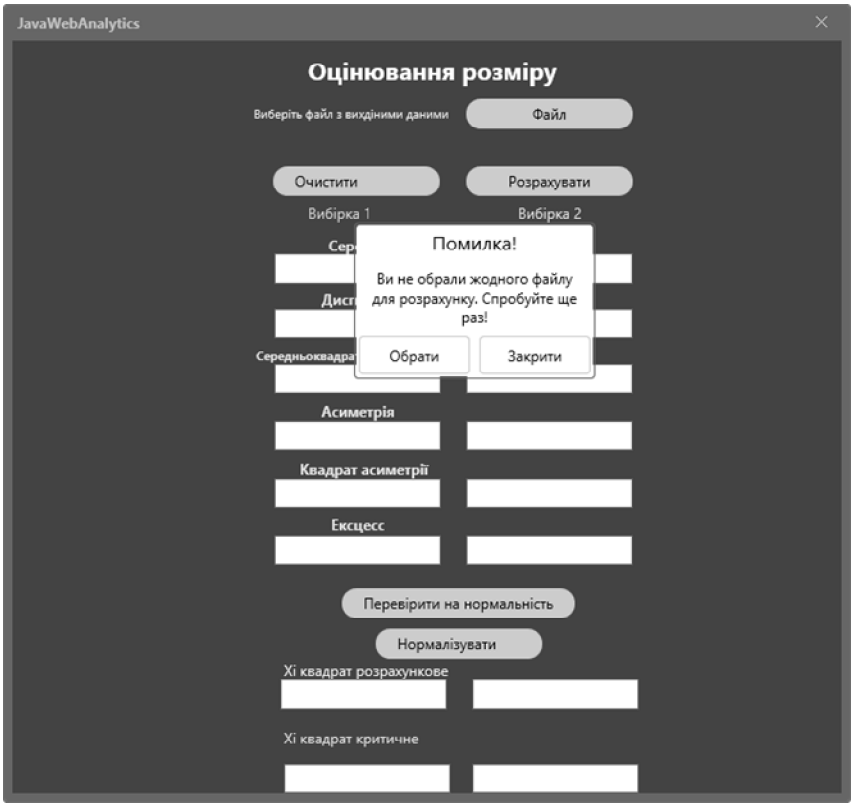

Рисунок 3.12 – Результат тестування розрахунку параметрів без завантаження файлу

Випробування на розрахунок параметрів після завантаження файлу.

Після завантаження файлу натиснули кнопку «Розрахувати». У відповідних текстових блоках з’явились розраховані параметри. Випробування пройдено успішно. Результат зображено на рис. 3.13.

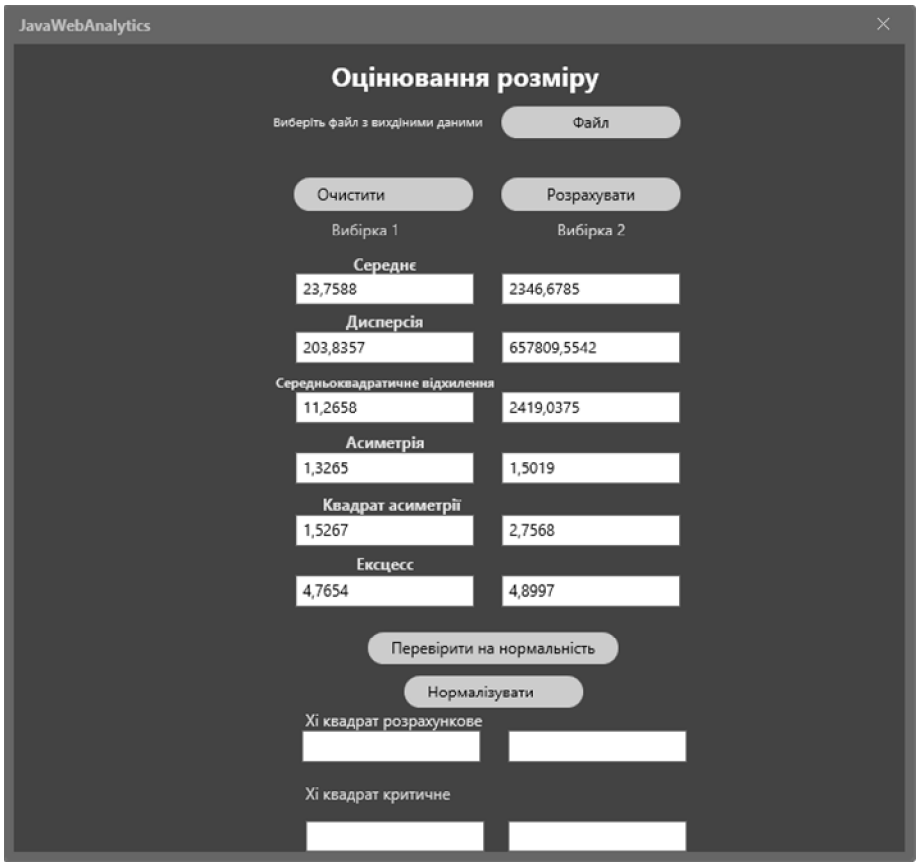

Рисунок 3.13– Результат тестування розрахунку параметрів після коректного завантаження файлу

Випробування на розрахунок параметрів Джонсона при некоректному вводі.

У вікні «Нормалізація» ввели некоректні параметри для початкового наближення та натиснули кнопку «Розрахувати». У результаті з’явилось відповідне системне повідомлення про необхідність ввести коректні параметри для розрахунку.

Випробування на розрахунок параметрів Джонсона при коректному вводі.

У вікні «Нормалізація» ввели коректні параметри для початкового

наближення та натиснули кнопку «Розрахувати параметри Джонсона». У результаті з’явилося відповідне вікно, в якому виведено розраховані параметри нормалізації та можливість перевірити нормальність розподілу нормалізованих вибірок. Результат тестування наведено на рисунку 3.14.

Рисунок 3.14 – Результат тестування розрахунку параметрів Джонсона

4 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ВЕБ-ЗАСТОСУНКІВ, РЕАЛІЗОВАНИХ МОВОЮ JAVA

ПЗ для оцінювання розміру веб-застосунків, реалізованих мовою Java, розроблене в рамках кваліфікаційної роботи, засноване на удосконаленому однофакторному рівнянні регресії для оцінювання розміру веб-застосунків, реалізованих мовою Java, основні формули та розрахунки якого наведені у розділі 2. Удосконалення однофакторного нелінійного рівняння регресії та методика розрахунку довірчого інтервалу і інтервалу прогнозування нелінійного рівняння регресії дозволило підвищити достовірність оцінювання розміру розміру веб-застосунків, реалізованих мовою Java.

Дане ПЗ може використовуватись як великими компаніями з розробки ПЗ, так і просто окремими програмістами для оцінки власних проектів. Воно не потребує особливих вмінь та навичок, розраховане на типового користувача з базовими знаннями комп’ютера.

Усі нижчезазначені функції були реалізовані в повному обсязі у розробленому ПЗ та випробувані під час етапу тестування:

– можливість завантажити файл з вихідними даними у форматах Excel (.xml, .xls, .xlsx,);

– можливість змінити файл з вихідними даними;

– зчитування даних з файлу;

– розрахунок параметрів для вихідних вибірок;

– розрахунок параметрів перетворення Джонсона;

– нормалізація вихідних вибірок за допомогою параметрів Джонсона;

– розрахунок параметрів для нормалізованих вибірок;

– перевірка вихідних та нормалізованих вибірок на нормальність розподілу;

– побудова лінійного рівняння регресії, довірчого інтервалу та інтервалу

прогнозування для нього;

– побудова нелінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування для нього;

– оцінювання розміру веб-застосунків, реалізованих мовою Java.

Було реалізовано декілька перевірок на коректність введених користувачем даних, а саме:

– перевірка на коректність формату файлу під час завантаження;

– перевірка на наявність файлу завантаження;

– перевірка на наявність введених початкових параметрів на етапі нормалізації;

– перевірка на коректність введених початкових параметрів на етапі нормалізації;

– перевірка на послідовність дій користувача.

Інтерфейс був розроблений максимально простим для зручності користувача, кольорова гамма приємна та спокійна для очей. Шрифт достатньо великий.

Також було реалізовано функцію завантаження файлу з розрахованими параметрами у форматі .txt.

Усі вимоги, наведені у технічному завданні (див. Додаток А) були виконані. В якості технічних ресурсів був використаний ПК з комплектацією, наведеною у технічному завданні.

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).

5 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Найбільш важливим моментом при проектуванні ПЗ для розробника з економічної точки зору, є процес формування його вартості. Вочевидь, що комп'ютерна програма являє собою дуже специфічний товар з безліччю особливостей.

Створення ПЗ вимагає одноразових витрат на його розробку, придбання необхідних технічних засобів і поточних витрат на функціонування продукту. Економія від функціонування програмного продукту визначається з урахуванням витрат на його експлуатацію . Відношення цієї економії до витрат на створення програмного продукту характеризує економічну ефективність капітальних вкладень. Економічні показники визначаються по діючим на момент розрахунку оптовим цінам, тарифам і ставкам заробітної плати.

5.1 Розрахунок витрат на створення й експлуатацію програмного забезпечення

Витрати на розробку ПЗ складаються з витрат на зарплату розробника, на амортизацію комп'ютера, на якому виконується розробка, на експлуатацію цього комп'ютера, на засоби розробки та витрат на матеріали і комплектуючі.

Розробка ПЗ виконується програмістом, місячний оклад якого складає 9000 грн. Додаткова заробітна плата складає 20% від основної . Виходячи з цього, основна і додаткова заробітна плата програміста 10800 грн/міс , а вартість сучасного ПК складає 7000 грн (приведена вартість комп'ютера на базі процесора Intel Celeron Dual-core J1800 (2,41 ГГц) / RAM 4 ГБ / HDD 500 ГБ / Intel HD Graphics / LAN / Windows 8 SL та монітора 18,5'' Asus VS190). При

вартості кіловат-години електроенергії у 1,68 грн розраховується вартість розробки ПЗ. Витрати на допоміжні матеріали приведені в таблиці 5.1

Таблиця 5.1 – Витрати на допоміжні матеріали

Пункти витрат	Сума
Заправлення картриджа до принтера	100,00
Папір	100,00
CD-диск	7,00
Непередбачені витрати	160,00
Разом матеріали і комплектуючі	367,00

Вартість ПЗ розрахуємо по формулі:

$$Спр (Ззп + Зсз + Ззг + Зе) * Т + Зм \quad (5.1)$$

де Т – тривалість розробки, міс; Ззп – основна і додаткова заробітна плата обслуговуючого персоналу, грн.; Зсз – відстрахування на соціальні заходи або єдиний соціальний внесок (36,76% від основної і додаткової заробітної плати), грн; Ззг – загальногосподарські витрати (10% від основної заробітної плати) грн.; Зе – витрати на електроенергію; Зм – витрати на основні і допоміжні матеріали. Зе при поживанні потужності 0,5 кВт, тривалості роботи за місяць рівної 22*8=176 годин, вартості кіловат-години електроенергії 1,68 грн складає

$$Зе = 176 * 1,68 * 0,5 = 147,84 грн.$$

Всі витрати на розробку ПЗ можна побачити в таблиці 5.2

Таблиця 5.2 – витрати на розробку програмного забезпечення

Найменування витрат	Одиницю	Кількість
Тривалість розробки	Міс	1
Основна і додаткова ЗП	Грн.	10800,00
Відрахування на соціальні заходи	Грн.	3963,60
Загальногосподарські витрати	Грн.	900,00
Витрати на матеріали	Грн.	367,00
Витрати на електроенергію	Грн.	147,84

Відповідно до формули (5.1) вартість програмного забезпечення:

Спр = (10800,00+3963,60+900+147,84)*1+367,00=16178,44 грн.

Амортизаційні відрахування на успадкування складають 60% балансованої вартості на рік:

Аоб=7000,00*0,6=4200,00 грн.

У масштабах підприємства річні витрати на основні і допоміжні матеріали (гнучкі диски, папір) визначають в розмірі 5% вартості основного успадкування:

Вм = 7000,00 * 0,05 = 350,00 грн.

Річний обсяг робіт ПК у годинах визначається в такий спосіб:

$$Фм = 264,5 * Тз, \qquad (5.2)$$

де тз – це середнє місячне завантаження успадкування (близько 4 годин); 264,5 – середня кількість робочих днів у році.

Отже, річний обяг роботи ПК скаладє:

Фм = 264,5 * 4 = 1058 годин.

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

Зе = 1058 * 0,5 * 1,68 = 888,72 грн.

Експлуатаційні витрати для ПК за рік складуть:

Ззр = 4200,00+350,00+888,72 = 5438,72 грн.

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

Зсе = 7000,00+16178,44+5438,72=28617,16 грн.

5.2 Економічна ефективність розробки й впровадження програмного забезпечення

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

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

– високі витрати на складування паперового 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 програмного забезпечення, відносяться:

– підвищення продуктивності праці;

– вивільнення робочого часу.

Крім того, не піддається прямій грошовій оцінці підвищення оперативності керування, якості одержуваних результатів, поліпшення організації праці і т.д. Обов’язковою умовою визначення економічної ефективності програмного забезпечення є порівнянність усіх показників у часі, за цінами й іншими нормами, використовування для визначення показників, за змістом і колом елементів витрат.

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

Зарплата 0,6 працівника в рік складає:

$$(9000{,}00*12)*0{,}6=64800{,}00 \text{ грн.}$$

Річний економічний ефект розраховується по формулі:

$$Э_{год}=\Delta C_n-E_n \cdot k,$$

де ΔC_n – вивільнені кошти після впровадження програмного забезпечення (64800,00 грн.) мінус експлуатаційні витрати (5438,72 грн.); E_n – коефіцієнт ефективності (дорівнює коефіцієнту амортизації – 0,6); k – одноразові витрати на впровадження програмного забезпечення (8423,86 грн.).

Річний економічний ефект буде

$$Э_{год}=(64800–5438,72)*0,6=35616,77 \text{ грн.}$$

Строк окупності програмного забезпечення розраховується по формулі:

$$T= k/\Delta C_n=28617,16/59361.28=0,48 \text{ (року)}$$

Отже строк окупності програмного забезпечення складає приблизно 6 місяців.

6 ОХОРОНА ПРАЦІ

Охорона праці – це система правових, соціально-економічних, організаційно-технічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження здоров'я і працездатності людини в процесі праці [41].

Державна політика в галузі охорони праці базується на принципах:

1) пріоритету життя і здоров'я працівників, повної відповідальності роботодавця за створення належних, безпечних і здорових умов праці;

2) соціального захисту працівників, повного відшкодування шкоди особам, які потерпіли від нещасних випадків на виробництві та професійних захворювань;

3) встановлення єдиних вимог з охорони праці для всіх підприємств та суб'єктів підприємницької діяльності незалежно від форм власності та видів діяльності;

4) адаптації трудових процесів до можливостей працівника з урахуванням його здоров'я та психологічного стану;

5) використання економічних методів управління охороною праці, участі держави у фінансуванні заходів щодо охорони праці, залучення добровільних внесків та інших надходжень на ці цілі, отримання яких не суперечить законодавству;

6) підвищення рівня промислової безпеки шляхом забезпечення суцільного технічного контролю за станом виробництв, технологій та продукції, а також сприяння підприємствам у створенні безпечних та нешкідливих умов праці;

7) комплексного розв'язання завдань охорони праці на основі загальнодержавної, галузевих, регіональних програм з цього питання та з урахуванням інших напрямів економічної і соціальної політики, досягнень в галузі науки і техніки та охорони довкілля;

8) використання світового досвіду організації роботи щодо поліпшення умов і підвищення безпеки праці на основі міжнародного співробітництва.

9) соціальний захист працівників , повне відшкодування шкоди особам, потерпілим від нещасних випадків на виробництва та професійних захворювань ;

10) установлення єдиних нормативів з охорони праці для всіх підприємств залежно від форм власності і видів їх діяльності;

11) інформування населення, проведення навчання, професійної підготовки і підвищення кваліфікації працівників з питань охорони праці;

12) забезпечення координації діяльності органів державної влади, установ, організацій, об'єднань громадян, що розв'язують проблеми охорони здоров'я, гігієни та безпеки праці, а також співробітництва і проведення консультацій між роботодавцями та працівниками (їх представниками), між усіма соціальними групами під час прийняття рішень з охорони праці на місцевому та державному рівнях;

13) використання економічних методів управління охороною праці ,проведення політики пільгового оподаткування , що сприяє створенню безпечних та нешкідливих умов праці , участь держави в фінансуванні заходів з охорони праці

14) здійснення навчання населення, професійної підготовки та підвищення кваліфікації працівників з питань охорони праці

15) забезпечення координації діяльності державних органів, установ, організацій та об’єднання громадян , що вирішують різні проблеми охорони здоров’я, гігієни та безпеки праці, а також співробітництво .

Умови праці на робочому місці , безпека технологічних процесів, машин, механізмів, устаткування та інших засобів виробництва , стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці.

Власник або уповноважений ним орган повинен впроваджувати сучасні засоби техніки безпеки, які запобігають виробничому травматизму і забезпечувати санітарно-гігієнічні умови, що запобігають виникненню професійних захворювань працівників.

На власника або уповноважений ним орган покладається систематичне проведення інструктажу (навчання) працівників з питань охорони праці, протипожежної охорони

Охорона праці – один з центральних інститутів трудового права. Він має виключно практичне значення . Недодержання вимог охорони праці створює небезпеку для здоров'я і життя працівників. У свою чергу і ті, кого законодавець називає власником або уповноваженим ним органом, несуть сувору, у тому числі й кримінальну відповідальність за порушення правил охорони праці.

6.1 Аналіз небезпечних і шкідливих факторів у офісному приміщенні з персональними комп'ютерами

Працівники у офісному приміщенні, що працюють з персональними комп'ютерами піддаються шкідливій дії ПК, тому робочі місця користувачів повинні відповідати безпечним і нешкідливим умовам праці.

У зв'язку з цим передбачається розробити комплекс мір, що забезпечують безпечні і нешкідливі умови праці і розглянути екологічні питання.

При виконанні посадових обов'язків на працівників у офісному приміщені , що працюють з ПК постійно або періодично діють наступні небезпечні і шкідливі фактори:

– забруднення повітря шкідливими речовинами, пилом, мікроорганізмами;

– невідповідність нормам параметрів мікроклімату;

– виникнення на екрані монітора електростатичних зарядів , що змушують часточки пилу рухатися до найближчого заземленого предмету, часто нею виявляється особа користувача;

– підвищений рівень шуму на робочому місці;

– підвищений рівень статичної електрики при невірно спроектованій робочій зоні;

– небезпечний рівень напруги в електричному ланцюзі, замикання якого може відбутися через тіло людини;

– широкий спектр випромінювання від дисплея, що включає рентгенівську ультрафіолетову й інфрачервоні області, а так само широкий діапазон електромагнітних випромінювань інших частот;

– підвищений рівень електромагнітних випромінювань;

– підвищений рівень іонізуючих випромінювань;

– відсутність або недолік природного світла;

– розумова перенапруга , що обумовлена характером розв'язуваних задач приводить до синдрому тривалих психологічних навантажень (СТПН);

– великий обсяг інформації , що переробляється, приводить до значних навантажень на органи зору та нервову систему;

– монотонність праці;

– нервово-психічні навантаження;

– небезпека виникнення пожежі;

– недостатня освітленість робочої зони;

– підвищена яскравість світла;

– знижена контрастність;

– пряма і зворотна блесткість;

– підвищена пульсація світлового потоку (мерехтіння зображення);

– тривале перебування в тому самому положенні і повторення тих самих рухів приводить до синдрому тривалих статичних навантажень (СТСН);

– нераціональна організація робочого місця;

– невідповідність ергономічних характеристик устаткування нормованим величинам.

6.2 Розрахунок системи штучного освітлення у офісному приміщені з персональними комп’ютерами

У приміщені необхідно організувати змішане освітлення, тобто сполучення природного і штучного освітлення. Природне освітлення створюється бічним освітленням через вікно . Штучне освітлення використовується при недостатньому природному освітлені. У даному приміщені використовується загальне штучне освітлення [42].

Розрахунок його здійснюється по методу світлового потоку з урахуванням потоку, відбитого від стін і стелі

Потрібно зробити розрахунок загального освітлення для офісного приміщення. Довжина А=4,5 м, ширина В=3 м, висота Н=2,5 м. Висота робочої поверхні hp = 1 м. Для освітленя використовується Світильник УПМ-15. Мінімальна освітленість лампи накалювання по нормах Emin=75 лк. Коефіцієнт відображення стелі Sn= 70%, стін Sc=50%, робочої поверхні Sp=10%. Напруга мережі 220 В.

Відстань від стелі до робочої поверхні

Но = Н – hp = 2.5 – 1 = 1.5 м

Відстань від стелі до світильника

Нс= 0.2 * Но = 0.2 * 1.5 = 0.3 м

Висота підвісу світильника над освітленою поверхнею

Н = Но – hc = 1.5 – 0.3 = 1.2 м

Висота підвісу світильника над підлогою

Нn = h + hp = 1.2+1= 2.2 м

Для досягнення найбільшої рівномірності освітлення приймаємо відношення L/h = 1.5

Відстань між центрами світильників:

$L = 1.5 * h = 1.5 * 1.2 = 1.8$ м

Необхідна кількість світильників:

$N= S/l^2 = (4.5 *3) / 1.8^2 = 4.2 \sim 4$

Приймаємо кількість світильників: 4

Індекс приміщення:

$I = (A * B) / (h * (A+B)) = (4.5 * 3) / (1.2 * (4.5+3)) = 1.5$

Коефіцієнт використання світлового потоку $\eta= 0.51$

$Ф= (Emin * S * K3 * Z) / (N* \eta) = (75* 13.5 * 1.3 * 1.15) / (4* 0.51) =$ 741 лм

За знайденим світловим потоком з таблиці «Параметри ламп розжарювання загального призначення з розрахунковими напругами 130 и 220 В» обираємо лампу Б215-225-75 потужністю 75 Вт , що має світловий потік 950 лм , найбільш близький до розрахункового.

Фактична освітленість

Еф= Emin * Фл / Фр = 75* 950/741 = 96,15 лк

Загальна потужність

Робщ= Рл * N= 75*4 = 300 Вт = 0.3 кВт

Таким чином розрахунок загального освітлення для офісного приміщення можна вважати виконаним, а його результати такими, що задовольняють загальноприйнятим нормам промислового освітлення.

6.3 Розробка заходів щодо зменшення впливу небезпечних і шкідливих факторів у офісному приміщенні з персональними комп’ютерами

Нормована освітленість робочого столу з ВДТ і ПК повинна бути в межах 300 – 500 лк. Допускається установка світильників місцевого освітлення для підсвічування документів. При цьому місцеве освітлення так само, як і загальне

не повинно створювати відблисків на екрані відеомоніторів. Зовнішня освітленість екрана монітора не повинна перевищувати 300 лк [43].

Бажано, щоб природне світло падало ліворуч, хоча допускається і таке розміщення робочого місця, коли віконні прорізи розташовані праворуч. Вікна повинні бути обладнані пристроями типу жалюзі, зовнішніх козирків і інше.

Зниження шуму, створюваного ПК є дуже важливою задачею. Зниження шуму в джерелі випромінювання можна забезпечити застосуванням пружних прокладок між підставкою ПК і опорною поверхнею. Як прокладки використовуються гума, пробка, різної конструкції амортизатори. Під настільні шумливі апарати можна підкладати м’які килимки із синтетичних матеріалів, а під ніжки столів, на яких вони встановлені, прокладки з м’якої гуми, повсті товщиною 6 – 8 мм. Кріплення прокладок можливо шляхом приклейки їх до опорних частин.

Можливо також застосування звукоізолюючих кожухів, що не заважають технологічному процесові. Не менш важливим для зниження шуму в процесі експлуатації є питання правильного та своєчасного регулювання, змазування і заміни механічних вузлів шумливого устаткування.

Раціональне планування приміщення, розміщення устаткування є важливим чинником, що дозволяє знизити шум при існуючому устаткуванні ПК. Машинний зал і приміщення для сервісної апаратури необхідно розташовувати удалині від шумливого і вібруючого устаткування.

Зниження рівня шуму, що проникає у виробниче приміщення ззовні, може бути досягнуто збільшенням звукоізоляції конструкцій, що обгороджують, ущільненням по периметру притворів вікон, дверей.

У такий спосіб для зниження шуму створюваного на робочих місцях внутрішніми джерелами, а також шуму, що проникає ззовні, потрібно:

– послабити шум самих джерел (застосування екранів, звукоізолючих кожухів);

– знизити ефект сумарного впливу відбитих звукових хвиль (звуковбирні поверхні конструкцій);

– застосовувати раціональне розташування устаткування;

– використовувати архітектурно-планувальні і технологічні рішення ізоляцій джерел шуму.

До робочого столу теж пред’являються певні вимоги. Його конструкція повинна забезпечувати оптимальне розміщення на його площі використовуваного устаткування. Робочий стілець користувача необхідно розташувати з таким розрахунком, щоб він дозволяв змінювати позу працюючого, а також був підйомно-поворотним. Це допоможе уникнути стомлення м’язів шийно-плечевої області і спини. Оптимальна відстань від очей до екрана монітора 600 – 700 мм, але не ближче 500 мм з урахуванням розмірів алфавітно-цифрових знаків і символів. Недотримання цього параметра може призвести до розвитку стомлення і перевтоми здорового аналізатора і зі збільшенням тривалості роботи сприяти розвитку короткозорості. Відстань між робочими столами з відеотерміналами повинна бути не менш 2.0 м, а відстань між бічними поверхнями відеомоніторів не менш 1.2 м. Робочі місця з ВДТ і ПК у залах електронно-обчислювальних машин чи у приміщеннях із джерелами шкідливих виробничих факторів розміщаються в ізольованих кабінах з організованим повітрообміном.

7 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Охорона навколишнього природного середовища, раціональне використання природних ресурсів, забезпечення екологічної безпеки життєдіяльності людини – невід'ємна умова сталого економічного та соціального розвитку України [44].

З цією метою Україна здійснює на своїй території екологічну політику, спрямовану на збереження безпечного для існування живої і неживої природи навколишнього середовища, захисту життя і здоров'я населення від негативного впливу, зумовленого забруднення навколишнього природного середовища, досягнення гармонійної взаємодії суспільства і природи, охорону, раціональне використання і відтворення природних ресурсів.

Відносини у галузі охорони навколишнього природного середовища в Україні регулюється Законом України № 1268-XII від 26.06.91р. «Про охорону навколишнього природного середовища» , а також розроблюваними відповідно до нього земельним, водним, лісовим законодавством, законодавством про надра, про охорону атмосферного повітря, про охорону і використання рослинного і тваринного світу та іншим спеціальним законодавством.

Цей Закон визначає правові, економічні та соціальні основи організації охорони навколишнього природного середовища в інтересах нинішнього і майбутніх поколінь.

Завданням законодавства про охорону навколишнього природнього середовища є регулювання відносин у галузі охорони, використання і відтворення природних ресурсів, забезпечення екологічної безпеки, запобігання і ліквідації негативного впливу господарської та іншої діяльності та навколишнє природне середовище, збереження природних ресурсів, генетичного фонду живої природи, ландшафтів та інших природних комплексів, унікальних територій та природних об'єктів, пов'язаних з історико-культурною спадщиною.

Охорона навколишнього природного середовища є системою державних і суспільних заходів, направлених на збереження, відтворювання і раціональне використання природних ресурсів і поліпшення полягання природного середовища, і є частиною прикладної екології.

В міру прискорення темпів науково-технічного прогресу дія людей на природу стає все більш могутньою. І в даний час воно все сумирно із дією природних чинників, що приводить до якісної зміни співвідношення сил між суспільством і природою. На сучасному етапі людство поставлено перед чинником виникнення в природі незворотних процесів, нових шляхів переміщення і перетворення енергії і речовини. В природу втручається все більше і більше нових речовин, чужих їй, часом сильно токсичними для організмів. Частина з них не включається в природний круговорот і нагромаджується в біосфері, що приводить до небажаних екологічних наслідків.

7.1 Забруднення навколишнього середовища в процесі використання комп'ютерної техніки

Основну загрозу для навколишнього середовища в межах галузі розробки ПЗ ставлять ПК, а саме електромагнітні випромінювання від них.

Дослідження останніх років показано, що електромагнітні випромінювання вищезгаданих електричних пристроїв містять торсі онову компоненту, котра несе інформацію про процеси, що відбуваються в тому чи іншому електронному пристрої.

Торсіонові поля мають високу проникаючу здатність і їх неможливо за екранувати.

Останніми роками при проектування, виготовленні і експлуатації технічних і систем управляння в різних сферах діяльності надзвичайно широко застосовуються персональні комп'ютери і всілякі комп'ютерні програми.

Робота з комп’ютерами програмістів, операторів і інших користувачів пов’язана з додатковими шкідниками і небезпечними чинниками, що негативно впливають на організм людини. Наприклад, шкідлива дія на зір надає не при тримання стандартних візуальних ергономічних параметрів екрану, розмірів мінімального елементу відображення, мерехтіння зображення, відбивна здатність і ін. Працюючий комп’ютер створює електромагнітне поле, що шкідливо діє на організм людини. Це поле може виникати радіоперешкоди, тобто завантажити роботі радіо- і телеаппаратури, що призводить до зниження надійності технічної системи або системи управління, до збільшення риски виникнення аварійної ситуації і виробничому середовищі. Для забезпечення безпеки роботи з комп’ютером розроблені і повинні повсюди застосовуватися стандарти на монітори, вимоги до приміщення для експлуатації комп’ютерів і до організації і успадкування робочих місць.

7.2 Використання програмного забезпечення для вирішення екологічних проблем

Широко використовуються фахівцями-екологами програми розрахунку забруднення атмосфери та інші спеціалізовані екологічні програми на рівні окремого підприємства досить повно вирішують завдання, що стоять в цій області. Однак для вирішення міських завдань необхідні принципово інші, адаптовані для таких робіт програмні засоби.

Розроблена фірмою «Інтеграл» комп'ютерна система розрахункового моніторингу стану якості повітряного басейну міста (регіону) «Еколог-Місто» призначена для автоматизації діяльності територіальних природоохоронних органів, екологічних служб адміністрацій міст (регіонів), проведення зведених розрахунків забруднення атмосфери міст.

Результати зведених розрахунків забруднення атмосфери можуть бути використані з метою нормування викидів забруднюючих речовин і для вирішення інших завдань.

Система «Еколог-місто» складається з двох підсистем:

- підсистеми ведення банку даних;
- підсистеми отримання попередньої оцінки якості атмосферного повітря.

Взаємодія між підсистемами здійснюється як на рівні обміну інформацією, так і на рівні передачі управління.

Основним завданням підсистеми ведення банку даних є підготовка інформації про викиди шкідливих речовин в атмосферу та інших даних, необхідних для коректної роботи підсистеми отримання оцінок забруднення атмосфери.

Основними особливостями файлів даних, записаних в розробленому форматі є:

1) універсальність з точки зору можливості використання різних програмних засобів при створенні і перетворенні файлів. Текстові файли даних можуть бути створені і заповнені або за допомогою програм системи «Еколог-місто» або за допомогою будь-якого текстового редактора;

2) сегментування інформації в файлах.

Кожен файл даних складається з секцій, кожна з яких містить певний вид інформації.

Інформація в секціях різниться:

- по виду описуваних нею величин і властивостей.
- за рівнем описуваних нею об'єктів в ієрархії об'єктів, що розглядаються під час вирішення завдань оцінки забруднення атмосфери міста (регіону).

Використовується інформація наступних видів:

- про параметри джерел забруднення атмосфери;
- про інвентаризацію викидів забруднюючих речовин;

- про умови розсіювання домішок в атмосфері;
- про використовувані системах координат;
- про місцезнаходження на місцевості зон зі специфічними вимогами до якості атмосферного повітря;
- про узагальнених характеристиках міста, його районів, підприємств, їх майданчиків і цехів.

За рівнем описуваних об'єктів інформація сегментована на розділи, що містять опис:

- міста (регіону);
- району міста (регіону);
- підприємства;
- майданчики і цеху підприємства;
- джерела забруднення атмосфери, що належить конкретній ділянці (можливо, також і цеху) підприємства.

В єдиному форматі передбачено більш докладний і змістовний опис джерел забруднення атмосфери (ІЗА) за рахунок:

1) можливості завдання різних варіантів наборів параметрів одного і того ж ІЗА, відповідних:

2) різним режимам викиду забруднюючих речовин з ІЗА;

3) різних стадіях проведення заходів (реконструкції) на виробництві;

4) вказівки тимчасових інтервалів, до яких належить той чи інший набір характеристик ІЗА.

Підсистема ведення банку даних складається з чотирьох самостійних великих блоків, взаємодія між якими здійснюється на рівні обміну інформацією:

- блоку підготовки вихідної інформації;
- блоку ведення міського банку даних;

- блоків перетворення даних про джерела викидів з форматів, використовуваних в програмі УПРЗА "Еколог" та інших програмах в єдиний формат даних системи "Еколог-Місто";

Блок підготовки вихідної інформації надає користувачеві наступні можливості:

1) занести дані про джерела викидів підприємства або прийняти дані про джерела викидів підприємства в текстовому форматі з файлу на дискеті або на жорсткому диску;

2) провести перевірку коректності занесених даних;

3) записати занесені дані в текстовий формат для надання їх в цьому форматі на дискеті в територіальні органи охорони навколишнього середовища;

4) блок ведення міського банку даних дозволяє:

5) прийняти інформацію про джерела викидів підприємств з файлу в єдиному форматі, INT, або підготувати її безпосередньо;

6) переглянути інформацію у вигляді таблиць і при необхідності її доповнити;

7) переглянути взаємне розташування джерел на тлі топооснови місцевості як для окремого підприємства, так і для сукупності підприємств.

Блоки перетворення даних про джерела викидів з форматів, використовуваних в інших програмах, в єдиний формат даних системи "Еколог-місто" дозволяють в ході перетворення інформації:

- доповнити її даними, що відносяться до рівня підприємства;
- в разі необхідності, змінити, в напівавтоматичному режимі, кодування

ВИСНОВКИ

Кваліфікаційна робота присвячена підвищенню достовірності оцінювання розміру веб-застосунків, реалізованих мовою Java, та розробці відповідного ПЗ.

Під час виконання кваліфікаційної роботи поставлені завдання було виконано у повному обсязі. Одержані наступні результати:

– проаналізовано та порівняно існуючі методи та моделі для оцінювання розміру веб-застосунків, реалізованих мовою Java;

– обґрунтовано необхідність удосконалення рівняння регресії для оцінювання розміру веб-застосунків, реалізованих мовою Java;

– удосконалено однофакторне рівняння регресії для оцінювання розміру веб-застосунків, реалізованих мовою Java;

– досліджено та визначено веб-застосунки, реалізовані мовою Java, які були використані для перевірки побудованого рівняння регресії;

– отримано необхідні метрики з кожного проекту;

– перевірено емпіричні дані на викиди;

– нормалізовано отримані емпіричні дані, використовуючи нормалізуюче перетворення Джонсона сім'ї S_B;

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

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

– розроблено ПЗ для оцінювання розміру веб-застосунків, реалізованих мовою Java, та проведені відповідні випробування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. TIOBE Index for April 2020 [Електронний ресурс] / Режим доступу: \www/ URL: https://www.tiobe.com/tiobe-index/ – 12.09.2020 р. – Загол. з екрану.

2. Рейтинг мов програмування 2020: JavaScript випередив Java, а Dart увійшов у першу лігу [Електронний ресурс] / Режим доступу: \www/ URL: https://dou.ua/lenta/articles/language-rating-jan-2020/ – 12.09.2020 р. – Загол. з екрану.

3. Briand, L.C. Property Based Software Engineering Measurement / L.C. Briand, S. Morasca, V.R. Basili // IEEE Transaction on Software Engineering. – 2009. – Vol. 22, no. 1. – p. 68–86.

4. Briand, L.C. Encyclopedia of Software Engineering / L.C. Briand, I. Wieczorek – John Wiley & Sons, Inc., New Jersey, 2002. – 1584 p.

5. Prykhodko, S. Estimating the software size of open-source PHP-based systems using non-linear regression analysis / S. Prykhodko, N. Prykhodko, L. Makarova // in Proceedings of International Conference on Advanced Computer Information Technologies (ACIT 2018), June 1-3, 2018, Ceske Budejovice, Czech Republic. – P. 199 – 202.

6. Панькін, І.Д. Удосконалення однофакторного рівняння регресії для оцінювання розміру веб-застосунків, реалізованих мовою Java / І.Д. Панькін, Л.М. Макарова // Матеріали ІІІ 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.58-60.

7. 1061-1998-IEEE Standard for a Software Quality Metrics Methodology [Електронний ресурс] / Режим доступу: \www/ URL: https://ieeexplore.ieee.org/document/749159 – 17.09.2020 р. – Загол. з екрану.

8. Tan, H.B.K. Estimating LOC for information systems from their conceptual data models [Text] / H.B.K. Tan, Y. Zhao, H. Zhang // in Proceedings of the 28th International Conference on Software Engineering (ICSE '06), May 20-28, 2006, Shanghai, China, pp. 321-330.

9. Briand, L.C. Resource Estimation in Software Engineering [Text] / L.C. Briand, I. Wieczorek // Encyclopedia of Software Engineering. – John Wiley & Sons, Inc., 2002. – 83 p.

10. Kaczmarek, J. Size and effort estimation for applications written in Java [Text] / J. Kaczmarek, M. Kucharski // Information and SoftwareTechnology, 2004, Vol. 46, Issue 9, pp. 589–601.

11. Prykhodko, N. The non-linear regression model to estimate the software size of open source Java-based systems [Text] / N.V. Prykhodko, S.B. Prykhodko // Радіоелектроніка, інформатика, управління. – 2018. – № 3. – С.158-166.

12. Prykhodko, N.V. Constructing the Nonlinear Regression Models on the Basis of Multivariate Normalizing Transformations [Text] / N.V. Prykhodko, S.B. Prykhodko // Електронне моделювання. – 2018. – Т. 40. № 6. – С.99-108.

13. Makarova, L.M. Information technology for size estimation of web-applications implemented in Java / L.M. Makarova, A.S. Andreeva // Інновації в суднобудуванні та океанотехніці: матеріали IX Міжнародної науково-технічної конференції. – Миколаїв: НУК, 2018 – С.405.

14. Дрейпер, Н. Прикладной регрессионный анализ: В 2-х кн. Кн. 2 / Пер. с англ. – 2-е изд., перераб. и доп. [Текст] / Н. Дрейпер, Г. Смит – М.: Финансы и статистика, 1987. – 351 с.

15. Айвазян, С.А. Прикладная статистика. Основы эконометрики: Учебник для вузов: В 2 т. 2-е изд., испр. – Т. 1: Теория вероятностей и прикладная статистика [Текст] / С.А. Айвазян, В.С. Мхитарян – М.: ЮНИТИ-ДАНА, 2001. – 656 с.

16. Кобзарь, А.И. Прикладная математическая статистика. Для инженеров и научных работников [Текст] / А.И. Кобзарь – М.: ФИЗМАТЛИТ, 2006. – 816 с.

17. Chatterjee, Samprit. Handbook of Regression Analysis [Text] / Samprit Chatterjee, Jeffrey S. Simonoff. – Wiley, 2012. – 240 p.

18. Prykhodko, S.B. Developing the software defect prediction models using regression analysis based on normalizing transformations / S.B. Prykhodko // in “Modern problems in testing of the applied software” (PTTAS-2016), Abstracts of the Research and Practice Seminar, Poltava, Ukraine, May 25-26, 2016, pp. 6-7.

19. Yan, Xin. Linear regression analysis: theory and computing [Text] / Xin Yan, Xiao Gang Su – Singapore: World Scientific Publishing Co. Pte. Ltd., 2009. – 328 p.

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

21. Карпов, И.Г. Модифицированные распределения Джонсона и их применение для аппроксимации законов распределения экспериментальных данных [Текст] / И.Г. Карпов, Ю.Т. Зырянов, А.Н. Грибков. // Известия Томского политехнического университета. – 2013. – Т. 322. – № 2. – С. 46–50.

22. Кендалл, М. Теория распределений [Текст] / М. Кендалл, А. Стюарт. – Пер. с англ. под ред. А.Н. Колмогорова. – М.: Наука. Гл. ред. физ.-мат. лит., 1966. – 588 с.

23. Seber, George A.F. Nonlinear Regression [Text] / George A.F. Seber, C.J. Wild. – John Wiley & Sons, Inc., 2003. – 792 p.

24. Johnson, N.L. System of Frequency Curves Generated by Methods of Translation [Text] / N.L. Johnson // Biometrica. – 1949. – Vol. 36, № ½. – P. 149–176.

25. Приходько, С.Б. Аналитическая зависимость для выбора семейства распределений Джонсона [Текст] / С.Б. Приходько, Л.Н. Макарова, А.С. Приходько // Проблеми інформаційних технологій. – Херсон: ХНТУ, 2016. – №02 (020). – С.105-110.

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

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

28. Приходько, С.Б. Методичні вказівки та завдання до виконання лабораторних робіт з дисципліни "Обробка експериментальних даних на комп`ютері" [Текст] / С.Б. Приходько, Л.Н. Макарова, К.С. Пугаченко – Миколаїв: НУК, 2018. – 76 с.

29. 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 .

30. Transform data to normal distribution [Електронний ресурс] / Режим доступу: \www/ URL: http://www.sigmamagic.com/forum/archives/297 – 17.10.2020 р. – Загол. з екрану.

31. Аппроксимация закона распределения экспериментальных данных – [Електронний ресурс] / Режим доступу: \www/ URL: http://opds.sut.ru/old/electronic_manuals/oed/f05.htm#r054 – 17.10.2020 р. – Загол. з екрану.

32. GitHub: Where the world builds software [Електронний ресурс] / Режим доступу: \www/ URL: https://github.com/ – 18.10.2020 р – Загол. з екрану.

33. Your teammate for Code Quality and Security [Електронний ресурс] / Режим доступу: \www/ URL: https://www.sonarqube.org/ – 18.10.2020 р – Загол. з екрану.

34 Уніфікована мова моделювання [Електронний ресурс] / Режим доступу: \www/ URL: https://bibliofond.ru/view.aspx?id=446563#1 – 25.10.2020 р – Загол. з екрану.

35. Діаграма активності [Електронний ресурс] / Режим доступу: \www/ URL: http://khpi-iip.mipk.kharkiv.edu/library/case/leon/gl7/gl7.html – 25.10.2020 р – Загол. з екрану.

36. Діаграма станів [Електронний ресурс] / Режим доступу: \www/ URL: http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?id=2808710 – 25.10.2020 р – Загол. з екрану.

37. Раскин, Джеф. Интерфейс: новые направления в проектировании компьютерных систем [Текст] / Дж. Раскин. – К.: Символ-Плюс, 2015. – 272 с.

38. Діаграма класів [Електронний ресурс] / Режим доступу: \www/ URL: http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?id=28088 – 28.10.2020 р – Загол. з екрану.

39. Діаграма послідовності [Електронний ресурс] / Режим доступу: \www/ URL: http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?r=16538 – 28.10.2020 р – Загол. з екрану.

40. Шилтд, Герберт. Java. Полное руководство. 10-е издание [Текст] / Г. Шилтд. – К.: Диалектика, 2019.– 730 с.

41. Поняття, мета і завдання охорони праці [Електронний ресурс] / Режим доступу: \www/ URL: http://ncpn.net.ua/oxrana_truda.html – 05.11.2020 р – Загол. з екрану.

42. ДСТУ Б В.2.2-6-97 (ГОСТ 24940-96) Здания и сооружения. Методы измерения освещенности [Текст]. – Введ. 1997-09-15 – М.: Стандартинформ, 1998. – 4 с.

43. 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. – 4 с.

44. Охорона навколишнього середовища [Електронний ресурс] / Режим доступу: \www/ URL: http://pidruchniki.com/13351207/ekonomika/ohorona_navkolishnogo_seredovisha – 05.11.2020 р – Загол. з екрану.

Додаток А Технічне завдання

Вступ

Назва розроблюваного проекту: «Програмне забезпечення для оцінювання розміру веб-застосунків, реалізованих мовою Java», коротка назва – «JavaWebAnalytics». Область застосування – підприємства, які займаються розробкою ПЗ.

1. Підстави для розробки

Підставою для розробки є завдання на кваліфікаційну роботу, видане кафедрою програмного забезпечення автоматизованих систем ННІКНУП НУК ім. адмірала Макарова.

2. Призначення розробки

2.1 Експлуатаційне призначення

Експлуатаційним призначенням ПЗ є спрощення бізнес-процесів IT-компаній, які займаються розробкою веб-застосунків, реалізованих мовою Java.

2.2 Функціональне призначення програми

Функціональним призначенням програмного забезпечення є автоматизація процесів:

– вводу статистичних даних;

– нормалізації вибірок за допомогою перетворення Джонсона;

– розрахунку параметрів для вхідних та нормалізованих вибірок;

– побудови лінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування для лінійного рівняння регресії;

– побудови нелінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування для нелінійного рівняння регресії;

– оцінювання розміру веб-застосунків, реалізованих мовою Java.

3. Вимоги до програмного забезпечення

3.1 Вимоги до функціональних характеристик

3.3.1 Вимоги до складу виконуваних функцій

Програмне забезпечення повинно виконувати наступні функції:

1) Загальні функції:

– можливість завантажити файл з вихідними даними у форматах Excel;

– можливість змінити файл з вихідними даними;

– зчитування даних з файлу;

– розрахунок параметрів для вхідних вибірок;

– розрахунок параметрів перетворення Джонсона;

– нормалізація вихідних вибірок за допомогою параметрів перетворення Джонсона;

– розрахунок параметрів для нормалізованих вибірок;

– перевірка вибірок на нормальність розподілу;

– побудова лінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування для лінійного рівняння регресії;

– побудова нелінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування для нелінійного рівняння регресії;

– оцінювання розміру веб-застосунків, реалізованих мовою Java.

3.1.2 Вимоги до організації вхідних та вихідних даних

Введення даних здійснюється за допомогою пристроїв введення даних (клавіатура або миша). Дані вводяться через завантаження файлу у вікні програми та за допомогою ручного вводу.

Виведення даних здійснюється за допомогою пристрою виведення даних – монітору комп’ютера.

Вхідні дані:

- файл у форматах Excel;
- параметри нормалізації Джонсона (λ, ϕ, η, γ) для першого наближення.

Вихідні дані:

1) Розраховані параметри для вихідної вибірки:

кількість значень вибірки, вибіркове середнє, дисперсія, середньоквадратичне відхилення, асиметрія, квадрат асиметрії, ексцес.

2) Розраховані параметри перетворення Джонсона:

λ, ϕ, η, γ.

3) Розраховані параметри для нормалізованої вибірки:

кількість значень вибірки, вибіркове середнє, дисперсія, середньоквадратичне відхилення, асиметрія, квадрат асиметрії, ексцес.

4) Параметри для перевірки вхідних та нормалізованих вибірок на нормальність розподілу:

значення χ^2 розрахункове, значення χ^2 критичне.

5) Графік побудованого лінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування для лінійного рівняння регресії виводиться на монітор користувача. Параметри лінійного рівняння регресії записуються у вихідний файл.

6) Графік для побудованого нелінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування для нелінійного рівняння регресії виводиться на монітор користувача. Параметри нелінійного рівняння регресії записуються у вихідний файл.

3.2 Вимоги до надійності

Програмний продукт повинен нормально функціонувати при безперебійній роботі ПК. При виникненні збоїв в роботі, відновлення нормальної роботи повинне проводитися після перезавантаження програми.

Відмови ПЗ можливі внаслідок некоректних дій користувача при взаємодії з ПЗ. Щоб зменшити кількість відмов ПЗ за вказаною вище причиною слід забезпечити зрозумілість ПЗ через надання підказок користувачеві у разі невірних дій.

3.3 Вимоги до умов експлуатації

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

температура повітря від +10^{o}C до +30^{o}C; 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 користувача повинна задовольняти вказаним мінімальним вимогам:

– CPU: Intel Pentium 4405Y (1.2 ГГц);
– RAM: 1 GB;
– 500 Mb вільного місця на диску.

3.5 Вимоги до інформаційної та програмної сумісності

Для повноцінного функціонування ПЗ необхідно використовувати: ОС Windows версії не нижче 7.

3.6 Вимоги до маркування та пакування

Додаток для оцінювання розміру веб-застосунків, реалізованих мовою Java, буде упаковано в електронний архів і мати найменування«JavaWebAnalytics.rar».

3.7 Вимоги до транспортування та зберігання

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

4. Вимоги до програмної документації

Програмна документація повинна містити:

– технічне завдання.
– опис програми
– текст програми
– інструкція користувача
– програма і методика випробувань ПЗ

5. Техніко-економічні показники не розраховувалися.

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. 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	4.1 Розробка робочого проекту	01.11.20	07.11.20
	4.2 Затвердження робочого проекту	07.11.20	15.11.20
	4.3 Розробка документації	15.11.20	20.11.20

7. Порядок контролю та прийому

Контроль за аналізом та проектуванням кожної окремої частини програмного забезпечення для автоматизації обробки інформації для оцінювання розміру веб-проектів мовою Java, відбувається на кожному етапі з

урахуванням вимог, визначених у технічному завдані. Кожна стадія розробки повинна бути представлена в зазначені строки та узгоджена із замовником.

Прийом готового програного проекту документують за допомогою протоколу проведення випробувань. У разі знаходження помилок під час прийому програмного виробу складається акт про знайдені помилки, який підписується представниками замовника і розробника і затверджується керівниками організації – замовника та організації – розробника. Розробник повинен протягом не більше ніж 2 тижнів виправити зазначені помилки, і оповістити замовника про повторне проведення перевірки.

Додаток Б Текст програми

```
package com.data.source.file;

public class TableValues {

   private BufferedReader file;
   private final Double[][] tableValues;
   private String tableCaption;
   private boolean argumentAscending;
   private boolean precisionAscending;
   private boolean withArgumentInterpolation;

   protected TableValues(String tableCaption, String filePath, boolean argumentAscending, boolean
precisionAscending) throws FunctionException {
      this(tableCaption, filePath, argumentAscending, precisionAscending, false);
   }

   protected TableValues(String tableCaption, String filePath, boolean argumentAscending, boolean
precisionAscending, boolean withInterpolation)
         throws FunctionException {
      this.tableCaption = tableCaption;
      this.argumentAscending = argumentAscending;
      this.precisionAscending = precisionAscending;
      this.withArgumentInterpolation = withInterpolation;
      try {
         this.file = new BufferedReader(new InputStreamReader(new FileInputStream(filePath)));
         tableValues = loadData();
      } catch (FileNotFoundException ex) {
         throw new FunctionException("Файл не найден: " + filePath, ex);
      }
   }

   private Double[][] loadData() throws FunctionException {
      ArrayList<Double[]> data = new ArrayList<Double[]>();
      int columnCount = 0;
      try {
         String readLine = "";
         Double[] row;
         while (readLine != null) {
            readLine = file.readLine();
            if (readLine == null) {
               continue;
            }
            String[] line = readLine.split(";");
            row = new Double[line.length];
            int i;
            for (i = 0; i < line.length; i++) {
               //DecimalFormat df = new DecimalFormat("0.0");
               row[i] = NumberFormat.getNumberInstance().parse(line[i]).doubleValue();
            }
            data.add(row);
            if (columnCount != 0 && i != columnCount) {
               throw new FunctionException("Load " + tableCaption + " values error - different columns count:
previos "
                     + columnCount + ", current " + i);
            }
            columnCount = i;
```

```
                }
            } catch (IOException ex) {
                throw new FunctionException(ex);
            } catch (ParseException ex) {
                throw new FunctionException(ex);
            }
            //System.arraycopy
            Double[][] res = new Double[data.size()][columnCount];
            return data.toArray(res);
        }

        public double getValue(double argument, double precision) throws FunctionException {
            int indexPrecision = 0;
            int indexArgument = 0;
            int indexArgument2 = 0;
            double res;
            for (int i = 1; i < tableValues[0].length; i++) {
                double tPrecision = tableValues[0][i];
                if (precisionAscending && tPrecision < precision) {
                    indexPrecision = i;
                } else if (!precisionAscending && tPrecision > precision) {
                    indexPrecision = i;
                } else {
                    if (tPrecision == precision) {
                        indexPrecision = i;
                    }
                    break;
                }
            }

            for (int i = 1; i < tableValues.length; i++) {
                double tArgument = tableValues[i][0];
                if (argumentAscending && tArgument < argument) {
                    indexArgument = i;
                } else if (!argumentAscending && tArgument > argument) {
                    indexArgument = i;
                } else {
                    if (tArgument == argument) {
                        indexArgument = i;
                    } else if (withArgumentInterpolation) {
                        indexArgument2 = i;
                    }
                    break;
                }
            }
            res = tableValues[indexArgument][indexPrecision];
            if (withArgumentInterpolation) {
                if (indexArgument2 == 0) {
                    indexArgument2 = indexArgument;
                }
                double res2 = tableValues[indexArgument2][indexPrecision];
                res = -((res - res2) * argument + (tableValues[indexArgument][0] * res2 -
tableValues[indexArgument2][0] * res))
                        / (tableValues[indexArgument2][0] - tableValues[indexArgument][0]);
            }

            return res;
        }
    }

    package com.data.distribution;

    class InvertNormalDistributionValues extends TableValues {
```

```
    private static InvertNormalDistributionValues instance;

    private InvertNormalDistributionValues() throws FunctionException {
      super("Pirson distribution percentage points", (String) OptionsController.getInstance().getOption(
          OptionType.INVERT_NORMAL_DISTRIBUTION_VALUES_FILE_PATH).getValue(), true,
true);
    }

    public static InvertNormalDistributionValues getInstance() throws FunctionException {
      if (instance == null) {
        instance = new InvertNormalDistributionValues();
      }
      return instance;
    }
  }

  package com.data.distribution;

  class PercentagePointsStudentDistribution extends TableValues {

    private static PercentagePointsStudentDistribution instance;

    private PercentagePointsStudentDistribution() throws FunctionException {
      super("Student distribution percentage points", (String) OptionsController.getInstance().getOption(
          OptionType.PERCENTAGE_POINTS_STUDENT_DISTRIBUTION_FILE_PATH).getValue(),
true, false, true);
    }

    public static PercentagePointsStudentDistribution getInstance() throws FunctionException {
      if (instance == null) {
        instance = new PercentagePointsStudentDistribution();
      }
      return instance;
    }
  }

  package com.data.function.check.kolmogorov;

  public class LaplasFunctionValues extends TableValues {

    private static LaplasFunctionValues instance;

    private LaplasFunctionValues() throws FunctionException {
      super("laplas function", (String) OptionsController.getInstance().getOption(
          OptionType.LAPLAS_FUNCTION_VALUES_FILE_PATH).getValue(), true, true);
    }

    public static LaplasFunctionValues getInstance() throws FunctionException {
      if (instance == null) {
        instance = new LaplasFunctionValues();
      }
      return instance;
    }
  }

  package com.data.function.check.pirson;

  public class Hi2CriticalValues extends TableValues {

    private static Hi2CriticalValues instance;

    private Hi2CriticalValues() throws FunctionException {
```

```
        super("Hi2 critical", (String) OptionsController.getInstance().getOption(
                OptionType.HI2_CRITICAL_VALUES_FILE_PATH).getValue(), true, false);
    }

    public static Hi2CriticalValues getInstance() throws FunctionException {
        if (instance == null) {
            instance = new Hi2CriticalValues();
        }
        return instance;
    }
}

package com.settings;

public class OptionDefinition {

    private final OptionType option;
    private final String caption;
    private final ElementTypes type;
    private final Object value;
    private final OptionGroup group;
    private final OptionAccess access;
    private final List<Map.Entry<Object, String>> valueList;

    public OptionDefinition(OptionType option, String caption, ElementTypes type,
            Object value, OptionGroup group, OptionAccess access, List<Map.Entry<Object, String>> valueList)
{
        this.option = option;
        this.caption = caption;
        this.type = type;
        this.value = value;
        this.group = group;
        this.access = access;
        this.valueList = valueList;
    }

    public OptionAccess getAccess() {
        return access;
    }

    public String getCaption() {
        return caption;
    }

    public OptionGroup getGroup() {
        return group;
    }

    public OptionType getOption() {
        return option;
    }

    public ElementTypes getType() {
        return type;
    }

    public Object getValue() {
        return value;
    }

    public List<Map.Entry<Object, String>> getValueList() {
        return valueList;
    }
```

```
}

package com.settings;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.util.AbstractMap;
import java.util.ArrayList;
import java.util.EnumMap;
import java.util.List;
import java.util.Map;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import javax.xml.bind.Unmarshaller;

public class OptionsController {

   private static OptionsController controller;
   private static final String JAXB_CONTEXT_PATH = "com.mln.atmstat.data.storage.options";
   private static final String OPTIONS_FILENAME = "options.xml";
   private ObjectFactory objFactory;
   private JAXBContext jaxbContext;
   private OptionCollection optionCollection;
   private Map<OptionType, Integer> optionsIndex;
   private String filename = OPTIONS_FILENAME;

   private OptionsController() throws JAXBException, OptionException {
      this(false);
   }

   // только для первого заведения
   private OptionsController(boolean withoutLoad) throws JAXBException, OptionException {
      objFactory = new ObjectFactory();
      jaxbContext = JAXBContext.newInstance(JAXB_CONTEXT_PATH);
      optionCollection = objFactory.createOptionCollection();
      optionsIndex = new EnumMap<>(OptionType.class);
      if (!withoutLoad) {
         load();
      }
   }

   private void load() throws JAXBException, OptionException {
      Unmarshaller unmarshaller = jaxbContext.createUnmarshaller();
      File file = new File(filename);
      if (file.exists()) {
         optionCollection = (OptionCollection) unmarshaller.unmarshal(file);
         for (int i = 0; i < optionCollection.getOption().size(); i++) {
            optionsIndex.put(OptionType.valueOf(optionCollection.getOption().get(i).getName()), i);
         }
         for (OptionType opt : OptionType.values()) {
            if (optionsIndex.get(opt) == null) {
               throw new IllegalArgumentException("Option value not define: " + opt);
            }
         }
      } else {
         throw new OptionException("Файл опций не найден: " + filename);
      }
   }

   public void save() {
      try {
```

```
            Marshaller marshaller = jaxbContext.createMarshaller();
            marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);
            marshaller.marshal(optionCollection, new FileOutputStream(filename));
        } catch (FileNotFoundException | JAXBException ex) {
            AtmStatApp.logException(ex);
        }
    }

    public void setOption(OptionType option, Object value) {
        Option opt = optionCollection.getOption().get(optionsIndex.get(option));
        opt.setValue(convertValueToString(ElementTypes.valueOf(opt.getType()), value));
    }

    private String convertValueToString(ElementTypes type, Object value) {
        if (value == null) {
            return "";
        }
        switch (type) {
            case CHECKBOX:
//              return (Boolean) value ? "1" : "0";
            case COMBOBOX:
            case NUMERIC:
            case INTEGER:
            case TEXT:
            default:
                return value.toString();
        }
    }

    // только для первого заведения
    public void createOption(OptionDefinition definition) {
        Option opt = objFactory.createOption();
        opt.setName(definition.getOption().name());
        opt.setCaption(definition.getCaption());
        opt.setType(definition.getType().name());
        opt.setValue(convertValueToString(definition.getType(), definition.getValue()));
        opt.setGroup(definition.getGroup().name());
        opt.setAccess(definition.getAccess().name());
        if (definition.getValueList() != null) {
            ValueList items = new ValueList();
            for (Map.Entry<Object, String> valueItem : definition.getValueList()) {
                ValueItem item = new ValueItem();
                item.setId((Integer) valueItem.getKey());
                item.setValue(valueItem.getValue());
                items.getItem().add(item);
            }
            opt.setValueList(items);
        }
        optionCollection.getOption().add(opt);
        optionsIndex.put(definition.getOption(), optionCollection.getOption().size() - 1);
    }

    public OptionDefinition getOption(OptionType option) {
        Option opt = optionCollection.getOption().get(optionsIndex.get(option));
        List<Map.Entry<Object, String>> valueList = null;
        if (opt.getValueList() != null) {
            valueList = new ArrayList<Map.Entry<Object, String>>();
            for (ValueItem item : opt.getValueList().getItem()) {
                valueList.add(new AbstractMap.SimpleEntry<Object, String>(item.getId(), item.getValue()));
            }
        }
        return new OptionDefinition(option, opt.getCaption(), ElementTypes.valueOf(opt.getType()),
                convertStringToValue(ElementTypes.valueOf(opt.getType()), opt.getValue()),
```

```
                OptionGroup.valueOf(opt.getGroup()), OptionAccess.valueOf(opt.getAccess()), valueList);
        }

        private Object convertStringToValue(ElementTypes type, String value) {
            if (value.isEmpty()) {
                return null;
            }
            switch (type) {
                case CHECKBOX:
                    return Boolean.valueOf(value);
                case NUMERIC:
                    return Double.valueOf(value);
                case COMBOBOX:
                case INTEGER:
                    return Integer.valueOf(value);
                case TEXT:
                default:
                    return value;
            }
        }

        void setFilename(String filename) {
            this.filename = filename;
        }

        public static OptionsController getInstance() {
            return getInstance(false);
        }

        public static OptionsController getInstance(boolean withoutLoad) {
            if (controller == null) {
                try {
                    controller = new OptionsController(withoutLoad);
                } catch (OptionException | JAXBException ex) {
                    AtmStatApp.logException(ex);
                }
            }
            return controller;
        }
    }

    package com.data;

    public class DataManagerEnvironmentController {

        private DataSourceControllerI dataSourceController;
        private ProblemGroupingMode problemGroupingMode;
        private int minProblemEventCount;
        private int maxProblemEventCount;

        public DataManagerEnvironmentController() throws DataControllerException {
            init();
        }

        final public void init() throws DataControllerException {
            MySQLDBController dbctrl = new MySQLDBController();
            OptionsController options = OptionsController.getInstance();
            boolean isGetDataFromDB = (Integer) options.getOption(OptionType.DATA_SOURCE).getValue() ==
0;
            DataSourceControllerI ctrl;
            if (isGetDataFromDB) {
                ctrl = new DBDataController(dbctrl, (String) options.getOption(OptionType.DB_URI).getValue());
            } else {
```

```
                ctrl = new FileDataController((String) options.getOption(OptionType.FILE_PATH).getValue());
            }
            this.dataSourceController = ctrl;
            this.problemGroupingMode = ProblemGroupingMode.values()[(Integer)
(OptionsController.getInstance().getOption(OptionType.PROBLEM_GROUPING_MODE)).getValue()];
            this.minProblemEventCount = (Integer)
OptionsController.getInstance().getOption(OptionType.MINIMUM_PROBLEM_EVENT_COUNT).getValue();
            this.maxProblemEventCount = (Integer)
OptionsController.getInstance().getOption(OptionType.MAXIMUM_PROBLEM_EVENT_COUNT).getValue();
        }

        public DataSourceControllerI getDataSourceController() {
            return dataSourceController;
        }

        public int getMaxProblemEventCount() {
            return maxProblemEventCount;
        }

        public int getMinProblemEventCount() {
            return minProblemEventCount;
        }

        public ProblemGroupingMode getProblemGroupingMode() {
            return problemGroupingMode;
        }
    }

    package com.data.distribution;

    import java.security.MessageDigest;
    import java.security.NoSuchAlgorithmException;
    import java.util.List;
    import java.util.logging.Level;
    import java.util.logging.Logger;

    public class DistributionData extends CalculateSetData {

        private String hash = "";

        public DistributionData(CalculateSetValues params, List<CalculateSetValues> intervals) {
            super(params, intervals);
            for (CalculateDataDatatypes datatype : DistributionDatatypes.values()) {
                boolean res = false;
                for (CalculateDataDatatypes gettedDatatype : params.getEtalonDatatypes()) {
                    if (datatype.getCaption().equals(gettedDatatype.getCaption())) {
                        res = true;
                    }
                }
                if (!res) {
                    throw new IllegalArgumentException("Not defined distribution param in calculate set of values: " +
datatype.getCaption());
                }
            }
            if ((Integer) params.getParam(DistributionDatatypes.INTERVAL_COUNT) != intervals.size()) {
                throw new IllegalArgumentException("Incorrect count distribution intervals");
            }
            setDistributionHash();
        }

        private void setDistributionHash() {
            MessageDigest md;
            try {
```

```
                md = MessageDigest.getInstance("SHA1"); //SHA-256 SHA1
                String hashData = getParam(DistributionDatatypes.COUNT).toString() +
getParam(DistributionDatatypes.DEVIATION)
                    + getParam(DistributionDatatypes.INTERVAL_COUNT) +
getParam(DistributionDatatypes.MIN_VALUE)
                    + getParam(DistributionDatatypes.MAX_VALUE);
                byte[] array = md.digest(hashData.getBytes());
                for (byte b : array) {
                   this.hash += Integer.toHexString(b + 128);
                }
             } catch (NoSuchAlgorithmException ex) {
                Logger.getLogger(DistributionData.class.getName()).log(Level.SEVERE, null, ex);
             }
          }

          public String getHash() {
             return hash;
          }
       }

       package com.data.distribution;

       import java.math.BigDecimal;
       import java.math.MathContext;
       import java.util.ArrayList;
       import java.util.LinkedHashMap;
       import java.util.List;
       import java.util.Map;

       public class DistributionCalculator {

          public static CalculateSetValues calcDistributionValues(double[] events) {
             int count;
             double average;
             double dispersion;
             double deviation;
             double asymmetry;
             double squaredAsymmetry;
             double excess;
             int intervalCount;
             double minValue;
             double maxValue;
             double intervalWidth;
             double avgConfidenceInt1;
             double avgConfidenceInt2;
             double deviationConfidenceInt1;
             double deviationConfidenceInt2;
             count = events.length;
             //AVERAGE
             double sum = getEventSum(events, 0, 1);
             average = sum / count;
             //DISPERSION
             sum = getEventSum(events, average, 2);
             dispersion = sum / count;
             //DEVIATION
             deviation = Math.sqrt(dispersion);
             //ASYMMETRY
             sum = getEventSum(events, average, 3);
             asymmetry = sum / count / Math.pow(deviation, 3);
             //SQUARED_ASYMMETRY
             squaredAsymmetry = Math.pow(asymmetry, 2);
             //EXCESS
             sum = getEventSum(events, average, 4);
```

```
            excess = sum / count / Math.pow(deviation, 4);
            //INTERVAL_COUNT
            intervalCount = (int) Math.round(3.322 * Math.log10(count) + 1);
            //MIN_VALUE,INTERVAL_WIDTH
            minValue = Double.MAX_VALUE;
            maxValue = Double.MIN_VALUE;
            for (double val : events) {
               if (minValue > val) {
                  minValue = val;
               }
               if (maxValue < val) {
                  maxValue = val;
               }
            }
            intervalWidth = (maxValue - minValue) / intervalCount;
            // confidence intervals
            double precision = getDefaultPrecision();
            avgConfidenceInt1 = average - PercentagePointsStudentDistributionCalculator.getValue(count, precision)
                  * deviation / Math.sqrt(count);
            avgConfidenceInt2 = average + PercentagePointsStudentDistributionCalculator.getValue(count,
precision)
                  * deviation / Math.sqrt(count);
            deviationConfidenceInt1 = deviation * Math.sqrt(count
                  / PercentagePointsPirsonDistributionCalculator.getValue(count, 1 - precision / 2));
            deviationConfidenceInt2 = deviation * Math.sqrt(count
                  / PercentagePointsPirsonDistributionCalculator.getValue(count, precision / 2));
            // ----------
            Map<CalculateDataDatatypes, Object> params = new LinkedHashMap();
            params.put(DistributionDatatypes.COUNT, count);
            params.put(DistributionDatatypes.AVERAGE, average);
            params.put(DistributionDatatypes.DISPERSION, dispersion);
            params.put(DistributionDatatypes.DEVIATION, deviation);
            params.put(DistributionDatatypes.ASYMMETRY, asymmetry);
            params.put(DistributionDatatypes.SQUARED_ASYMMETRY, squaredAsymmetry);
            params.put(DistributionDatatypes.EXCESS, excess);
            params.put(DistributionDatatypes.INTERVAL_COUNT, intervalCount);
            params.put(DistributionDatatypes.MIN_VALUE, minValue);
            params.put(DistributionDatatypes.MAX_VALUE, maxValue);
            params.put(DistributionDatatypes.INTERVAL_WIDTH, intervalWidth);
            params.put(DistributionDatatypes.AVERAGE_CONFIDENCE_INTERVAL1, avgConfidenceInt1);
            params.put(DistributionDatatypes.AVERAGE_CONFIDENCE_INTERVAL2, avgConfidenceInt2);
            params.put(DistributionDatatypes.AVERAGE_CONFIDENCE_INTERVAL_LENGTH,
avgConfidenceInt2 - avgConfidenceInt1);
            params.put(DistributionDatatypes.DEVIATION_CONFIDENCE_INTERVAL1,
deviationConfidenceInt1);
            params.put(DistributionDatatypes.DEVIATION_CONFIDENCE_INTERVAL2,
deviationConfidenceInt2);
            params.put(DistributionDatatypes.DEVIATION_CONFIDENCE_INTERVAL_LENGTH,
deviationConfidenceInt2 - deviationConfidenceInt1);
            return new CalculateSetValues(params, DistributionDatatypes.values());
         }

         public static List<CalculateSetValues> calcDistributionIntervals(CalculateSetValues params, double[]
events) {
            ArrayList<CalculateSetValues> intervalsList = new ArrayList();
            int intervalCount = (Integer) params.getParam(DistributionDatatypes.INTERVAL_COUNT);
            double intervalMin = (Double) params.getParam(DistributionDatatypes.MIN_VALUE);
            double intervalWidth = (Double) params.getParam(DistributionDatatypes.INTERVAL_WIDTH);
            int[] intervals = new int[intervalCount];
            for (double val : events) {
               int ind = (int) ((val - intervalMin) / intervalWidth);
               if (ind > intervalCount - 1) {
                  ind = intervalCount - 1;
```

```
                }
                intervals[ind]++;
            }
            for (int i = 0; i < intervalCount; i++) {
                Map<CalculateDataDatatypes, Object> vparams = new LinkedHashMap();
                vparams.put(DistributionIntervalDatatypes.START, intervalMin + intervalWidth * i);
                vparams.put(DistributionIntervalDatatypes.END, intervalMin + intervalWidth * (i + 1));
                vparams.put(DistributionIntervalDatatypes.FREQUENCY, intervals[i]);
                double relFreq = ((double) intervals[i]) / events.length;
                vparams.put(DistributionIntervalDatatypes.RELATIVE_FREQUENCY, relFreq);
                vparams.put(DistributionIntervalDatatypes.REDUCED_FREQUENCY, relFreq / intervalWidth);
                CalculateSetValues v = new CalculateSetValues(vparams, DistributionIntervalDatatypes.values());
                intervalsList.add(v);
            }
            return intervalsList;
        }

        protected static double getEventSum(double[] events, double average, int powDegree) {
            double sum = 0;
            for (double val : events) {
                sum += Math.pow((double) val - average, powDegree);
            }
            return sum;
        }

        private static double getDefaultPrecision() {
            double precision = (Double)
OptionsController.getInstance().getOption(OptionType.FUNCTION_CHECK_PRECISION).getValue();
            return new BigDecimal(1 - precision).round(new MathContext(3)).doubleValue();
        }

    }
```

Додаток В Опис програми

ПЗ «JavaWebAnalytics» для оцінювання розміру веб-застосунків, реалізованих мовою Java, спрямоване на використання на підприємствах, які займаються розробкою ПЗ, переважно веб-застосунків, реалізованих мовою Java.

Дане ПЗ розроблене з метою спрощення бізнес-процесів ІТ-компаній, які займаються розробкою веб-застосунків, реалізованих мовою Java.

Функціональним призначенням програмного забезпечення є автоматизація процесів:

– вводу статистичних даних;

– нормалізації вибірок за допомогою перетворення Джонсона;

– розрахунку параметрів для вхідних та нормалізованих вибірок;

– побудови лінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування для лінійного рівняння регресії;

– побудови нелінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування для нелінійного рівняння регресії;

– оцінювання розміру веб-застосунків, реалізованих мовою Java.

Для розробки ПЗ було обрано мову програмування Java та середовище розробки Eclipse.

При розробці додатку значну увагу було приділено інтерфейсу користувача. Важливим завданням було створити його якомога простим у використанні і в той самий час багатофункціональним і зручним.

Додаток Г Інструкція користувача

ПЗ для оцінювання розміру веб-застосунків, реалізованих мовою Java, заархівовано в архів з назвою JavaWebAnalytics.rar.

Після розпакування архіву заходимо в папку з установленним ПЗ та відкриваємо програму JavaWebAnalytics. Початкове вікно ПЗ приведено на рисунку Г.1.

Рисунок Г.1 – Початкове вікно ПЗ

Опис функцій та інформаційних блоків, наявних на початковому вікні ПЗ приведено на рисунку Г.2.

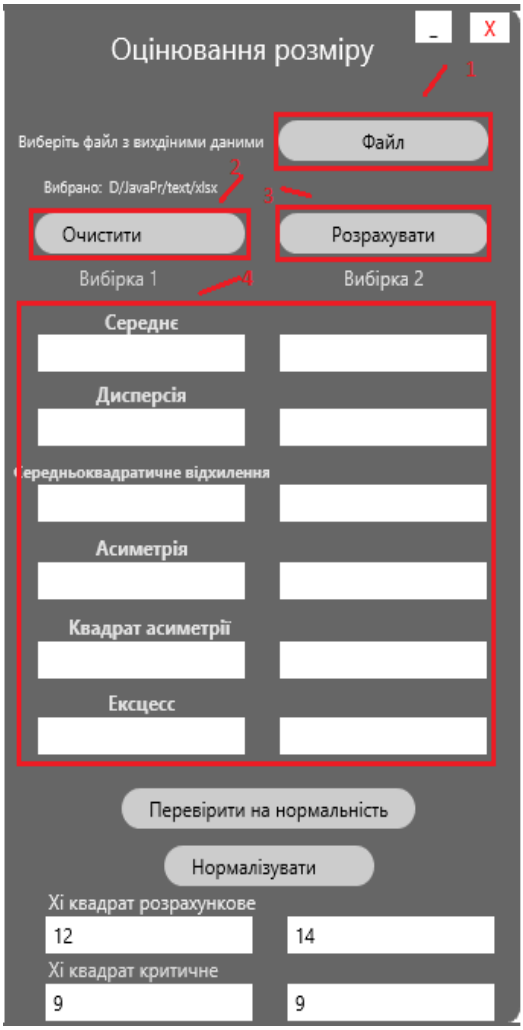

Рисунок Г.2 – Функції та блоки початкового вікна ПЗ

На рисунку Г.2 присутні такі Функції та блоки:

1. Кнопка для завантаження файлу.

2. Кнопка для видалення файлу.

3. Кнопка для розрахунку параметрів вихідних вибірок.

4. Блок параметрів вихідних вибірок.

Порядок дій для оцінювання розміру веб-застосунків, реалізованих мовою Java:

1. Натиснути кнопку «Файл» (1), відкриється вікно вибору файлу, вибирати потрібний файл.

2. Натиснути кнопку «Розрахувати» (3), отримаємо розраховані параметри у блоці 4 (див. рис. Г.2).

3. Після розрахунку параметрів з’явиться можливість перевірити

нормальність вибірок (5) та перейти до нормалізації (6) (див. рис. Г.3).

4. Натискаємо кнопку «Перевірити на нормальність» (5) та отримуємо розраховані параметри та повідомлення про нормальність у блоці нижче.

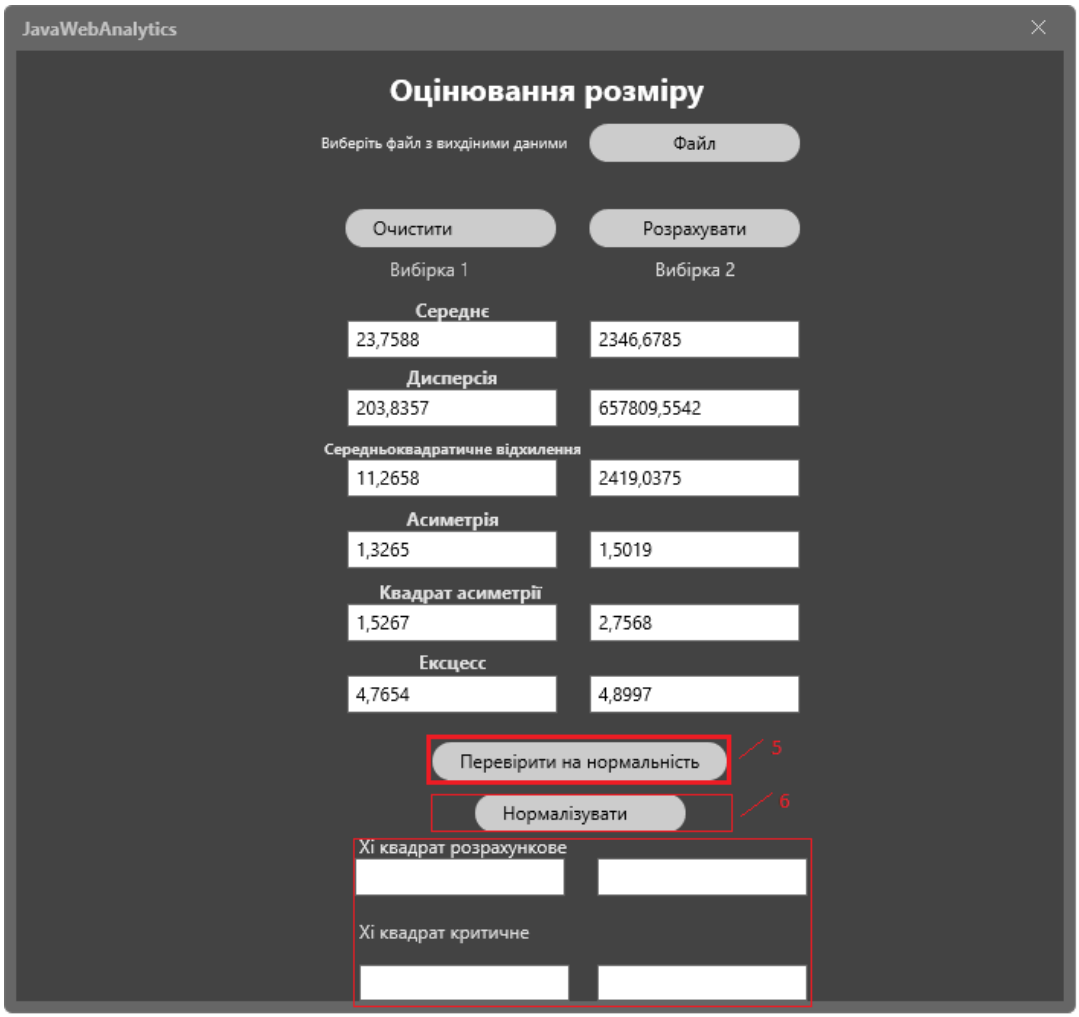

Рисунок Г.3 – Перевірка нормальності розподілу вхідних вибірок

5. Після перевірки на нормальність розподілу натискаємо «Нормалізувати» (6). Відкривається нове вікно програми (див. рис. Г.4).

6. Вводимо вручну параметри для початкового наближення в блок параметрів (7) та натискаємо кнопку «Розрахувати» (8).

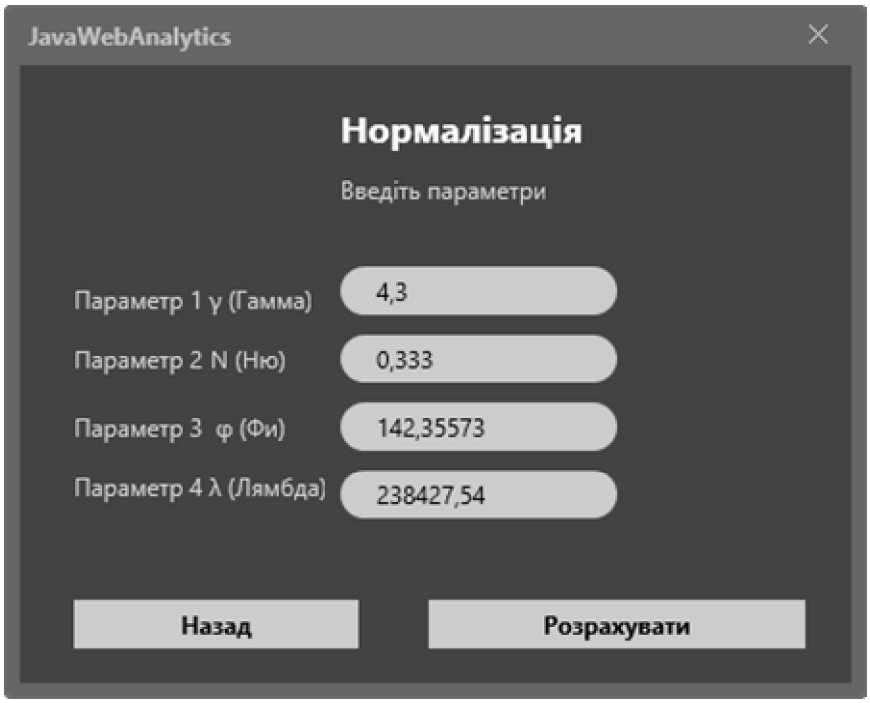

Рисунок Г.4 – Введення початкових параметрів перетворення Джонсона

7. Відкривається нове вікно (див. рис. Г.5) з розрахованими параметрами Джонсона для нормалізації (9). Перевіряємо нормалізовані вибірки на нормальність, натиснувши «Перевірити на нормальність» (10).

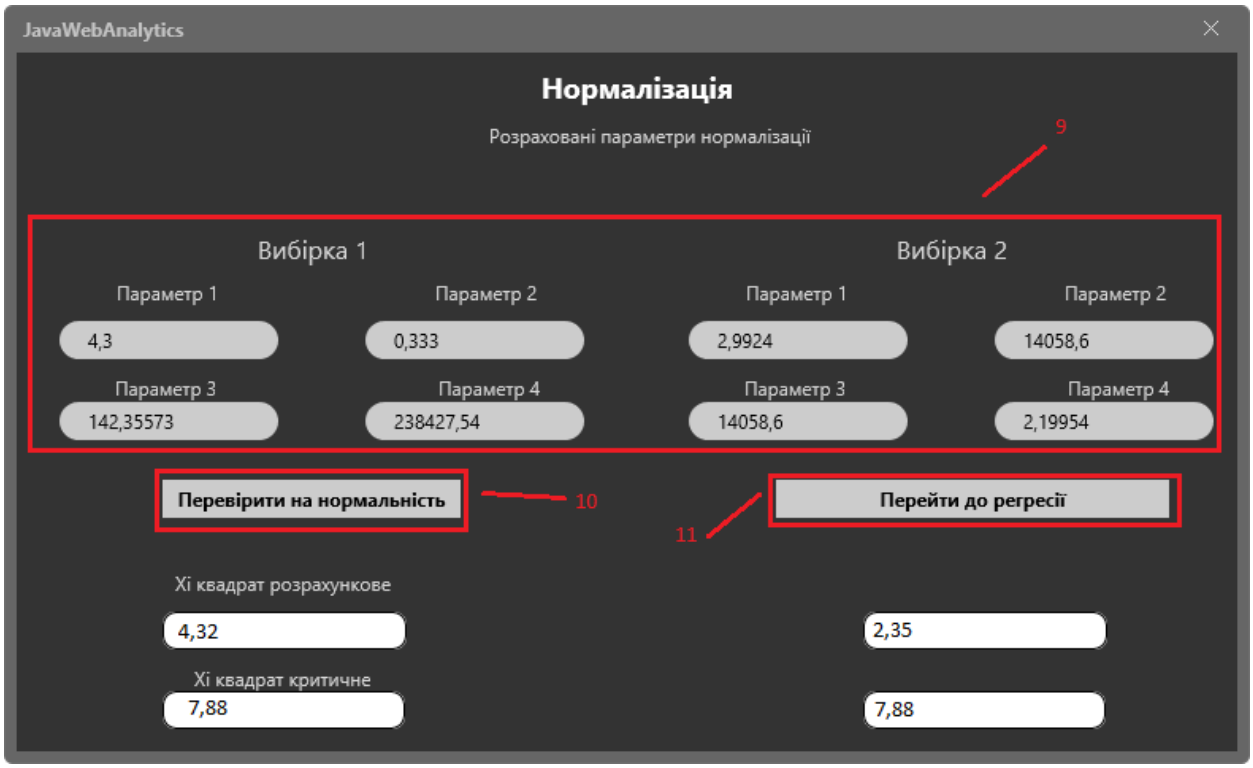

Рисунок Г.5 – Розраховані оптимальні параметри перетворення Джонсона

8. Натискаємо кнопку «Перейти до регресії» (11). Виводиться нове вікно «Рівняння регресії» з двома вкладками для лінійного та нелінійного рівняння (див. рис. Г.6 – Г.7).

Рисунок Г.6 – Лінійне рівняння регресії

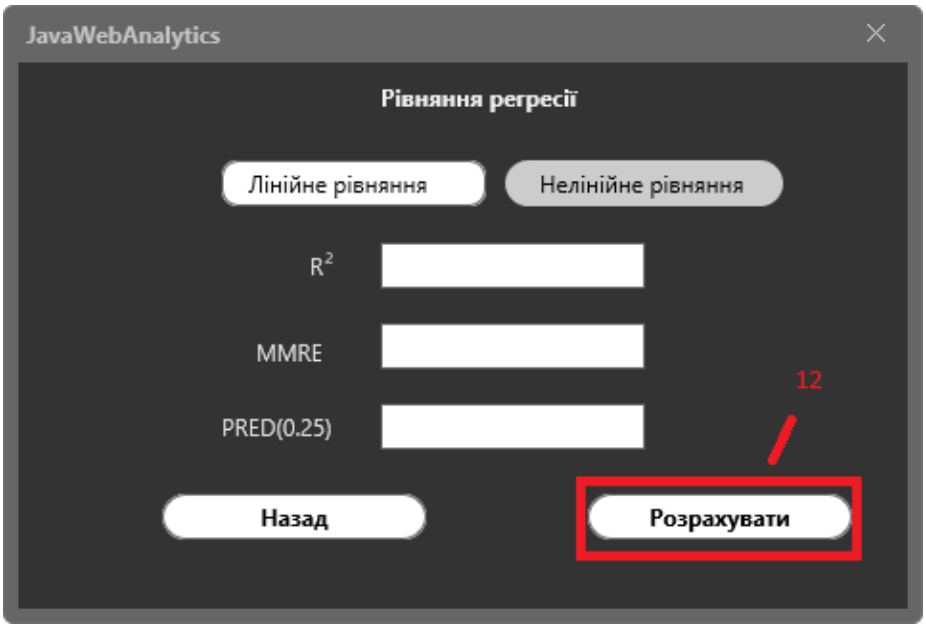

Рисунок Г.7 – Нелінійне рівняння регресії

9. Натискаємо кнопку «Розрахувати» (12) та отримуємо розраховані параметри для відповідного типу рівняння, побудований графік для відповідного рівнняня регресії, його довірчого інтервалу та інтервалу прогнозування (див. рис. Г.8).

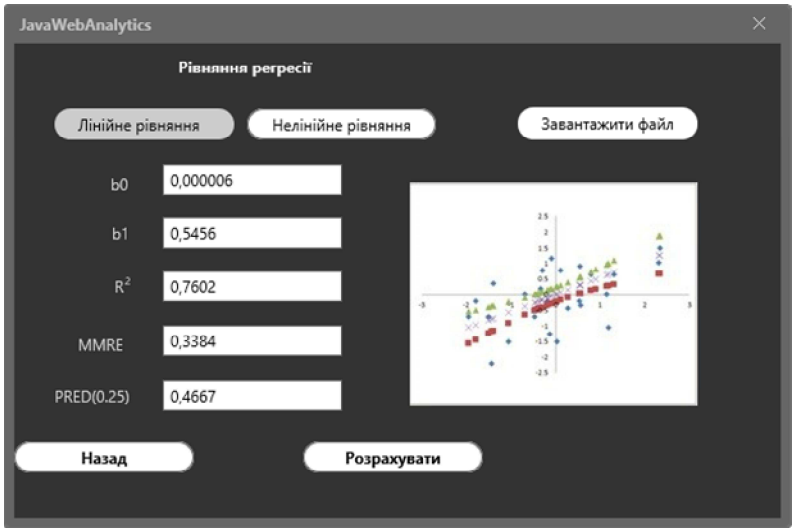

Рисунок Г.8 – Лінійне рівняння регресії, довірчий інтервал та інтервал прогнозування лінійного рівняння регресії

Натискаємо кнопку «Оцінювання» у вікні «Нелінійне рівняння». Виводиться нове вікно «Оцінювання розміру», яке зображене на рис. Г.9.

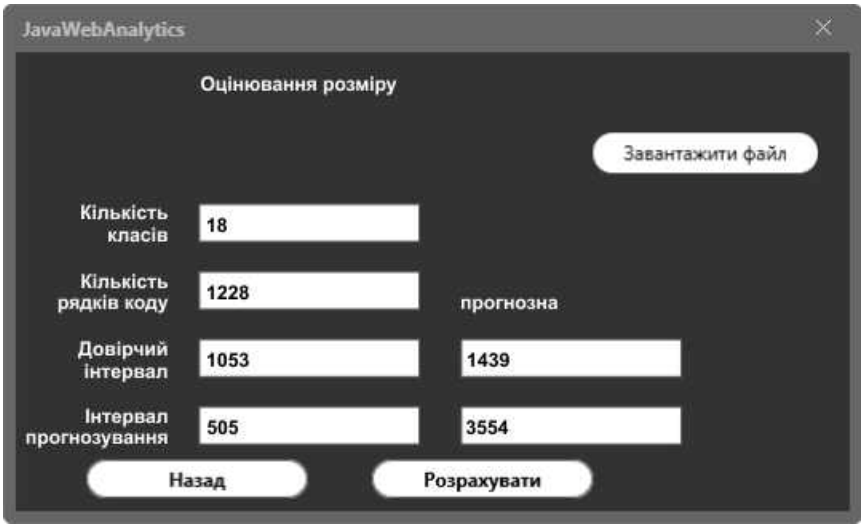

Рисунок Г.9 – Оцінювання розміру веб-застосунків, реалізованих мовою Java

10. Натиснувши кнопку «Завантажити файл», отримуємо файл у форматі txt з розрахованими параметрами.

Додаток Д Програма і методика випробувань програмного забезпечення

1 Об'єкт випробування

Об’єктом випробування є ПЗ для оцінювання розміру веб-застосунків, реалізованих мовою Java – JavaWebAnalytics.

2 Мета випробування

Метою проведення випробування є перевірка працездатності даного ПЗ, перевірка відповідності характеристик та вимог розробленого ПЗ, викладених у технічному завдані, приведення прикладу роботи та отримання відповідних навичок.

3 Вимоги до додатка

При проведенні випробувань функціональні характеристики (можливості) ПЗ підлягають перевірці на відповідність вимогам, викладеним у пункті «Вимоги до функціональних характеристик» технічного завдання.

4 Вимоги до програмної документації

4.1 Склад програмної документації, пропонованої на випробування

Програмна документація повинна включати в себе наступні документи:

1) Технічне завдання;

2) Текст програми;

3) Опис програми;

4) Інструкція користувача;

5) Програма та методика випробувань.

4.2 Спеціальні вимоги

Спеціальні вимоги до програмної документації не висуваються.

5 Засоби і порядок випробувань

5.1 Технічні засоби, що використовуються під час випробувань

Склад технічних засобів, що використовувалися під час випробувань:

– CPU: Intel Pentium 4405Y (1.2 ГГц);
– RAM: 1 GB;
– Жорсткий диск 70Гб RAID 1;
– Роздільна здатність екрану – 1920×1080 пікселів;
– Клавіатура 101/102-х клавішна рус / лат;
– Маніпулятор «миша».

5.2 Програмні засоби, що використовуються під час випробувань

ПЗ не пред'являє ніяких вимог до апаратної платформи.

Склад програмних засобів, що використовувалися під час випробувань:

Операційні системи:

- Windows 10.

5.3 Порядок проведення випробувань

Порядок проведення випробувань складається з перевірки вимог до функціональних характеристик ПЗ та перевірки вимог до програмної документації.

6 Методи випробувань

6.1 Методика проведення перевірки вимог до програмної документації

Перевірка дотримання вимог програмної документації на ПЗ виконується візуально представником служби, відповідальної за експлуатацію ПЗ. У ході перевірки зіставляється склад і комплектність програмної документації, представленої розробником, з переліком програмної документації, наведеним у пункті «Склад програмної документації, пропонованої на випробування» цього документа.

Перевірка вважається завершеною у випадку відповідності складу та комплектності програмної документації, представленої розробником, переліку програмної документації, наведеному у зазначеному вище пункті.

6.2 Методика проведення перевірки вимог до функціональних характеристик ПЗ

Перевірка працездатності ПЗ виконується згідно з пунктом «Вимоги до функціональних характеристик»технічного завдання.

Перевірка вважається завершеною у разі відповідності складу і послідовності дій, при виконанні даної перевірки, вказаному вище пункту технічного завдання.

В процесі тестування була перевірена функціональність програмного забезпечення для оцінювання розміру веб-застосунків, реалізованих мовою Java. У таблиці Д.1 наведено перелік випробувань основних функціональних можливостей ПЗ.

Таблиця Д.1 – Методика випробування

№	Дія	Очікуваний результат	Результат перевірки	Зауваження
1	2	3	4	5
1	Випробування на завантаження некоректного формату файлу	Виводиться повідомлення про помилку	Виконано	
2	Випробування на завантаження коректного формату файлу	У результаті файл успішно завантажився та нижче було виведено його назву	Виконано	
3	Випробування на розрахунок параметрів без завантаження файлу	У результаті отримали повідомлення про необхідність вибрати файл	Виконано	

Продовження табл. Д.1

1	2	3	4	5
4	Випробування на розрахунок параметрів після завантаження файлу	У відповідних текстових блоках з’явились розраховані параметри	Виконано	
5	Випробування на розрахунок параметрів Джонсона при некоректному вводі	У результаті з’явилось відповідне системне повідомлення, про необхідність ввести коректні параметри для розрахунку	Виконано	
6	Випробування на розрахунок параметрів Джонсона при коректному вводі	У відповідних текстових блоках з’явились розраховані параметри	Виконано	