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: «Математична модель для оцінювання трудомісткості розробки скраперів на Java та створення програмного забезпечення для її реалізації»

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

Цуман Д.С.
(підпис, прізвище та ініціали)
Керівник Латанська Л.О.
(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

Цуману Дмитру Станіславовичу
(прізвище, ім'я, по батькові)

1.Тема магістерської роботи: Математична модель для оцінювання трудомісткості розробки скраперів на Java та створення програмного забезпечення для її реалізації

керівник роботи Латанська Людмила Олексіївна, к.ф.-м. 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 Розрахунок економічної ефективності від розробки та впровадження програмного забезпечення
• Розділи з охорони праці та охорони навколишнього середовища Охорона праці, Охорона 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 _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Цуман Дмитро Станіславович

Математична модель для оцінювання трудомісткості розробки скраперів на Java та створення програмного забезпечення для її реалізації

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

Обсяг роботи: 102 стор., 7 таблиць, 19 рисунків, 19 використаних джерел, 5 додатків.

Актуальність теми. Застосування методів та моделей оцінювання трудомісткості розробки програмного забезпечення є одним з головних аргументів при техніко-економічному обгрунтуванні вартості розробки.

Але відомо, що не існує універсальних методів чи моделей, які є оптимальними для всіх типів програмного забезпечення і середовищ розробки. Тому існує необхідність удосконалювати існуючі та розробляти нові методи та моделі оцінювання трудомісткості розробки програмного забезпечення обраних типів, зокрема скраперів на Java. Це дозволить підвищити точність оцінювання, що робить задачу побудови математичної моделі для оцінювання трудомісткості розробки скраперів на Java актуальною.

Мета і завдання дослідження. Метою даної роботи є підвищення достовірності оцінювання трудомісткості розробки скраперів на Java. До завдань необхідно віднести: аналіз існуючих методів та моделей оцінювання трудомісткості розробки програмного забезпечення, удосконалення математичної моделі для оцінювання трудомісткості розробки скраперів на Java, розробку програмного забезпечення для її реалізації.

Об'єкт дослідження: процес оцінювання трудомісткості розробки скраперів на Java.

Предмет дослідження: нелінійна регресійна модель для оцінювання трудомісткості розробки скраперів на Java.

Методи дослідження. Для вирішення поставлених задач були застосовані методи теорії ймовірностей та математичної статистики, а також регресійного аналізу.

Наукова новизна одержаних результатів полягає в удосконаленні регресійної моделі для оцінювання трудомісткості розробки скраперів на Java з використанням нормалізуючого перетворення на основі натурального логарифму, що дозволило отримати більш точну оцінку.

Практичне значення одержаних результатів. Розроблені в рамках кваліфікаційної роботи модель, алгоритм та програма оцінювання трудомісткості розробки скраперів на Java дають можливість скоротити час оцінювання та підвищити його точність та достовірність.

Апробація результатів досліджень. Основні положення роботи були оприлюднені на Всеукраїнській науково-практичній інтернет конференції «ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ: МОДЕЛІ, АЛГОРИТМИ, СИСТЕМИ (ITMAS – 2020)» (м. Миколаїв, 26-28 жовтня 2020).

Ключові слова: МАТЕМАТИЧНА МОДЕЛЬ, ОЦІНЮВАННЯ ТРУДОМІСТКОСТІ, СКРАПЕРИ НА JAVA.

ABSTRACT

Tsuman Dmytro

Mathematical model for estimating the complexity of developing scrapers in Java and creating software for its implementation

Major's thesis for gaining the educational level of major in 121 – «Software engineering».Admiral Makarov National University of Shipbuilding. Mykolaiv, 2020.

The work contains: 102 pages, 7 tables, 19 figures, 19 references, 5 appendices.

Relevance of the topic: The use of methods and models for estimating the complexity of software development is one of the main arguments in the feasibility study of the cost of development.

But it is known that there are no universal methods or models that are optimal for all types of software and development environments. Therefore, there is a need to improve existing and develop new methods and models for estimating the complexity of software development of selected types, including scrapers in Java. This will increase the accuracy of the estimate, which makes the task of building a mathematical model for estimating the complexity of developing scrapers in Java relevant.

Purpose and tasks of the research: The purpose of this work is to increase the reliability of estimating the complexity of developing scrapers in Java. Tasks include: analysis of existing methods and models for estimating the complexity of software development, improving the mathematical model for estimating the complexity of developing scrapers in Java, software development for its implementation.

Object of research: the process of estimating the complexity of developing scrapers in Java.

Subject of research: nonlinear regression model for estimating the complexity of developing scrapers in Java.

Methods of research. Methods of probability theory and mathematical statistics, as well as regression analysis were used to solve the problems.

Scientific novelty of the obtained results is to improve the regression model for estimating the complexity of developing scrapers in Java using a normalizing transformation based on the natural logarithm, which allowed to obtain a more accurate estimate.

The practical value of the results. Developed in the master's thesis model, algorithm and program for estimating the complexity of developing scrapers in Java make it possible to reduce the evaluation time and increase its accuracy and reliability.

Approbation of study results: The main provisions of the work were announced at the All-Ukrainian scientific-practical Internet conference «INFORMATION TECHNOLOGIES: MODELS, ALGORITHMS, SYSTEMS (ITMAS - 2020)» (Mykolaiv, October 26-28, 2020).

Keywords: MATHEMATICAL MODEL, ESTIMATION OF HARD WORK, SCRAPERS IN JAVA.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасних реаліях розробка програмного забезпечення ведеться в умовах обмеженого часу та фінансових ресурсів. За статистикою більш ніж 37 відсотків усіх проектів є проблемними, із них 25 відсотків все ж таки закінчуються успіхом після значних доробок, а 12 відсотків проектів провалюються. Великий відсоток проектів закінчується невдачею через неправильно визначені часові рамки та недостатність виділених на розробку коштів [1]. Саме тому максимально точне оцінювання необхідних трудових витрат і необхідного часу є однією з основних проблем при управлінні проектами.

Існуючі моделі та методи для оцінювання трудомісткості використовують різні фактори (кількість функціональних точок, сторінок програмно-технічної та користувацької документації, досвід членів команди, які розроблюють програмний продукт та ін.). На сьогоднішній день найпопулярнішими моделями для оцінки трудомісткості є COCOMO і COCOMO II. В основі COCOMO лежить формула регресії з параметрами. Ці параметри визначають за допомогою даних отриманих з попередньо завершених проектів. Модель складається із ієрархії 3 послідовно доповнюваних і розширюваних форм, які враховують характеристики проекту, кваліфікацію персоналу та стан апаратного забезпечення. Недоліками цієї моделі є те, що результат використання цієї моделі дуже сильно залежить від правильності налаштування її до потреб саме даної організації, також оцінка кваліфікації персоналу, стану апаратного забезпечення є непростим завданням і потребує експертних оцінок, які частково мають суб’єктивний характер, також вона має недостатню точність передбачення [2].

Модель COCOMO II є модифікацією моделі COCOMO, створеної у 1991 році. В ній існують 2 основні етапи: попередня оцінка перед початком

розробки програмного продукту та більш детальна оцінка, яка проводиться після створення архітектури. Суттєвими відмінностями моделі COCOMO II від своєї попередниці є використання функціональних точок та об’єктно-орієнтовані підходи до оцінки. До недоліків цієї моделі можна віднести недостатню точність передбачення [3].

Як бачимо існуючі моделі оцінки трудомісткості розробки програмного забезпечення мають ряд недоліків і найбільший з них це недостатня точність. Саме тому виникає необхідність удосконалити регресійну модель оцінювання трудомісткості розробки скраперів на Java. Це дозволить підвищити точність оцінювання трудомісткості, що робить задачу актуальною.

Мета і завдання дослідження. Метою даної роботи є підвищення достовірності оцінювання трудомісткості розробки скраперів на Java. До завдань необхідно віднести: аналіз існуючих методів та моделей оцінювання трудомісткості розробки програмного забезпечення, удосконалення математичної моделі для оцінювання трудомісткості розробки скраперів на Java, розробку програмного забезпечення для її реалізації.

Об’єкт дослідження: процес оцінювання трудомісткості розробки скраперів на Java.

Предмет дослідження: нелінійна регресійна модель для оцінювання трудомісткості розробки скраперів на Java.

Методи дослідження. Для вирішення поставлених задач були застосовані методи теорії ймовірностей та математичної статистики, а також регресійного аналізу.

Наукова новизна одержаних результатів полягає в удосконаленні регресійної моделі для оцінювання трудомісткості розробки скраперів на Java з використанням нормалізуючого перетворення на основі десяткового логарифму, що дозволило отримати більш точну оцінку трудомісткості розробки скраперів на Java.

Практичне значення одержаних результатів. Розроблені в рамках кваліфікаційної роботи модель, алгоритм та програма оцінювання трудомісткості розробки скраперів на Java дають можливість скоротити час оцінювання та підвищити його точність та достовірність.

Особистий внесок здобувача. Кваліфікаційна робота є самостійно виконаною працею.

Апробація результатів досліджень. Основні положення роботи були оприлюднені на Всеукраїнській науково-практичній інтернет конференції «ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ: МОДЕЛІ, АЛГОРИТМИ, СИСТЕМИ (ITMAS – 2020)» (м. Миколаїв, 26-28 жовтня 2020).

1 ПОНЯТТЯ СКРАПІНГУ ТА СКРАПЕРІВ НА JAVA ТА АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО ОЦІНЮВАННЯ ТРУДОМІСТКОСТІ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Поняття скрапінгу та скраперів на Java

Скрапінг (scrapping) – процес отримання даних зі сторінок веб-сайтів за допомогою автоматизації звернень та підготовка цих даних для подальшої обробки. З юридичної точки зору процес скрапінга не може вважатися незаконним, оскільки інформація, що отримується, є доступною всім [4].

Скрапер – програма, яка шукає в інтернеті інформацію по заданим параметрах, іноді вона ж автоматично публікує інформацію на іншому сайті, використовується для наповнення сайтів контентом.

Історія скрапінгу починається з тих часів, коли з’явився інтернет. Однак по мірі росту кількості інформації в інтернеті зростає і важливість скрапінгу.

Процес веб-скрапінга є ключовим моментом сучасних веб-технологій. Він складається з двох основних етапів:

– Витяг коду веб-сторінок з інтернету.

– Витяг необхідної інформації з коду веб-сторінок.

Існує маса готових рішень для скрапінга веб-сайтів. Серед них:

– Окремі сервіси.

– Проекти з відкритим кодом на різних мовах програмування.

– Збереження структурованих даних в таблицях або базах даних [4].

Веб-скрапери використовуються для автоматизації наступних задач:

– Копіювання даних з інтернету.

– Пошуку необхідної інформації.

– Моніторингу оновлень інформації на сайтах[4].

Однак скрапери розробляються програмістами і в наш час, тому що ідеального скрапера не існує. Це пояснюється тим, що:

– Жоден сайт не має ідеальної верстки з точки зору вимог веб-дизайну.

– Кожен веб-розробник пише код під себе або просто, як вміє. Далеко не завжди код виходить грамотним і якісним. Все це робить код абсолютно нечитабельним для скраперів.

– Деякі ресурси містять різноманітні захисти від копіювання даних, а значить і від скрапінга.

– В залежності від сезону або тематики цільового матеріалу на сайті можуть бути використані різні макети.

– У всіх сайтів може бути різне кодування та інші.

Все це ускладнює веб-скрапінг. В результаті якість контенту може значно впасти, що абсолютно неприйнятно, так як повнота інформації є її найважливішим критерієм.

Існує думка, що:

– При необхідності отримувати дані з невеликої кількості джерел, краще написати свій скрапер і налаштувати його під потрібні сайти.

– Якщо потрібно отримувати інформацію з великої кількості джерел, то використовувати комплексний підхід у виборі скрапера.

Деякі адміністратори веб-сайтів протидіють веб-скраперам. Активність веб-скраперів може негативно позначитися на продуктивності веб-сервера чи надати перевагу над конкурентам. В свою чергу розробники скраперів, у відповідь на блокування, намагаються його обійти.

1.2 Аналіз сучасних підходів до оцінювання трудомісткості розробки програмного забезпечення

При оцінюванні трудомісткості розробки програмного забезпечення часто вирішують наступні проблеми [5]:

– яку метрику розміру програмного забезпечення використовувати?

– які метод чи модель оцінювання трудомісткості вибрати?

– яку оцінку можна вважати оптимальною?

1.2.1 Метрики розміру програмного забезпечення

Більшість моделей оцінювання трудомісткості базуються на використанні різних метрик розміру розроблюваного програмного забезпечення. Точність оцінки трудомісткості напряму залежить від точності оцінки розміру. Розглянемо основні метрики розміру програмного забезпечення, що використовуються при оцінюванні трудомісткості [5]:

– кількість рядків коду (кількість непустих рядків вихідного тексту без коментарів);

– метрики Холстеда (довжина коду та об’єм);

– функціональні точки (оцінювання об’єму функціональності за рахунок вивчення вимог);

– точки властивостей (враховують не тільки вимоги до системи, але і особливості її реалізації);

– об’єктні точки (вікна, звіти, компоненти та ін.).

Найпоширенішою метрикою вихідного коду, що відображає розмір програмного проекту, є показник кількості рядків коду. Ця метрика має багато похідних метрик (кількість пустих рядків; кількість рядків, що містять коментарі і т.д.), що дозволяють оцінити різні показники проекту. В даній роботі для оцінювання трудомісткості розробки програмного забезпечення буде використана метрика кількості рядків коду. Як правило, кількість рядків коду програми є негаусівською випадковою величиною.

1.2.2 Основні методи та моделі оцінювання трудомісткості розробки програмного забезпечення

На сьогодні існує велика кількість підходів до оцінювання трудомісткості робіт. Кожен з них базується на певних методах чи моделях оцінки трудомісткості, деякі з яких універсальні, інші розроблені під конкретні проекти. В залежності від способу отримання даних для визначення трудомісткості, розглянемо наступні основні методи та моделі [2,3, 6-8]:

– *Метод експертних оцінок.* Метод ґрунтується на опитуванні декількох експертів по технології розробки. Кожен з них дає свою оцінку трудомісткості проекту, після чого всі оцінки фіксуються та обговорюються до тих пір, поки не буде сформована єдина інтегрована консенсусна оцінка. Метод застосовується в проектах, які використовують нові технології, нові процеси або вирішують інноваційні задачі.

– *Метод оцінки за аналогією.* Метод грунтується на порівнянні запланованого проекту з попередніми проектами, які мають подібні характеристики. Метод використовується в тому випадку, коли в даній області застосування створюваного програмного проекту вже реалізовані аналогічні проекти.

– *Метод алгоритмічного моделювання.* Метод базуються на аналізі статистичних даних про раніше виконані проекти, при цьому виділяється залежність трудомісткості проекту від будь-якого кількісного показника програмного продукту, зазвичай це розмір програмного коду. Проводиться оцінювання цього показника для даного проекту, після чого за допомогою моделі прогнозуються майбутні витрати.

– *Модель Путнема* (SLIM (Software Life-cycle Model)). Це нелінійна модель, яка запропонована в 1978 р Л. Патнамом. Грунтується на твердженні, що витрати на розробку програмного забезпечення розподіляються згідно кривим Нордена-Рейлі, які є графіками функції, що

представляє розподіл робочої сили за часом. Дана модель найбільш пристосована для оцінки великих проектів, так як її базу складають реальні дані Міністерства оборони США.

– *Метод функціональних точок.* Метод розроблений Аланом Альбрехтом (Alan Albrecht) в середині 70-х років та призначений для оцінки трудомісткості на основі логічної моделі обсягу програмного продукту, виміряного кількістю функціоналу, який затребуваний замовником та постачається розробником.

– *Модель COCOMO* (Constructive COst MOdel). Модель розроблена Б. Боемом. Вона дозволяє оцінити трудомісткість розробки програмного продукту. Вперше була опублікована в 1981 році у вигляді результату аналізу 63 проектів компанії «TRW Aerospace». Модель COCOMO є ієрархією з трьох рівнів: базового, середнього та детального. У 1997 модель була вдосконалена і отримала назву COCOMO II. Калібрування параметрів проводилося по 161 проекту розробки. У моделі використовується формула регресії з параметрами, визначеними на основі галузевих даних і характеристик конкретного проекту.

– *Нейронні мережі.* Це методи оцінки з можливістю навчання по аналогії з допомогою техніки штучного інтелекту, інструмент пошуку закономірностей, прогнозування, якісного аналізу.

– *Методи імітаційного моделювання*. В цих методах система, що досліджується, замінюється моделлю, яка з достатньою точністю описує реальну систему. З моделлю проводяться експерименти з метою отримання інформації про систему.

– *Динамічні методи.* В методах використовують припущення про те, що показники, які впливають на вартість і тривалість проекту (наприклад, досвід розробників, вихідні вимоги, необхідність навчання і ін.), змінюються протягом розробки.

– *Композитні методи.* Такі методи являються комбінацією двох і більше методів для формування найбільш підходящої системи оцінки.

Як випливає з аналізу сучасних підходів до розрахунку трудомісткості розробки, на даний момент існує велика кількість методів та моделей для її оцінювання. Використання розробниками цих методів та моделей є одним з головних аргументів при техніко-економічному обгрунтуванні вартості розробки. Але частина з цих підходів вже застаріла та їх використання дає неадекватний результат, який приводить до несприятливих наслідків. Окрім того, прогнози щодо точності оцінювання трудомісткості розробки програмного забезпечення існуючими методами та моделями, зазвичай, правдиві лише для певного типу проектів, аналогічних тим, на яких ті чи інші моделі чи методи були випробувані. Тобто, не існує універсальних методів чи моделей, які є оптимальними для всіх проектів і середовищ розробки. Також в більшості підходів враховують лише лінійні зв’язки між факторами. Тому існує необхідність удосконалювати існуючі та розробляти нові методи та моделі оцінки трудомісткості розробки програмного забезпечення, в тому числі і скраперів на Java, які будуть враховувати і нелінійність зв’язків між факторами.

1.2.3 Критерії оптимальної оцінки

Оптимальна оцінка повинна бути [5]:

– зрозумілою та підтриманою менеджером проекту та групою розробників;

– базуватися на чіткій моделі, що заслуговує довіри;

– базуватися на даних подібного проекту;

– детально визначеною, зі зрозумілими областями ризику та з об’єктивною оцінкою ймовірності успіху.

1.2.4 Перевірка якості математичної моделі для оцінювання трудомісткості розробки програмного забезпечення

Для розроблених чи вдосконалених методів та моделей обов'язковою є перевірка їх якості [5, 9].

Для перевірки якості розробленої математичної моделі використаємо наступні показники: коефіцієнт детермінації R^2, середню величину відносної похибки $MMRE$, рівень прогнозування $PRED(l)$ [9, 10, 11]:

– Коефіцієнт детермінації R^2 оцінює долю дисперсії залежної змінної, яка пояснюється з допомогою незалежної змінної в лінійній регресійній моделі. R^2 обчислюється за формулою:

$$R^2 = 1 - (\sum_1^n (y_i - \hat{y}_i)^2 / \sum_1^n (y_i - \bar{y})^2), \quad (1.1)$$

де y_i – емпіричне значення;

$\hat{y}_i$ – розраховане значення;

$\bar{y}$ – середнє значення;

n – кількість емпіричних даних.

Коефіцієнт детермінації приймає значення від 0 (відсутній лінійний зв'язок між показниками) до 1 (відсутній кореляційний зв'язок між показниками). Моделі з $R^2 \geq 0{,}5$ вважаються допустимими, а з $R^2 \geq 0{,}8$ – достатньо ефективними.

– Середня величина відносної похибки *MMRE* визначається за формулою:

$$MMRE = \frac{1}{n} \cdot \sum_1^n \left| \frac{y_i - \hat{y}_i}{y_i} \right| . \quad (1.2)$$

Прийнятне значенням $MMRE \leq 0{,}25$.

– Рівень прогнозування $PRED(l)$ обчислюється за формулою:

$$PRED(l) = \frac{k}{n}, \tag{1.3}$$

де k – кількість значень $\left|\frac{y_i - \hat{y}_i}{y_i}\right| \leq l$ $(i = \overline{1,n})$.

Зазвичай $l = 0{,}25$. Задовільне значенням $PRED(0{,}25) \geq 0{,}75$.

Із вищевикладеного випливає, що на сьогодні існує велика кількість методів та моделей оцінювання трудомісткості розробки програмного забезпечення. Їх використання є одним з головних аргументів при техніко-економічному обгрунтуванні вартості розробки.

Але частина з цих підходів має низьку точність оцінювання, не враховує нелінійність зв'язків між факторами, призначена для даних, що підпорядковуються нормальному закону розподілу. Також відомо, що не існує універсальних методів чи моделей, які є оптимальними для всіх типів програмного забезпечення і середовищ розробки. Тому існує необхідність удосконалювати існуючі та розробляти нові методи та моделі оцінювання трудомісткості розробки програмного забезпечення обраних типів, зокрема скраперів на Java. Це дозволить підвищити точність оцінювання трудомісткості розробки скраперів на Java з урахуванням нелінійних зв'язків між даними, що не підпорядковуються нормальному закону розподілу.

2 УДОСКОНАЛЕННЯ МАТЕМАТИЧНОЇ МОДЕЛІ ДЛЯ ОЦІНЮВАННЯ ТРУДОМІСТКОСТІ РОЗРОБКИ СКРАПЕРІВ НА JAVA

При побудові удосконаленої математичної моделі оцінювання трудомісткості розробки скраперів на Java будуть використовуватися наступні дані:

- кількість рядків коду (*X*);
- трудомісткість розробки веб-скраперів на Java (*Y*).

Так як зв'язки між даними моделі не завжди є лінійними і дані не підпорядковуються нормальному закону розподілу, то для застосування лінійної теорії для гаусівських даних нелінійну модель для негаусівських даних за допомогою нормалізуючих перетворень змінних зведемо до лінійної моделі для перетворених змінних. За допомогою методу найменших квадратів визначимо коефіцієнти лінійного співвідношення, по відомим формулам знайдемо довірчий інтервал та інтервал прогнозування. Потім за допомогою зворотнього перетворення знайдемо параметри вихідної

В якості нормалізуючого перетворення було обрано натуральний логарифм. Із аналізу методів та моделей для оцінювання трудомісткості розробки програмного забезпечення можна зробити висновок про те, що регресійна модель, яка розробляється, дасть змогу скоротити витрати праці та спростити визначення трудомісткості [12,13].

2.1 Математична модель оцінювання трудомісткості розробки скраперів на Java

При побудові удосконаленої математичної моделі оцінювання трудомісткості розробки скраперів на Java використовувалися дані:

- кількість рядків коду (*X*);

– трудомісткість розробки веб-скраперів на Java (*Y*).

За допомогою χ^2 критичного визначили, що розподіл *X,Y* не є гаусівським. Тому необхідно провести нормалізацію вихідних даних [14].

В якості нормалізуючого перетворення обираємо натуральний логарифм. Отримуємо нормалізовані дані Z_Y, Zx. Нормалізовані дані перевіряли на викиди, використовуючи квадрат відстані Махаланобіса [15] :

$$d_i^2 = (Z_i - \overline{Z})^T S_N^{-1} (Z_i - \overline{Z}), \tag{2.1}$$

де Z_i – i - ий елемент;

N – кількість елементів;

$\overline{Z}$ – вектор з середніх значень по нормалізованим Z_Y, Zx;

S_N – коваріаційна матриця:

$$S_N = \frac{1}{N} \sum_{i=1}^{N} (Z - \overline{Z})\,(Z - \overline{Z})^T. \tag{2.2}$$

Для знаходження викидів з допомогою критерія Фішера для d_i^2 розраховуються статистики

$$T_{S_i} = (N - k) N d_i^2 / (N^2 - 1) k, \tag{2.3}$$

де N – всього елементів,

k – кількість параметрів.

Ті нормалізовані дані, для яких T_{S_i}більше критичного значення Фішера $F_{k,N-k,\alpha}$, являються викидами. Їх необхідно вилучити з вихідних даних та повторити розрахунок.

Емпіричні та нормалізовані дані без викидів наведені в таблиці 2.1

Таблиця 2.1 – Емпіричні та нормалізовані дані

№	*Y* люд/год	*X* кількість строк коду	Z_Y нормаліз. *Y*	*Zx* нормаліз. *X*
1	2	3	4	5
1	20,20	168,00	3,01	5,12
2	23,80	232,00	3,17	5,45
3	17,20	120,00	2,84	4,79
4	18,60	97,00	2,92	4,57
5	23,10	168,00	3,14	5,12
6	19,00	118,00	2,94	4,77
7	19,00	149,00	2,94	5,00
8	25,80	285,00	3,25	5,65
9	23,00	211,00	3,14	5,35
10	19,80	169,00	2,99	5,13
11	17,20	146,00	2,84	4,98
12	23,40	171,00	3,15	5,14
13	17,20	172,00	2,84	5,15
14	17,00	146,00	2,83	4,98
15	23,00	170,00	3,14	5,14
16	14,40	111,00	2,67	4,71
17	25,20	199,00	3,23	5,29
18	23,20	189,00	3,14	5,24
19	17,80	119,00	2,88	4,78
20	20,60	174,00	3,03	5,16
21	14,00	120,00	2,64	4,79
22	18,40	141,00	2,91	4,95
23	18,40	143,00	2,91	4,96
24	19,80	205,00	2,99	5,32
25	26,80	275,00	3,29	5,62
26	17,60	141,00	2,87	4,95
27	19,20	124,00	2,95	4,82
28	23,20	159,00	3,14	5,07
29	19,20	173,00	2,95	5,15
30	30,20	277,00	3,41	5,62

На рисунках 2.1.а та 2.1.б представлені відповідно вихідний та нормалізований розподіли Y.

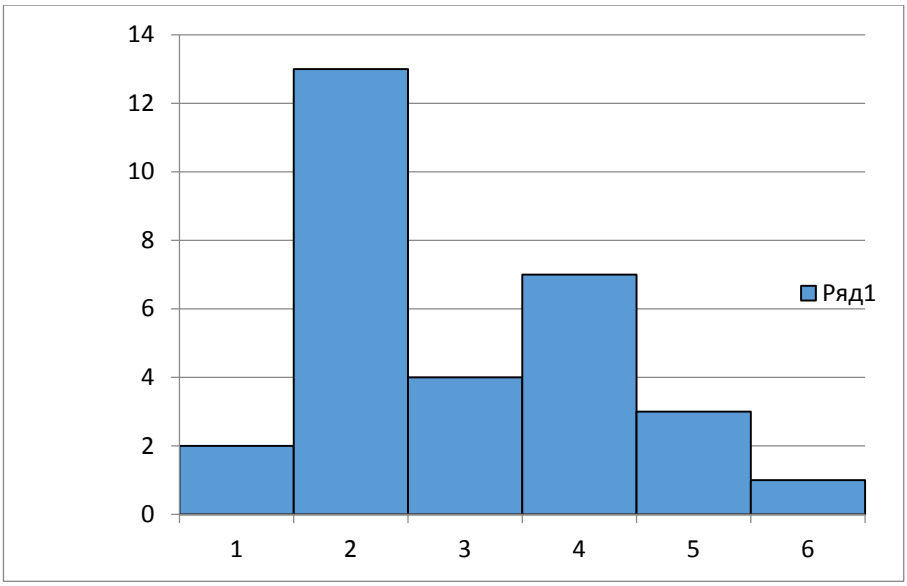

Рисунок 2.1.а – Емпіричний розподіл для вибірки Y

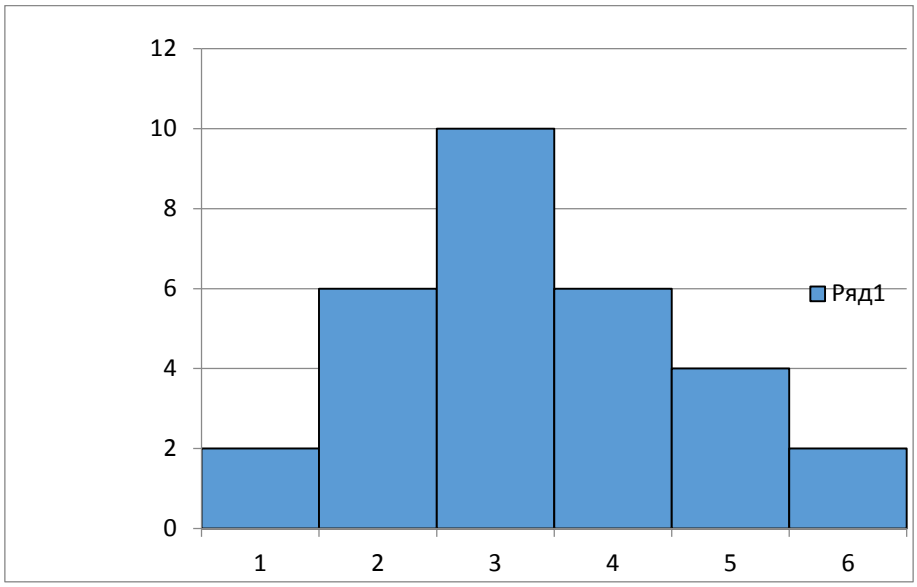

Рисунок 2.1.б – Нормалізований розподіл для вибірки Y

Емпіричний розподіл для вибірки X представлено на рис. 2.2.а, а її нормалізований розподіл представлений на рис. 2.2.б.

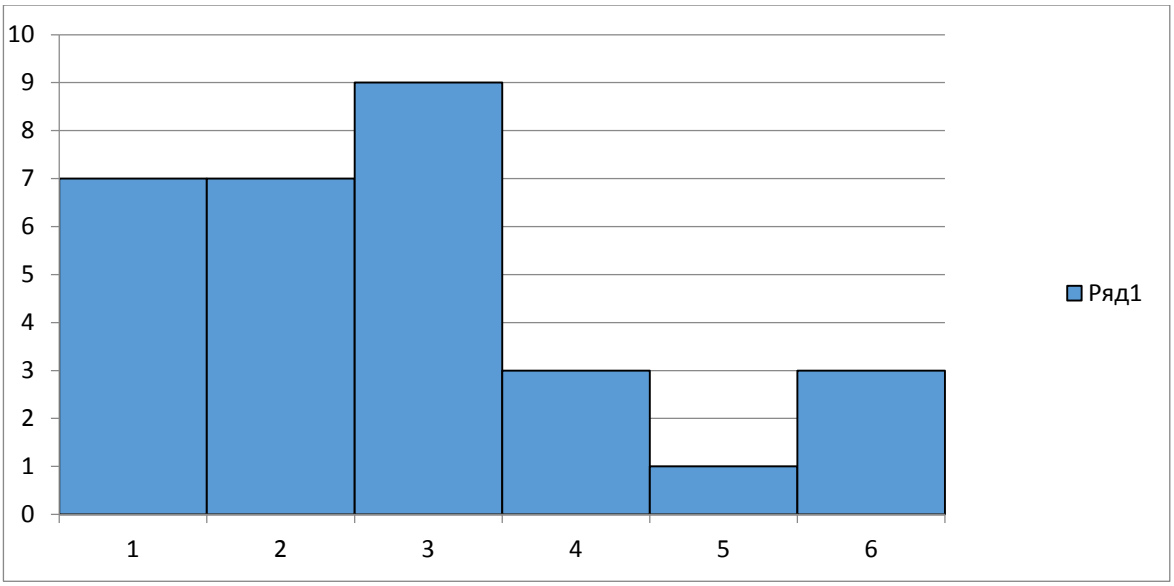

Рисунок 2.2.а – Емпіричний розподіл для вибірки X

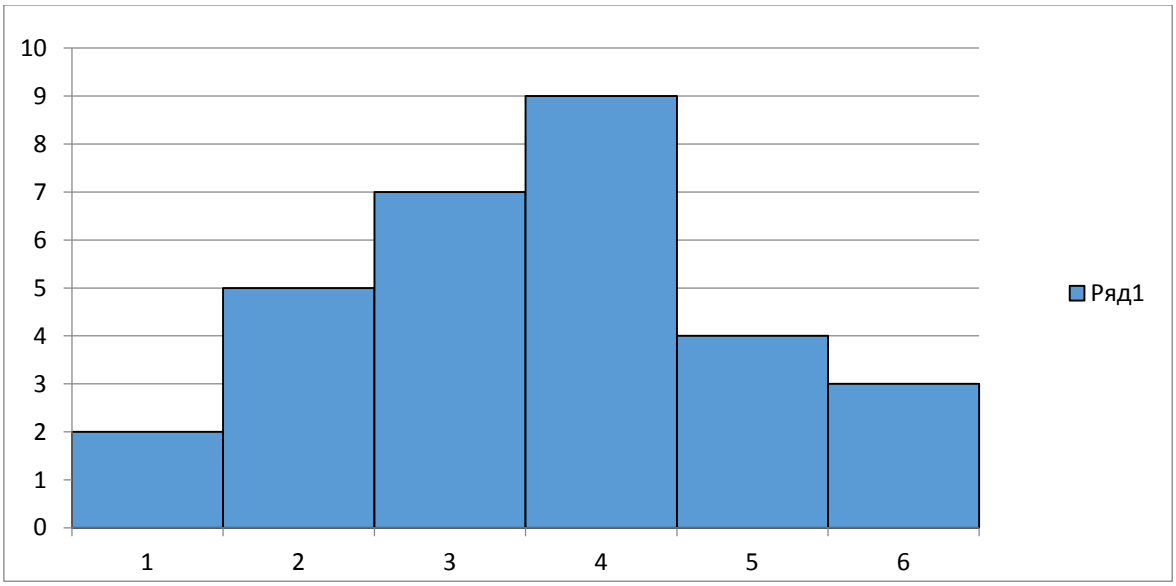

Рисунок 2.2.б – Нормалізований розподіл для вибірки X

Лінійна регресійна модель для нормалізованих даних Z_x, Z_y має вигляд [14]:

$$\hat{Z}_y = \mathrm{b}_0 + \mathrm{b}_1 \cdot Z_x + \varepsilon\,, \tag{2.4}$$

де b_0, b_1 – коефіцієнти лінійної регресії;

ε – випадкова помилка.

Значення коефіцієнтів були розраховані за методом найменших квадратів: b_0 = -3,8729, b_1 = 0,5553.

Отримали рівняння:

$$\hat{Z}_y = - 3{,}8729 + 0{,}5553Z_x + \varepsilon.$$

Для нормалізованих даних після побудови регресійного рівняння будуємо довірчий інтервал. Довірчий інтервал (confidence interval) – інтервал, через який вихідна функція проходить з певною ймовірністю (зазвичай 95%). Нижня та верхня межі довірчого інтервалу для лінійної регресії обчислюються за формулою [16]:

$$\hat{Z}_y \pm t_{\alpha/2,n-2} \cdot S_{Z_y} \sqrt{\frac{1}{n} + \frac{(Z_x - \overline{Z_x})^2}{\sum_n^1 (z_{xi} - \overline{Z_x})^2}}, \quad (2.5)$$

де $\hat{Z}_y$ – значення, розраховані за рівнянням регресії;

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

α – рівень статистичної значущості;

n – кількість елементів в вибірці;

$$S_{Z_y} = \sqrt{\frac{1}{n-2} \cdot \sum_1^n \left(z_{yi} - \hat{z}_{y_i}\right)^2}.$$

Інтервал передбачення (prediction interval) – інтервал, в який потраплять нові точки із заданою ймовірністю (зазвичай 95%). Нижня та верхня межі інтервалу передбачення для лінійної регресії представлені формулою (2.6):

$$\hat{Z}_y \pm t_{\alpha/2,n-2} \cdot S_{Z_y} \sqrt{1 + \frac{1}{n} + \frac{(Z_x - \overline{Z_x})^2}{\sum_n^1 (z_{x_i} - \overline{Z_x})^2}}. \quad (2.6)$$

Шляхом використання зворотнього перетворення отримуємо нелінійну регресійну модель оцінювання розміру програмного забезпечення [17]:

$$\hat{Y} = e^{b_0+\varepsilon} \cdot X^{b_1} . \quad (2.7)$$

Після підстановки

$$\hat{Z}_y = 0{,}3072 + 0{,}5298 Z_x + \varepsilon$$

$\hat{Y}$ набуде вигляду:

$$\hat{Y} = e^{0{,}3072+\varepsilon} * X^{0{,}5298}.$$

Через зворотнє перетворення отримуємо нижню та верхню межі довірчого інтервалу та інтервалу передбачення.

Також в роботі була побудована лінійна модель для вихідних даних у припущенні про нормальність їх розподілу. Вона має вигляд

$$\hat{Y} = 9{,}7343 + 0{,}0637X + \varepsilon.$$

Для цієї моделі також були побудовані довірчий інтервал та інтервал прогнозування.

Функція, довірчий інтервал та інтервал прогнозування для нелінійної моделі з нормалізацією представлені в таблиці 2.2

Таблиця 2.2 – Результати дослідження нелінійної моделі

№ п / п	Функція	Ліва границя довірчого інтервалу	Права границя довірчого інтервалу	Ліва границя інтервалу прогнозув.	Права границя інтервалу прогнозув.
1	2	3	4	5	6
1	20,5288	19,6146	21,4855	15,9554	26,4130
2	24,3571	22,6016	26,2491	18,8009	31,5554
3	17,1770	16,0389	18,3958	13,2817	22,2147
4	15,3458	13,9082	16,9321	11,7536	20,0359
5	20,5288	19,6146	21,4855	15,9554	26,4130
6	17,0247	15,8626	18,2720	13,1563	22,0306
7	19,2641	18,3670	20,2050	14,9666	24,7957
8	27,1622	24,4664	30,1552	20,7555	35,5466
9	23,1631	21,7526	24,6651	17,9366	29,9125
10	20,5934	19,6739	21,5559	16,0053	26,4968
11	19,0577	18,1486	20,0122	14,8028	24,5355
12	20,7222	19,7908	21,6974	16,1044	26,6640
13	20,7863	19,8483	21,7685	16,1537	26,7474
14	19,0577	18,1486	20,0122	14,8028	24,5355
15	20,6579	19,7326	21,6265	16,0550	26,5804
16	16,4820	15,2316	17,8350	12,7068	21,3788
17	22,4556	21,2234	23,7593	17,4148	28,9556
18	21,8505	20,7492	23,0103	16,9626	28,1470
19	17,1010	15,9510	18,3340	13,2192	22,1228
20	20,9140	19,9618	21,9116	16,2517	26,9138
21	17,1770	16,0389	18,3958	13,2817	22,2147
22	18,7091	17,7721	19,6954	14,5248	24,0986
23	18,8492	17,9245	19,8215	14,6368	24,2739
24	22,8118	21,4928	24,2117	17,6784	29,4358
25	26,6531	24,1375	29,4309	20,4082	34,8089
26	18,7091	17,7721	19,6954	14,5248	24,0986
27	17,4780	16,3861	18,6426	13,5286	22,5803
28	19,9386	19,0529	20,8655	15,4970	25,6532
29	20,8502	19,9053	21,8399	16,2028	26,8306
30	26,7556	24,2040	29,5762	20,4784	34,9570

Отримані результати для лінійної моделі без нормалізації вихідних даних (функція, довірчий інтервал та інтервал передбачення) представлені в таблиці 2.3

Таблиця 2.3 – Результати дослідження лінійної моделі в припущенні про нормальність вихідних даних

№ п / п	Функція	Ліва границя довірчого інтервалу	Права границя довірчого інтервалу	Ліва границя інтервалу прогнозув.	Права границя інтервалу прогнозув.
1	2	3	4	5	6
1	20,20	19,5891	21,2950	18,7729	22,1112
2	23,80	23,1144	25,9279	22,7972	26,2451
3	17,20	16,1628	18,6025	15,6800	19,0854
4	18,60	14,3777	17,4558	14,1760	17,6574
5	23,10	19,5891	21,2950	18,7729	22,1112
6	19,00	16,0096	18,5007	15,5497	18,9606
7	19,00	18,3067	20,1554	17,5562	20,9058
8	25,80	25,6687	30,1297	26,0505	29,7479
9	23,00	22,0500	24,3153	21,4890	24,8764
10	19,80	19,6530	21,3585	18,8366	22,1749
11	17,20	18,0936	19,9860	17,3632	20,7164
12	23,40	19,7798	21,4867	18,9640	22,3024
13	17,20	19,8426	21,5513	19,0277	22,3662
14	17,00	18,0936	19,9860	17,3632	20,7164
15	23,00	19,7166	21,4224	18,9003	22,2386
16	14,40	15,4701	18,1480	15,0931	18,5250
17	25,20	22,0546	22,7811	20,7361	24,0995
18	23,20	21,4510	22,1099	20,1058	23,4552
19	17,80	16,8632	17,7746	15,6149	19,0230
20	20,60	20,5227	21,1261	19,1550	22,4939
21	14,00	16,9321	17,8332	15,6800	19,0854
22	18,40	18,3649	19,0773	17,0409	20,4013
23	18,40	18,4996	19,1976	17,1699	20,5273
24	19,80	22,4123	23,1883	21,1130	24,4875
25	26,80	26,4762	28,0475	25,4415	29,0821
26	17,60	18,3649	19,0773	17,0409	20,4013
27	19,20	17,2071	18,0681	15,9401	19,3351
28	23,20	19,5608	20,1760	18,1978	21,5389
29	19,20	20,4597	21,0617	19,0913	22,4301
30	30,20	26,5909	28,1876	25,5635	29,2151

Для оцінювання якості побудованих моделей, для них були обчислені R^2, *MMRE* та *PRED*(25) [18]. Отримали: для нелінійної моделі коефіцієнт детермінації R^2 = 0,6977, а для лінійної – R^2 = 0,7035; для нелінійної моделі середня величина відносної похибки MMRE = 0,0828, а для лінійної – MMRE

= 0,0829; для нелінійної моделі рівень прогнозування PRED(25) = 1 і для лінійної – R^2 = 1. Таким чином, побудована нелінійна модель має негірші показники якості, ніж лінійна модель у припущенні про нормальність вихідних даних [19].

На рисунку 2.3 зображені довірчий інтервал, інтервал прогнозування та саме рівняння регресії для ненормалізованих даних.

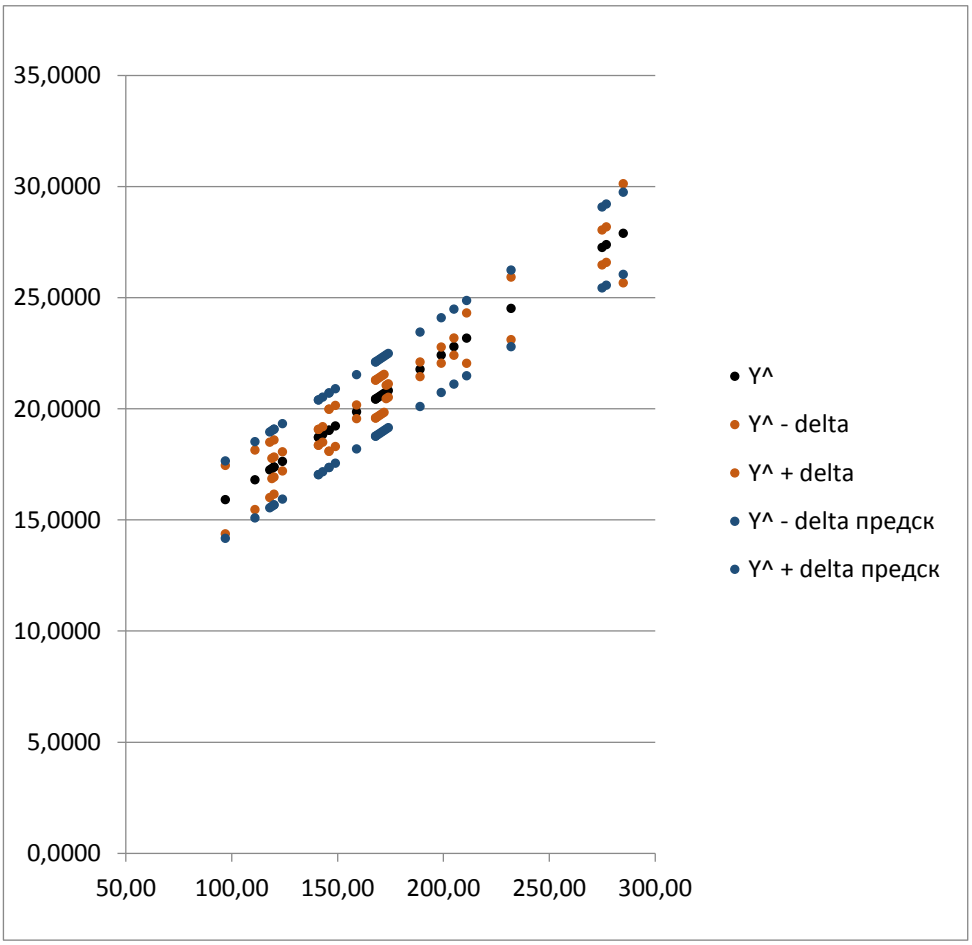

Рисунок 2.3 – Довірчий інтервал, інтервал прогнозування та рівняння регресії для ненормалізованих даних.

На рисунку 2.4 зображені довірчий інтервал, інтервал прогнозування, нелінійне рівняння регресії для даних, нормалізованих за допомогою натурального логарифму.

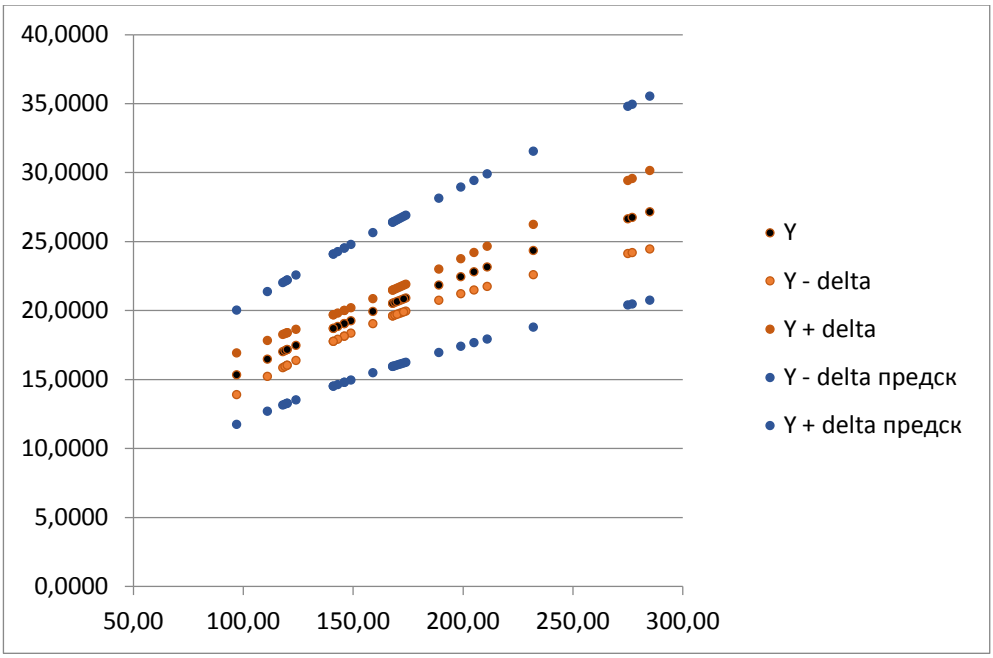

Рисунок 2.4 – Довірчий інтервал, інтервал прогнозування та нелінійне рівняння регресії для даних, нормалізованих за допомогою натурального логарифму

Таким чином, прогнозування для нормалізованих даних дає кращий результат, ніж для ненормалізованих.

2.2 Постановка задачі на розробку ПЗ для оцінювання трудомісткості розробки скраперів на Java

Виходячи з вищевикладеного сформулюємо постановку задачі на розробку програмного забезпечення. В даній роботі необхідно розробити програмне забезпечення для оцінювання трудомісткості розробки скраперів на Java, яке буде реалізовувати удосконалену математичну модель.

Програма повинна надавати наступні функції:

1) Введення емпіричних даних без викидів;

2) Нормалізація емпіричних даних з допомогою натурального логарифму;

3) Побудова лінійної регресійної моделі для нормалізованих даних;

4) Побудова довірчого інтервалу для нормалізованих даних;

5) Побудова інтервалу прогнозування для нормалізованих даних;

6) Перехід від нормалізованих даних до вихідних за допомогою зворотнього перетворення;

7) Пункти 3-6 повторити для вихідних даних у припущені про нормальність розподілу;

8) Обчислити метрики якості лінійної та нелінійної регресії;

9) Побудувати графіки з довірчим інтервалом, інтервалом прогнозування та прогнозними значеннями для ненормалізованих даних;

10) Побудувати графіки з довірчим інтервалом, інтервалом прогнозування та прогнозними значеннями для нормалізованих даних.

3 ПРОЕКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ ТРУДОМІСТКОСТІ РОЗРОБКИ СКРАПЕРІВ НА JAVA

Під час розробки проекту програмного забезпечення використовується об’єктно-орієнтована методологія та уніфікована мова моделювання UML. В якості CASE- засобу був обраний Rational Rose.

3.1 Ескізній проект програмного забезпечення для оцінювання трудомісткості розробки скраперів на Java

Під час ескізного проектування була побудована модель програмного забезпечення, яка представлена діаграмою варіантів використання і специфікаціями варіантів використання. Для формалізації представлення сценаріїв варіантів використання були побудовані діаграми діяльності для основних варіантів використання, згідно з якими був спроектований проект користувальницького інтерфейсу, а також розроблена інформаційна модель (концептуальна модель даних предметної галузі).

3.1.1 Модель варіантів використання ПЗ

Для визначення та аналізу вимог до програмного забезпечення розробимо модель варіантів використання.

Особу, що працює з програмним забезпеченням, названо «Користувач». Короткий опис актора наведено в таблиці 3.1.

Таблиця 3.1 – Опис суб’єктів взаємодії

Актор	Опис варіантів використання
Користувач	У користувача є функція введення емпіричних даних, приведення їх до нормальної форми, побудови рівняння регресії, оцінювання якості отриманого рівняння

Розроблюване програмне забезпечення має певні функції, які в мові UML відображаються у вигляді варіантів використання (табл. 3.2).

Таблиця 3.2 – Короткий опис варіантів використання

Актор	Назва	Короткий опис
Користувач	Ввести емпіричні дані	Прецедент відповідає за введення емпіричних даних
Користувач	Привести дані до нормальної форми	Прецедент відповідає за приведення даних до нормальної форми
Користувач	Побудувати рівняння регресії	Прецедент відповідає за побудову рівняння регресії
Користувач	Побудувати довірчий інтервал для оцінки трудомісткості розробки ПЗ	Прецедент відповідає за побудову довірчого інтервалу для оцінки трудомісткості розробки ПЗ
Користувач	Побудувати інтервал прогнозування для оцінки трудомісткості розробки ПЗ	Прецедент відповідає за побудову інтервалу прогнозування для оцінки трудомісткості розробки ПЗ
Користувач	Побудувати графіки довірчих інтервалів та інтервалів прогнозування	Прецедент відповідає за побудову графіків довірчих інтервалів та інтервалів прогнозування
Користувач	Оцінити якість	Прецедент відповідає за обчислення коефіцієнту детермінації, середньої величини відносної похибки, рівня прогнозування

Варіанти використання призначені в першу чергу для визначення функціональних вимог до системи і керують усім процесом розробки. Під час аналізу і проектування варіанти використання дозволяють зрозуміти як результати, які хоче отримати користувач, впливають на архітектуру системи і як повинні себе вести компоненти системи, для того щоб реалізувати потрібну для користувача функціональність. Модель варіантів використання зображена на рисунку 3.1.

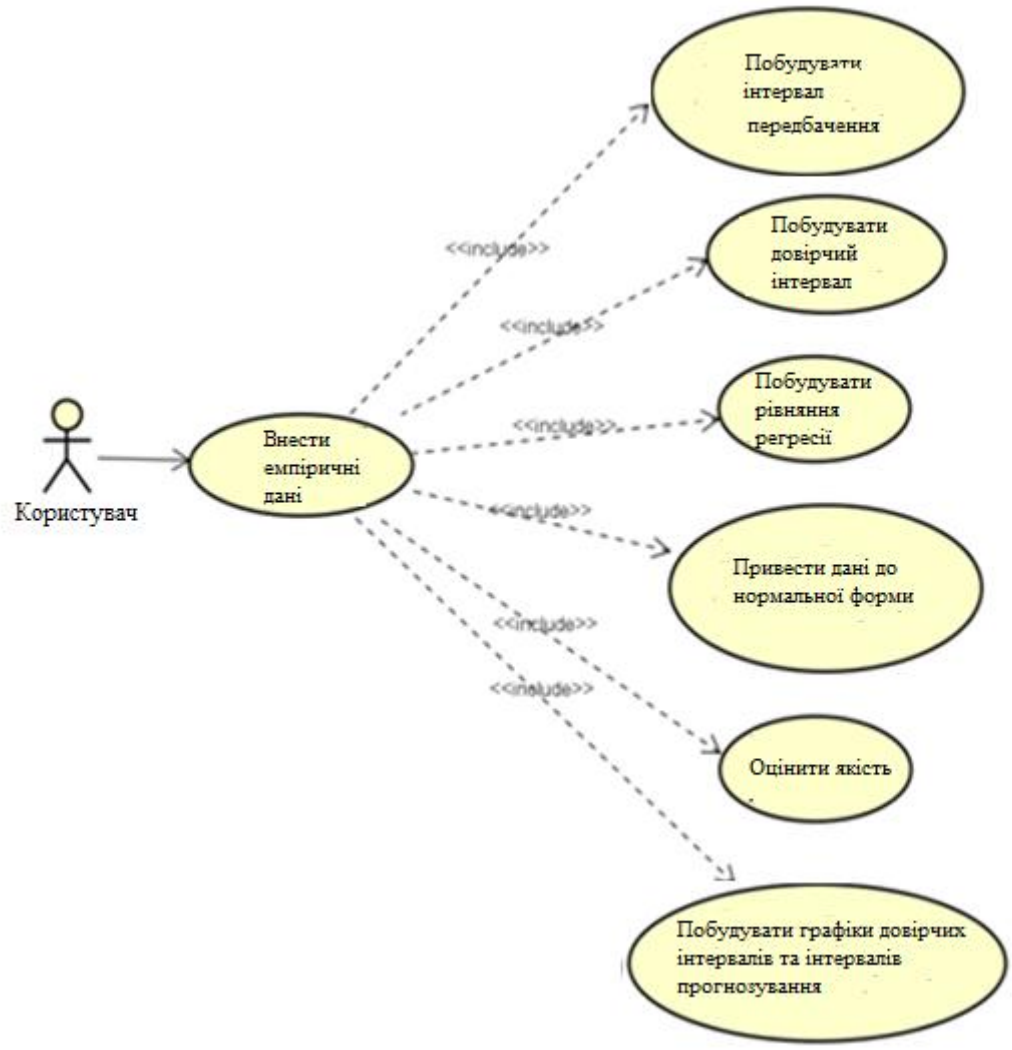

Рисунок 3.1 – Модель варіантів використання ПЗ

Для більш зрозумілого сприйняття моделі варіантів використання, наведеної на рисунку 3.1, необхідно привести специфікації варіантів використання.

Специфікація варіанту використання «Ввести емпіричні дані»

Основна дійова особа: Користувач.

Інші учасники прецеденту: Відсутні.

Короткий опис: Прецедент відповідає за введення емпіричних даних після їх очищення від викидів.

Потоки подій

Базовий потік:

1. Користувач обирає файл з емпіричними даними без викидів
2. Програма зчитує та аналізує дані з файлу

Альтернативні потоки:

1. Користувач обирає файл з емпіричними даними без викидів
2. Програма зчитує файл та видає повідомлення про невідповідність формату.

Спеціальні умови: Відсутні.

Передумова: Відсутня.

Постумова:Відсутня.

Специфікація варіанту використання «Привести дані до нормальної форми»

Основна дійова особа: Користувач.

Інші учасники прецеденту: Відсутні.

Короткий опис: Прецедент відповідає за нормалізацію даних, використовуючи нормалізуюче перетворення lnx до нормальної форми.

Потоки подій

Базовий потік:

Програма застосовує нормалізуюче перетворення до емпіричних даних та отримує дані, які мають гаусовий розподіл

Альтернативні потоки: Відсутні

Спеціальні умови: Відсутні.

Передумова: Наявність введеного тексту.

Постумова:Відсутня.

Специфікація варіанту використання «Побудувати рівняння регресії»

Основна дійова особа: Користувач.

Інші учасники прецеденту: Відсутні.

Короткий опис: Прецедент відповідає за побудову рівняння регресії.

Потоки подій

Базовий потік:

Програма, застосовуючи метод найменших квадратів, обчислює коефіцієнти рівняння регресії.

Альтернативні потоки: Відсутні

Спеціальні умови: Відсутні.

Передумова: Наявність введеного та нормалізованого тексту.

Постумова:Відсутня.

Специфікація варіанту використання «Побудувати довірчий інтервал»

Основна дійова особа: Користувач.

Інші учасники прецеденту: Відсутні.

Короткий опис: Прецедент відповідає за побудову довірчого інтервалу для оцінки трудомісткості розробки ПЗ.

Потоки подій

Базовий потік:

1. Програма обчислює ліву та праву межі довірчого інтервалу для оцінки трудомісткості розробки ПЗ

2. Програма по запиту виводить на друк довірчі інтервали для нормалізованих та вихідних даних

Альтернативні потоки: Відсутні

Спеціальні умови: Відсутні.

Передумова: Наявність введених і нормалізованих даних, а також розрахованого передбачення

Постумова:Відсутня.

Специфікація варіанту використання «Побудувати інтервал прогнозування»

Основна дійова особа: Користувач.

Інші учасники прецеденту: Відсутні.

Короткий опис: Прецедент відповідає за побудову інтервалу прогнозування для оцінки трудомісткості розробки ПЗ

Потоки подій

Базовий потік:

1. Програма обчислює ліву та праву межі інтервалу прогнозування для оцінки трудомісткості розробки ПЗ

2. Програма по запиту виводить на друк інтервали прогнозування для нормалізованих та вихідних даних

Альтернативні потоки: Відсутні

Спеціальні умови: Відсутні.

Передумова: Наявність введених і нормалізованих даних, а також розрахованого передбачення

Постумова:Відсутня.

Специфікація варіанту використання «Оцінити якість»

Основна дійова особа: Користувач.

Інші учасники прецеденту: Відсутні.

Короткий опис: Прецедент відповідає за обчислення метрик регресійного рівняння.

Потоки подій

Базовий потік:

1. Програма обчислює коефіцієнт детермінації
2. Програма обчислює середню величину відносної похибки
3. Програма обчислює рівень прогнозування

4. Програма виводить обчислені метрики на екран

Альтернативні потоки: Відсутні

Спеціальні умови: Відсутні.

Передумова: Наявність введених і нормалізованих даних, а також розрахованого передбачення

Постумова:Відсутня.

Специфікація варіанту використання «Побудувати графіки довірчих інтервалів та інтервалів передбачення»

Основна дійова особа: Користувач.

Інші учасники прецеденту: Відсутні.

Короткий опис: Прецедент відповідає за побудову графіків довірчих інтервалів та інтервалів передбачення

Потоки подій

Базовий потік:

1. Програма будує графіки довірчих інтервалів та інтервалів передбачення для нормалізованих та ненормалізованих даних

Альтернативні потоки: відсутні

Спеціальні умови: Відсутні.

Передумова: Наявність введених і нормалізованих даних, а також розрахованого передбачення

Постумова:Відсутня.

3.1.2 Сценарії основних варіантів використання

Сценарії основних варіантів використання у вигляді діаграм діяльності зазвичай створюються для формалізації представлення основних варіантів використання. Діаграми діяльності можуть розроблятися або для окремих варіантів використання, або для всього програмного забезпечення у випадку його невеликих розмірів.

На рисунку 3.2 зображена діаграма діяльності для варіанту використання «Оцінити якість» програмного забезпечення для оцінювання трудомісткості розробки скраперів на Java.

Рисунок 3.2 – Діаграма діяльності для варіанту використання «Оцінити якість» ПЗ для оцінювання трудомісткості розробки скраперів на Java

На рисунку 3.3 зображена діаграма діяльності для варіанту використання «Ввести дані» програмного забезпечення для оцінювання трудомісткості розробки скраперів на Java.

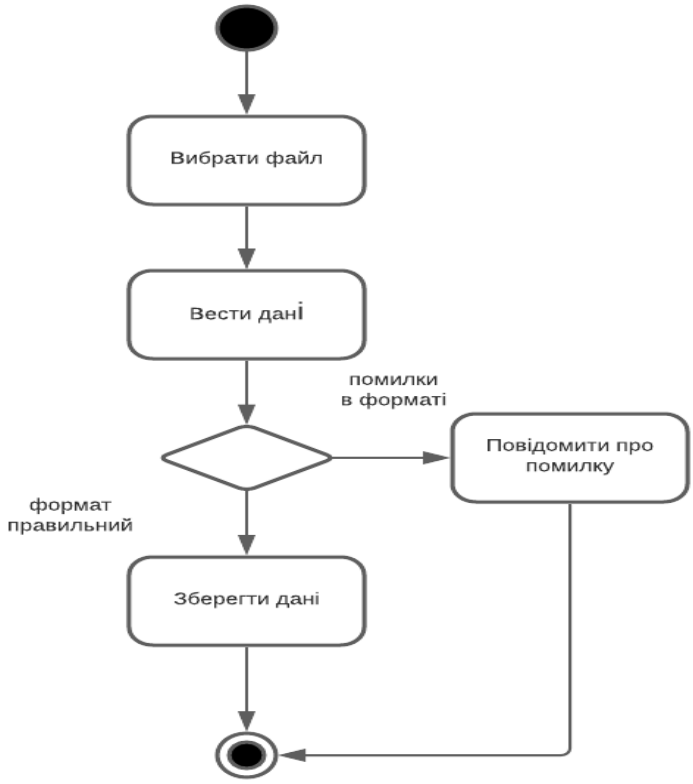

Рисунок 3.3 – Діаграма діяльності ПЗ для оцінювання трудомісткості розробки скраперів на Java

3.1.3 Інформаційна модель ПЗ для оцінювання трудомісткості розробки скраперів на Java

Основними завданнями інформаційної моделі є накопичення інформації, організація комфортного доступу до даних та документів за допомогою легкого в використанні інтерфейсу, аналіз інформації завдяки спеціальним інструментам, що дозволяють вирішувати аналітичні завдання, та візуалізація даних. Побудова інформаційної моделі переслідує основну

мету – отримати якісну базу даних – сховище інформації взаємозв’язаних даних.

В результаті аналізу предметної галузі програмного забезпечення для оцінювання трудомісткості розробки скраперів на Java була побудована інформаційна (концептуальна) модель даних, яка наведена на рисунку 3.4.

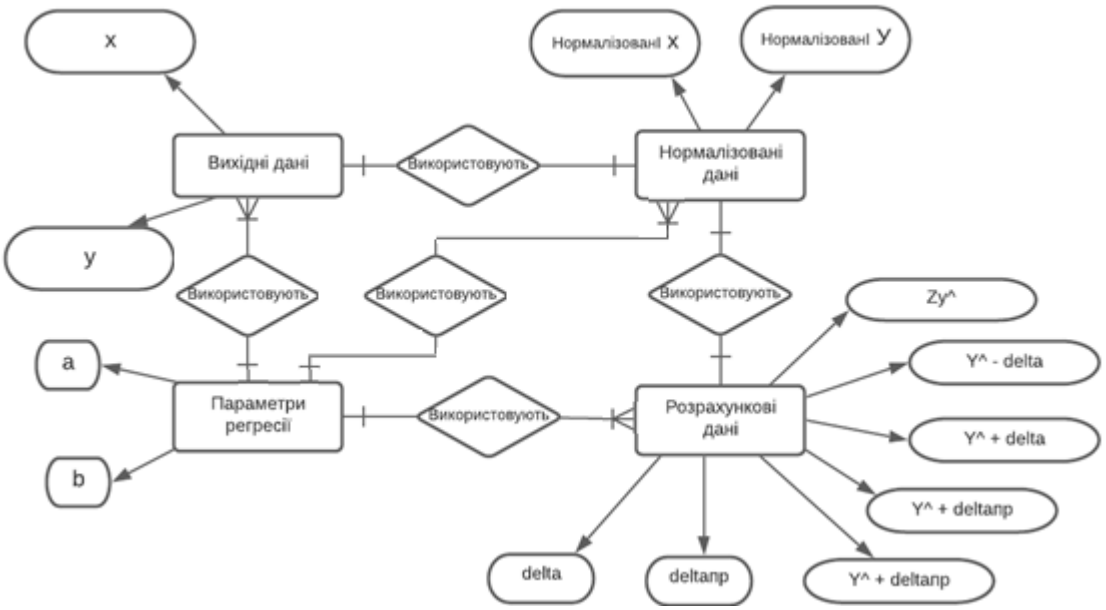

Рисунок 3.4 – Інформаційна модель ПЗ для оцінювання трудомісткості розробки скраперівї на Java

3.1.4 Розробка інтерфейсу користувача ПЗ для оцінювання трудомісткості розробки скраперів на Java

Важливим етапом при проектуванні програмного забезпечення є розробка інтерфейсу. Беручи до уваги вимоги, що зазначені постановці задачі, визначимо зовнішній інтерфейс. Для забезпечення зручної навігації необхідно розробити інтуїтивно-зрозумілий інтерфейс користувача (клієнта):

Головна форма програми представлена на рисунку 3.5.

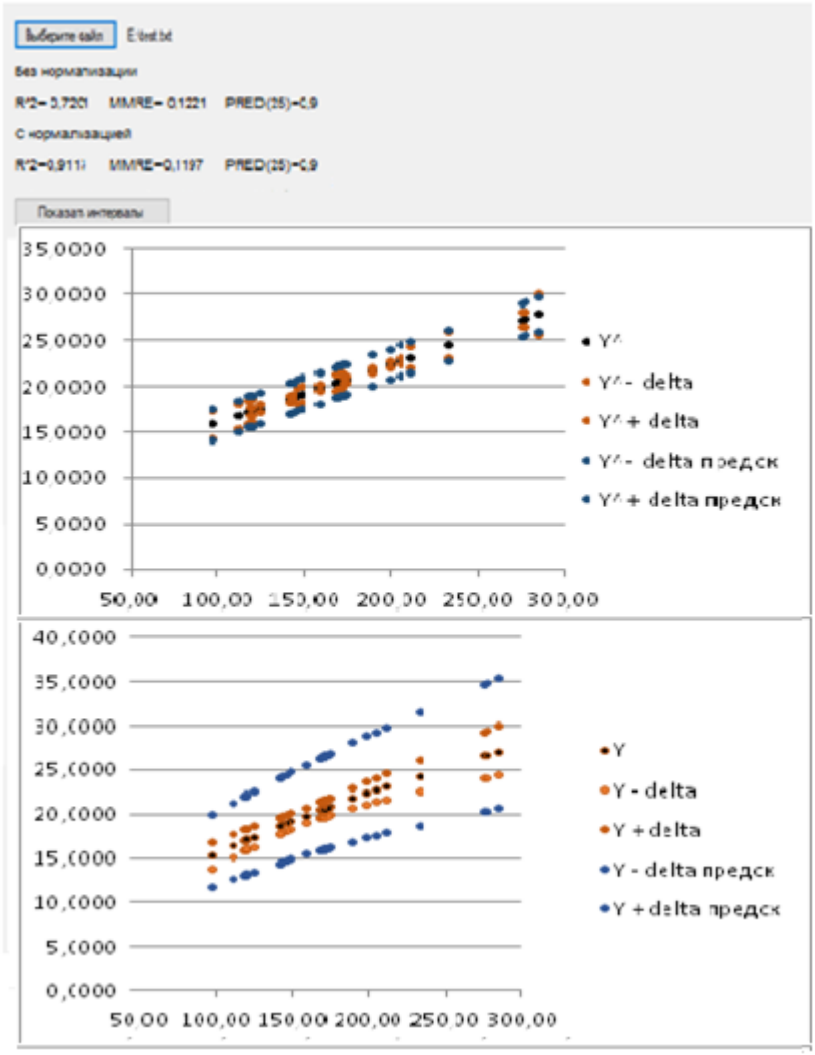

Рисунок 3.5 – Головна форма програми

3.2 Технічний проект програмного забезпечення для оцінювання трудомісткості розробки скраперів на Java

Проаналізувавши ескізний проект, розробили технічний проект даного ПЗ шляхом побудови статичної і динамічної моделей. Статична модель представлена діаграмою класів, а динамічна – діаграмою послідовності.

3.2.1 Діаграма класів програмного забезпечення

На рисунку 3.6 зображено діаграму класів програмного забезпечення

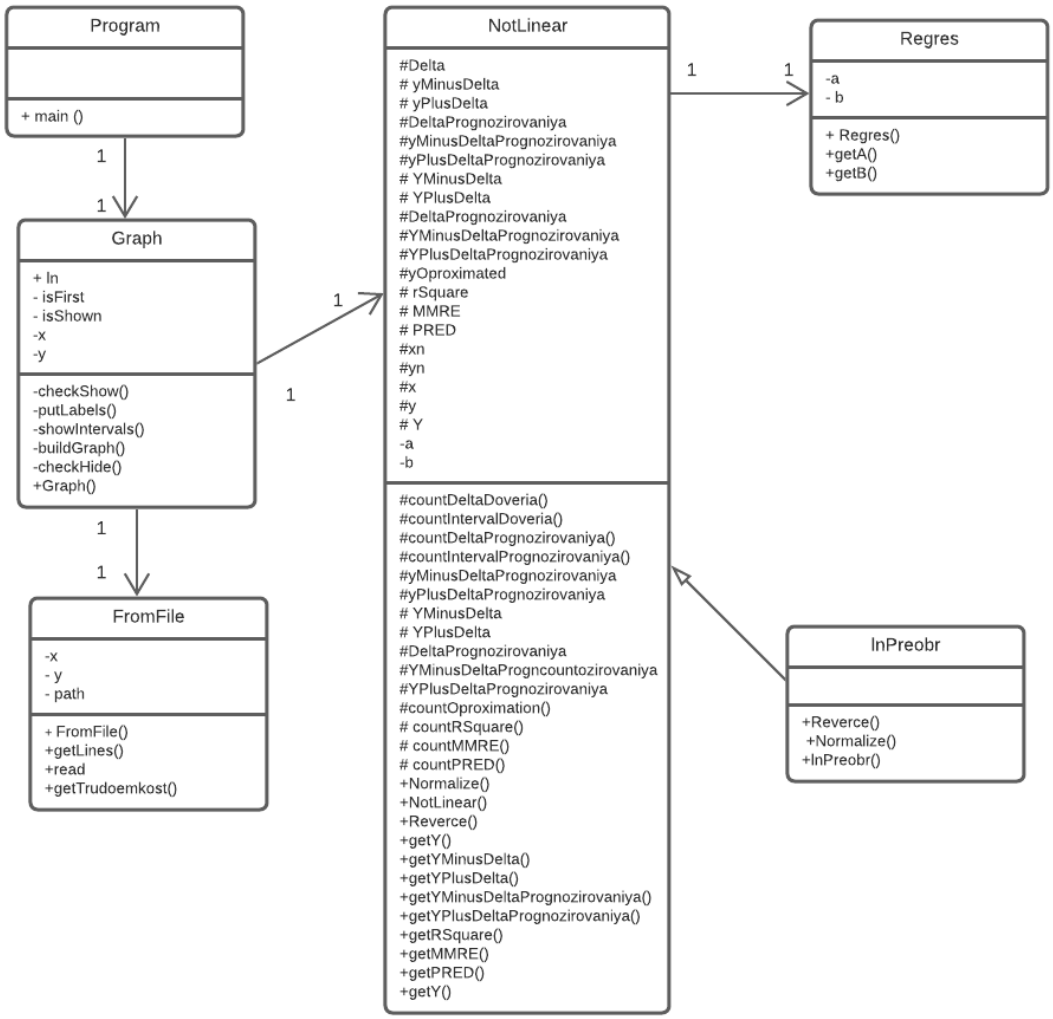

Рисунок 3.6 – Діаграма класів програмного забезпечення

3.2.2 Специфікації класів програмного забезпечення

Розглядаємо специфікації класів

Клас: Program

Призначення: призначений для запуску програми

Включає в себе наступні методи:

Main() – метод для запуску програми

Клас: Regres

Клас призначений для отримання коефіцієнтів рівняння регресії.

Включає в себе наступні атрибути:

a – значення коефіцієнту a

b – значення коефіцієнту b

Включає в себе наступні методи:

Regres() – конструктор класу

getA() – метод призначений для отримання коефіцієнту a

getB() – метод призначений для отримання коефіцієнту b

Клас Graph

Клас призначений для створення графічного інтерфейсу програми.

Включає в себе наступні атрибути:

x – масив значень строк коду

y – масив значень трудомісткості

isShown – змінна, яка показує чи потрібно відображати таблицю зі значеннями інтервалів

ln – зміна, призначена для зберігання екземпляру класа lnPreobr

Включає в себе наступні методи:

Graph() – конструктор класу

buildGraph() – метод для побудови і відображення графіків інтервалів

putLabels() – метод для відображення метрик якості

showIntervals() – метод призначений для відображення порахованих інтервалів

checkShow() – метод для перевірки того, що в даний момент інтервали відображаються на формі

checkHide() – метод для перевірки того, що в даний момент інтервали не відображаються на формі

Клас: FromFile

Клас призначений для зчитування даних з файлу.

Включає в себе наступні атрибути:

x – призначений для зберігання id даної програми

y – призначений для зберігання імені даної програми

path – призначений для зберігання шляху до файлу з даними

Включає в себе наступні методи:

FromFile() – конструктор класу

getLines() – метод для отримання масиву зі значеннями строк коду

getTrudoemkost() – метод для отримання масиву зі значенями трудомісткості

read() – метод для зчитування даних з файлу

Клас: lnPreobr

Клас призначений для нормалізації даних за допомогою натурального логарифму і для подальшого зворотнього перетворення.

Включає в себе наступні методи:

Клас: lnPreobr () – конструктор класу

Normalize() – метод для нормалізації даних за допомогою натурального логарифму

Reverse() – метод для зворотнього перетворення

Клас: NotLinear

Клас, який відповідає за основні розрахунки

Включає в себе наступні атрибути:

x – масив значень строк коду

y – масив значень трудомісткості

yOproximated – масив апроксимованих y

xn – нормалізовані x

yn – нормалізовані y

delta – Δ для довірчого інтервалу

yMinusDelta – нижня границя довірчого інтервалу

yPlusDelta – верхня границя довірчого інтервалу

deltaPrognozirovaniya – Δ інтервалу передбачення

yMinusDeltaPrognozirovaniya – нижня границя інтервалу передбачення

yPlusDeltaPrognozirovaniya – верхня границя інтервалу передбачення

Y – y після зворотнього перетворення

YMinusDelta – нижня границя довірчого інтервалу після зворотнього перетворення

YPlusDelta – верхня границя довірчого інтервалу після зворотнього перетворення

YMinusDeltaPrognozirovaniya – нижня границя інтервалу передбачення після зворотнього перетворення

YPlusDeltaPrognozirovaniya – верхня границя інтервалу передбачення після зворотнього перетворення

a – значення коефіцієнту a

b – значення коефіцієнту b

rSquare – значення R^2

MMRE – значення MMRE

PRED – значення PRED(25)

Включає в себе наступні методи:

Клас: NotLinear() – конструктор класу

countOproximation() – метод для підрахунку апроксимованих значень y

countDeltaDoveria() – метод для підрахунку Δ довірчого інтервалу

countIntervalDoveria() – метод для підрахунку довірчого інтервалу

countDeltaPrognozirovaniya() – метод для підрахунку Δ інтервалу передбачення

countIntervalPrognozirovaniya() – метод для підрахунку інтервалу передбачення

countRSquare() – метод для підрахунку R^2

countMMRE() – метод для підрахунку MMRE

countPRED() – метод для підрахунку PRED(25)

getY() – метод для отримання значень у після зворотнього перетворення

getYMinusDelta() – метод для отримання нижньої границі довірчого інтервалу після зворотнього перетворення

getYPlusDelta() – метод для отримання верхньої границі довірчого інтервалу після зворотнього перетворення

getYMinusDeltaPrognozirovaniya() – метод для отримання нижньої границі для інтервалу передбачення після зворотнього перетворення

getYPlusDeltaPrognozirovaniya() – метод для отримання верхньої границі для інтервалу передбачення після зворотнього перетворення

getRSquare() – метод для отримання R^2

getMMRE() – метод для отримання MMRE

getPRED() – метод для отримання PRED(25)

3.2.3 Динамічна модель програмного забезпечення

На основі моделі сценаріїв варіантів використання і статичної моделі програми була створена динамічна модель ПЗ, яка представлена діаграмою послідовності програмного забезпечення (рис.3.7). На діаграмі послідовності зображена взаємодія користувача з системою у часі

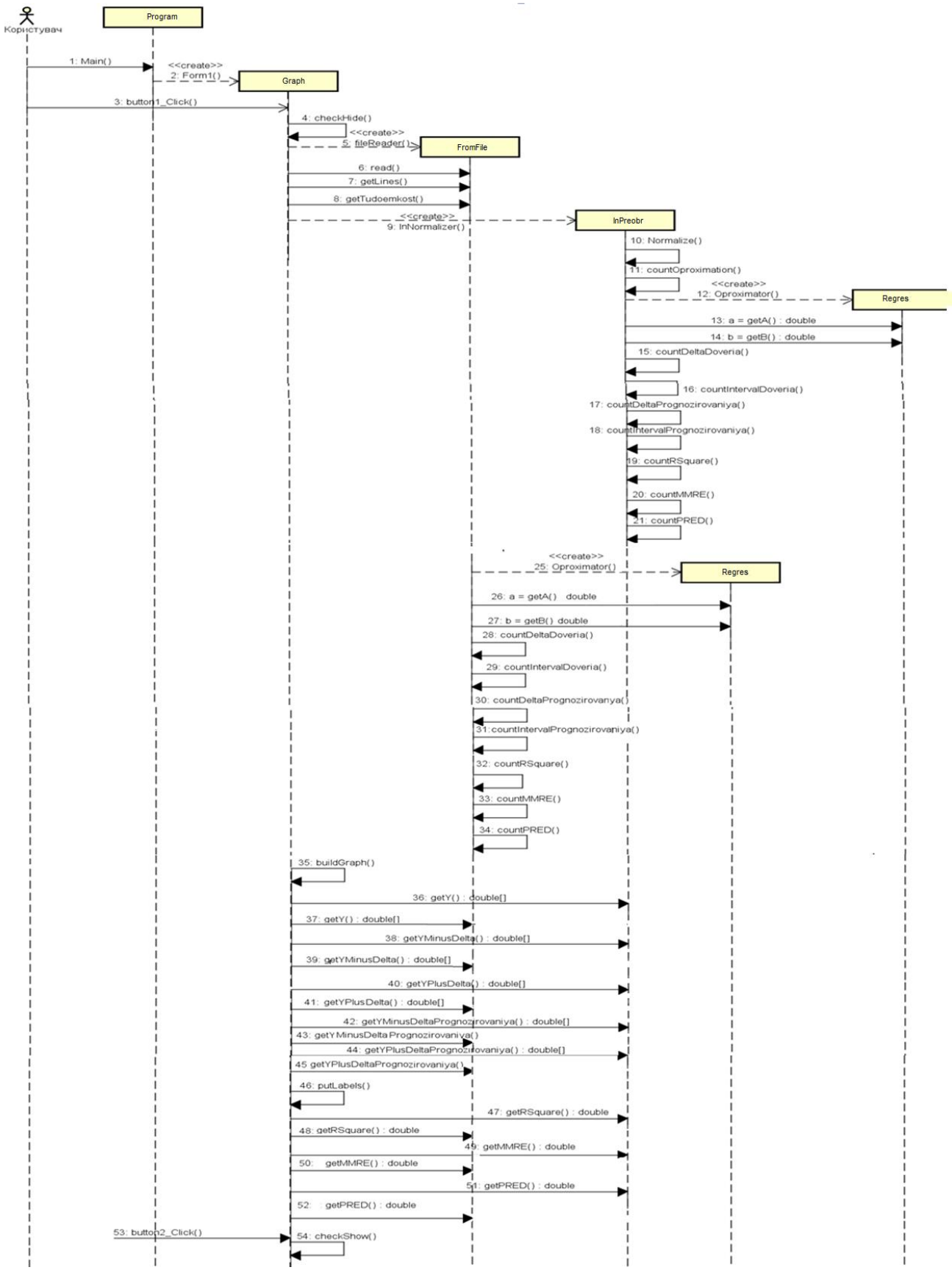

Рисунок 3.7 – Динамічна модель програмного забезпечення

3.3 Робочий проект програмного забезпечення для оцінювання трудомісткості розробки скраперів на Java

На стадії робочого проекту рішення, прийняті на попередніх етапах, перетворюються у форму, доступну користувачеві. Вибираються засоби розробки програмного забезпечення, створюється діаграма компонентів, виконуються реалізація, тестування та випробування програмного забезпечення.

В межах сучасної технології докладна розробка алгоритму кожної програми комплексу повинна бути повністю закінчена до початку його реалізації на мові програмування.

3.3.1 Обґрунтування вибору мови програмування та засобів розробки програмного забезпечення

Програмне забезпечення проектувалося під розробку на об’єктно-орієнтованій мові програмування. Найбільш поширеними об’єктно-орієнтованими мовами на даному етапі є Java та C#.

Початок створення мови програмування Java – 1991 рік. Її створили Джеймс Гослінг, Майк Шеридан та Патрік Нотон. Java – об'єктно-орієнтована, проста і безпечна. Синтаксис Java схожий з синтаксисом C і C ++. Але спеціалісти вважають, що Java не розвивається.

Мова C# розроблена Андерсом Хейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft. Вона є мовою з високим рівнем абстракції, по синтаксису схожа на C ++ та Java.

Переваги C #:

– C# розроблялася разом з новим каркасним фреймворком Framework.Net і, відповідно, позбавлена всіх «принад» підстроювання, адже основний девіз .Net – це єдина платформа для множини мов.

– У C# необмежені можливості успадкування та універсалізації.

– C# проста в освоєнні.

– C # надійна.

– Код C # досить ефективний.

– Висока швидкодія.

– Невеликий розмір результуючих програм.

Тому в якості мови програмування обрана мова програмування C #.

3.3.2 Діаграма компонентів програмного забезпечення для оцінювання трудомісткості розробки скраперів на Java

Діаграма компонентів призначена для візуалізації загальної організації структури вихідного коду програми.

Діаграма компонентів програмного забезпечення для оцінювання трудомісткості розробки скраперів на Java представлена на рисунку 3.8

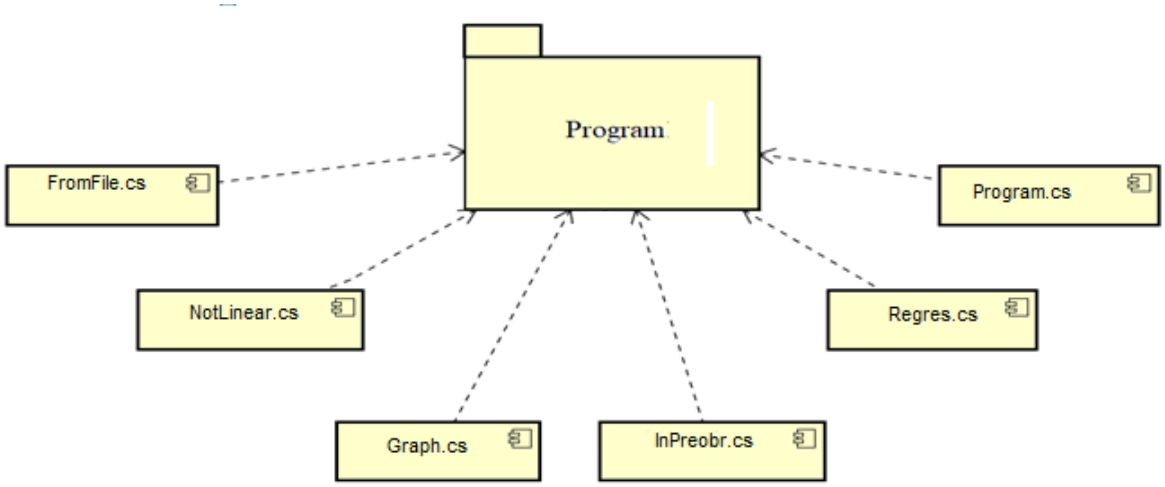

Рисунок 3.8 – Діаграма компонентів програмного забезпечення для оцінювання трудомісткості розробки скраперів

3.3.3 Кодування та тестування програмного забезпечення для оцінювання трудомісткості розробки скраперів на Java

Мова програмування – C#. Код програми наведений в додатку D.

Тестування здійснювали одним метод «чорного ящика», а саме, – розбиттям на класи еквівалентності. Розглянемо тестування класів програми. Програма була протестована повністю. В роботі наведені приклади тестування.

– Додавання даних до програми:

- Натиснути на кнопку «Вибрати файл».
- Вибрати шлях до файлу.
- Файл відкривається і поблизу кнопки з’являється назва файлу

– Перегляд довірчих інтервалів та інтервалів передбачення

- Прочитати дані з файлу.
- Натиснути на кнопку «Показати інтервали».
- Інтервали зявляються на екрані у вигляді таблиці.
- З’являється текст «Сховати інтервали» замість «Показати інтервали».

– Введення файлу з даними неправильного формату

- Ввести дані неправильного формату.
- З’являється повідомлення про неправильний формат даних.

3.3.4 Випробування програмного забезпечення для оцінювання трудомісткості розробки скраперів на Java

Програма і методика випробувань програмного забезпечення наведена в Додатку В.

Тести «чорної скриньки» пишуться із використанням специфікації тестового програмного забезпечення. У результаті опрацювання специфікації

до програмного забезпечення, що підлягає тестуванню, було отримано ряд тестів «чорної скриньки».

Результати тестування за принципом "чорного ящика" наведені в таблиці 3.3.

Таблиця 3.3 – Результати тестування за принципом "чорного ящика"

Тест	Сценарій	Результат
Ввести дані	Користувач увів дані правильного формату	Програма зчитала дані з програми
	Користувач ввів дані неправильного формату	Відображаються повідомлення про неправильний формат даних
Показати інтервали	Користувач натиснув кнопку «Показати інтервали»	Програма відобразила інтервали, кнопка змінила текст на «Сховати інтервали»
Сховати інтервали	Користувач натиснув кнопку «Сховати інтервали»	Програма сховала інтервали і кнопка змінила текст на «Показати інтервали»
Показати графіки інтервалів	Користувач ввів дані правильного формату	Програма відобразила графіки інтервалів
Розрахувати метрики якості	Користувач ввів дані правильного формату	Програма виводить метрики якості для обох видів нормалізації

Дані випробування показали, що розроблене програмне забезпечення функціонує нормально.

4 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ ТРУДОМІСТКОСТІ РОЗРОБКИ СКРАПЕРІВ НА JAVA

У даній кваліфікаційній роботі на основі удосконаленої математичної моделі було розроблене програмне забезпечення для оцінювання трудомісткості розробки скраперів на Java.

Під час виконання роботи були вирішені наступні завдання:

- проаналізовано існуючі моделі та методи оцінювання трудомісткості розробки ПЗ, сформулювано постановку задачі;

- обрано в якості нормалізуючого перетворення для емпіричних даних, отриманих із попередньо завершених проектів, натуральний логарифм;

- розроблено нелінійну регресійну модель оцінювання трудомісткості розробки програмного забезпечення в залежності від кількості рядків коду;

- перевірено якість моделі для оцінювання трудомісткості розробки програмного забезпечення;

- для порівняння нелінійної регресійної моделі для нормалізованих даних розроблено лінійну регресійну модель у припущенні про нормальність вихідних даних.

- порівняно отримані результати по двом моделям.

- розроблено ПЗ для оцінювання трудомісткості розробки програмного забезпечення, використовуючи отриману модель.

Також в роботі були розроблені додатки: Додаток А –Технічне завдання, Додаток Б – Опис програми, Додаток В – Програма і методика випробувань, Додаток Г – Інструкція користувача, Додаток Д – текст програми.

Розмір вихідного файлу становить 340 КБ.

ПЗ для оцінювання трудомісткості розробки призначене для використання на персональних комп'ютерах, що працюють під управлінням наступних операційних систем MS Windows XP/7/8/8.1/10

Для ПЗ пред'являються такі апаратні вимоги до ПЕОМ:

– оперативна пам'ять 512 МБ;

– дисковий простір – не менше 512 МБ вільного місця на диску;

– роздільна здатність екрану – 1280 x 1024 пікселів.

Також було виконано тестування та випробування програмного забезпечення, які показали відповідність програмного забезпечення технічному завданню.

5 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ РОЗРОБКИ ТА ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ ТРУДОМІСТКОСТІ РОЗРОБКИ СКРАПЕРІВ НА JAVA

Дана кваліфікаційна робота присвячена розробці програмного забезпечення для оцінювання трудомісткості розробки скраперів на Java, яке може використовуватися у багатьох випадках.

Обґрунтувати доцільність розробки з економічної точки зору можливо, розрахувавши собівартість програмного забезпечення та прибуток, що буде отриманий в результаті продажу. Основним показником, що визначає економічну доцільність витрат на створення продукту є річний економічний ефект. Характерним показником економічної ефективності створення і функціонування програмного продукту, що розраховується на стадії створення і впровадження – є ефективність витрат, яку характеризує строк окупності програмного забезпечення.

У даному підрозділі розраховується собівартість розробки програмного забезпечення. Також виконується оцінка ринкової ціни та об’ємів продажу копій програми. На основі цих даних розраховується річний економічний ефект та строк окупності розробки.

5.1 Розрахунок витрат на створення та впровадження програмного забезпечення

Витрати на розробку програмного забезпечення складаються з витрат на заробітну плату розробника, на амортизацію та експлуатацію ЕОМ, на якій виконується розробка, на засоби розробки та витрати на матеріали і комплектуючі.

Розробка програмного забезпечення виконується спеціалістом, місячний оклад якого складає 15000 грн. Додаткова заробітна плата складає 10% від основної. Виходячи з цього, основна і додаткова заробітна плата розроблювача системи 16500 грн./міс, а вартість сучасної ПЕОМ складає 9000 грн. (середня вартість машини на базі AMD FX). Вартість кіловат-години електроенергії дорівнює 1,68 грн. Витрати на допоміжні матеріали наведені в табл. 5.1

Таблиця 5.1 – Витрати на допоміжні матеріали

Пункти витрат	Сума, грн.
Папір	90
Заправлення картриджа до принтера	120
Непередбачені витрати	200
Разом матеріали і комплектуючі вироби.	215

Вартість розробки системи розраховується за формулою:

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

де Т – тривалість розробки, міс.

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

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

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

$З_{е}$ – витрати на електроенергію;

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

Розрахуємо $З_{е}$ при споживанні потужності 0,3 Вт, тривалості роботи на місяць рівної $21 * 8 = 168$ годин і вартості кіловат-години електроенергії 1,68 грн.,

$$З_{е} = 168 * 1{,}68 * 0{,}3 = 45{,}36 \text{ грн.}$$

Представимо всі поточні витрати на розробку програмного забезпечення в таблиці 5.2.

Таблиця 5.2 – Витрати на розробку програмного забезпечення

Найменування витрат	Одиниця виміру	Кількість
Тривалість розробки	міс.	4
Основна і додаткова заробітні плати	грн.	16500
Відрахування на соціальні заходи	грн.	6270
Загальногосподарські витрати	грн.	1500
Витрати на допоміжні матеріали	грн.	215
Витрати на електроенергію	грн.	45,36

Відповідно до формули (5.1) вартість розробки ПЗ складає:

$$С_{пр} = (16500 + 6270 + 1500 + 45{,}36) * 3 + 215 = 73161.08 \text{ грн.}$$

Амортизаційні відрахування на устаткування складають 60% балансової вартості в рік: $А_{об} = 9000 * 0{,}6 = 5400$ грн.

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

$$В_м = 9000 * 0{,}05 = 450 \text{ грн.}$$

Річний обсяг робіт ПЕОМ у годинах визначається в такий спосіб:

$$Ф_м = 253{,}3 * Т_з$$

де $Т_з$ – це середнє місячне завантаження устаткування (близько 7 годин), 253,3 – середня кількість робочих днів у році:

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

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

$$З_е = 1773{,}1 * 0{,}3 * 1{,}68 = 478{,}74 \text{ грн.}$$

Експлуатаційні витрати для ПЕОМ за рік складуть:

$$З_{зр} = 5400 + 450 + 478{,}74 = 6328.74 \text{ грн.}$$

В перший рік витрати на створення й експлуатацію програмного забезпечення складуть:

$$З_{се} = 73161.08 + 6328.74 = 79489.82 \text{ грн.}$$

5.2 Розрахунок економічної ефективності розробки

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

Зарплата одного працівника в рік складає:

$$\mathbf{7500 * 12 * 1 = 90000}\ \text{грн.}$$

Річний економічний ефект розраховується по формулі:

$$Э_{рік} = \Delta C_n - E_n \cdot k\ , \qquad (5.2)$$

де $\boldsymbol{\Delta C_n}$ – вивільнені кошти після впровадження системи (90000 грн.) мінус експлуатаційні витрати (6328.74 грн.);

$\boldsymbol{E_n}$ – коефіцієнт ефективності(дорівнює коефіцієнту амортизації (0,6));

k – одноразові витрати на впровадження продукту (73161.08).

$$Э_{рік} = \mathbf{90000 - 6328.74 - 73161.08 * 0,6 = 39774.612}\ \text{грн.}$$

Строк окупності системи розраховується по формулі:

$$\boldsymbol{T = \frac{k}{\Delta C} = \frac{73161.08}{83671.26} \approx 0.87}\ \text{року}$$

Отже строк окупності програмного забезпечення складає приблизно 10 місяців.

5.3 Висновки

У цьому розділі були здійснені розрахунки на створення й експлуатацію програмного забезпечення, а також розрахунки економічної ефективності та строку окупності програмного забезпечення. Виходячи з розрахунків, впровадження такого програмного забезпечення окупить себе протягом першого року експлуатації за 10 місяців. Отже, можна зробити висновок, що дане програмне забезпечення економічно вигідне та обґрунтоване.

6 ОХОРОНА ПРАЦІ

6.1 Основні положення та норми охорони праці

Охорона праці – діюча на підставі відповідних законодавчих та інших нормативних актів система соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, що забезпечують збереження здоров'я і працездатності людини в процесі праці.

Умови праці – це сукупність взаємозв'язаних санітарно-гігієнічних, психофізіологічних, естетичних і соціальних факторів конкретної праці, обумовлених рівнем розвитку продуктивних сил суспільства, які визначають стан середовища та впливають на здоров'я і працездатність людини

Система управління охороною праці (СУОП) – це сукупність суб'єкта та об'єкта управління, які на підставі комплексу нормативної документації проводять цілеспрямовану, планомірну діяльність з метою забезпечення здорових, безпечних і високопродуктивних умов праці. Охорона праці базується на законах та інших нормативно-правових актах, які є головним джерелом зовнішньої інформації, що надходить до СУОП.

Створення СУОП здійснюється шляхом послідовного визначення мети і об'єкта управління, завдань і заходів щодо охорони праці, функцій і методів управління, побудови організаційної структури управління, складання нормативно-методичної документації.

Організаційно-методичну роботу по управлінню охороною праці, підготовку, правлінських рішень і контроль за їх своєчасною реалізацією здійснює служба охорони праці підприємства, що підпорядкована безпосередньо керівнику підприємства (головному інженеру). Суб'єкт управління аналізує інформацію про стан охорони праці в структурних

підрозділах підприємства та приймає рішення спрямовані на приведення фактичних показників охорони праці у відповідність з нормативними.

Об'єктом управління в СУОП є діяльність структурних підрозділів та служб по забезпеченню безпечних і здорових умов праці на робочих місцях.

Охорона праці базується на законодавчих, директивних та нормативно-технічних документах. При управлінні охороною праці не повинні прийматись рішення та проходити заходи, що суперечать чинному законодавству, державним нормативним актам про охорону праці, стандартам безпеки праці, правилам та нормам охорони праці.

До основних функцій управління охороною праці належать:

- облік показників стану умов і безпеки праці;
- аналіз та оцінка стану умов і безпеки праці;
- прогнозування і планування робіт, їх фінансування;
- організація та координація робіт;
- контроль за функціонуванням СУОП.

Основні завдання управління охороною праці:

- навчання працівників безпечним методам праці та пропаганда питань охорони праці;
- забезпечення безпечності технологічних процесів, устаткування, будівель і споруд;
- професійної підготовки і підвищення кваліфікації працівників з питань охорони праці, пропаганди безпечних методів праці;
- вибору оптимальних режимів праці і відпочинку працівників;
- професійного добору виконавців для визначених видів робіт.

Правові та організаційні питання:

- нормалізація санітарно-гігієнічних умов праці;
- забезпечення працівників засобами індивідуального захисту;
- забезпечення оптимальних режимів праці та відпочинку;
- організація лікувально-профілактичного обслуговування;
- удосконалення нормативної бази з питань охорони праці.

6.2 Аналіз небезпечних і шкідливих виробничих факторів, характерних для офісу комп'ютерної фірми

Шкідливим фактором називається такий фактор, вплив якого на робітника в певних умовах призводить до захворювання або зниження працездатності.

Небезпечний фактор – фактор, дія якого за певних умов може призвести до травм або іншого раптового погіршення здоров'я працівника.

Відповідно до ДСТ 12 0.003-74 небезпечні та шкідливі фактори за природою дії поділяються на такі групи:

- фізичні;
- хімічні;
- біологічні;
- психофізіологічні.
- Фізичні виробничі фактори поділяються на:
- рухомі машини та механізми;
- рухомі частини виробничого обладнання;
- шум;
- вібрацію (загальну і локальну);
- інфразвук;
- ультразвук;
- електромагнітні поля радіочастотного діапазону;
- електричні поля промислової частоти;
- електростатичні поля;
- лазерне випромінювання;
- іонізуюче випромінювання;
- освітленість;
- ультрафіолетове випромінювання;

- мікроклімат у виробничому приміщенні (температура повітря, швидкість руху повітря, відносна вологість повітря, інтенсивність
- інфрачервоного (теплового) випромінювання);
- аероіонізація повітря;
- атмосферний тиск (підвищений, знижений).

Хімічні виробничі фактори поділяються за характером впливу на організм людини на:

- токсичні;
- дратівливі;
- канцерогенні;
- мутагенні і т.д.

Хімічні виробничі фактори за способом проникнення в організм людини поділяються на проникаючі через:

- органи дихання;
- шлунково-кишковий тракт;
- шкірні покриви і слизові оболонки.

Біологічні виробничі фактори включають біологічні об'єкти:

- патогенні мікроорганізми (бактерії, віруси, гриби найпростіші і т.д.);
- продукти їх життєдіяльності;
- мікроорганізми (рослини, тварини);
- природні компоненти організму.

Психофізіологічні виробничі фактори визначаються показниками важкості та напруженості трудового процесу.

Основними показниками важкості трудового процесу є:

- величина фізичної динамічного навантаження;
- разова величина вантажу, що піднімається вручну (маса що піднімається і переміщуваного вантажу вручну);
- статичне навантаження;
- робоча поза і переміщення в просторі (робоча поза; нахили корпусу;

- переміщення в просторі, обумовлені технологічним процесом);
- темп роботи (стереотипні робочі руху);

Основними показниками напруженості трудового процесу є:

- інтелектуальні навантаження;
- напруженість уваги (сенсорні навантаження);
- напруженість функцій аналізаторів (сенсорні навантаження);
- монотонність (монотонність навантажень);
- емоційне напруження (емоційні навантаження);
- естетичний дискомфорт;
- фізіологічний дискомфорт;
- змінність (режим роботи).

6.3 Розрахунок системи штучного освітлення в офісі комп'ютерної фірми

До сучасного виробничого освітлення пред'являються високі вимоги як гігієнічного, так і техніко-економічного характеру. Правильно спроектоване і виконане освітлення забезпечує високий рівень працездатності, надає позитивний психологічний вплив на працюючих, сприяє підвищенню продуктивності праці.

До систем освітлення висувають наступні вимоги:

- відповідність рівня освітленості робочих місць характеру виконуваної зорової роботи;
- досить рівномірний розподіл яскравості на робочих поверхнях і в навколишньому просторі;
- сталість освітленості в часі;
- оптимальна спрямованість випромінюваного освітлювальними приладами світлового потоку;
- довговічність, економічність, електро- та пожежобезпечність, естетичність, зручність і простота експлуатації.

У тих випадках, коли одного природного освітлення в приміщенні недостатньо, створюють поєднане освітлення. При цьому додаткове штучне освітлення застосовують не тільки в темний, але й у світлий час доби.

По конструктивному виконанню штучне освітлення може бути загальними і місцевим. При загальному освітленні всі робочі місця одержують освітлення від загальної освітлювальної установки. Комбіноване освітлення поряд із загальним включає місцеве освітлення, яке зосереджує світловий потік безпосередньо на робочих місцях. Застосування тільки місцевого освітлення неприпустимо, тому що виникає необхідність частої адаптації зору, створюються глибокі і різкі тіні та інші несприятливі чинники. Для штучного освітлення приміщень використовують люмінесцентні лампи, у яких висока світлова віддача і тривалий термін служби.

Розрахунок штучного освітлення будемо виконувати методом коефіцієнта використання світлового потоку.

Розрахуємо потрібну кількість світильників для приміщення з комп'ютерами при загальному рівномірному освітленні. Довжина приміщення А=7 м, ширина приміщення В=5 м, висота приміщення Н=3 м. Висота робочої поверхні $h_р$=1 м. Стеля обладнається світильниками з люмінесцентними лампами денного світла ЛД40. Коефіцієнт відбиття стелі $S_п$=70%, стін $S_с$=50%, робочої поверхні $S_р$=10%. Напруга мережі 220В.

Відстань від стелі до робочої поверхні:

$H_0=H - h_р=3 – 1=2$ м.

Відстань від стелі до світильника:

$hc=0,2*H_0=0,2 * 2=0,4$ м.

Висота підвісу світильника над освітлюваною поверхнею:

$h= H_0 – hc=2 – 0,4=1,6$ м.

Висота підвісу світильника над підлогою:

$H_п=h + h_р=1,6 + 1=2,6$ м.

Для досягнення найбільшої рівномірності освітлення приймаємо відношення L/h=1,5. Тоді відстань між центром і світильником:

L=1,5 * h=1,5 * 1,6=2,4 м.

Потрібна кількість світильників:

$N=S / L^2=(7*5)/2,4^2=6,076$ світильників.

Приймаємо 6 світильників (2 ряди по 3 світильники);

Індекс приміщення:

i =(A*B)/(h*(A+B))=(7*5)/(1,6(7+5))=1,82; (найближче значення в таблиці 2).

По таблиці коефіцієнтів використання світлового потоку при i=2, $S_п$=70%, $S_с$=50%, $S_р$=10% для світильника типу «ЛД» коефіцієнт використання світлового потоку η=0,59.

Світловий потік однієї лампи:

$Ф=(E_{min}*S*K_з*Z) / (N* η)=(100*35*1,5*1,1) / (6*0,59)=1631$ лм;

де $K_з$ – коефіцієнт запасу, який враховує запилення світильників і знос джерел світла в процесі експлуатації; для приміщень, освітлюваних люмінесцентними лампами $K_з$=1,5 за умови чищення світильників не рідше двох разів на рік; Z – коефіцієнт нерівномірності освітлення (Z =1,1).

По знайденому значенню світлового потоку кожної лампи з таблиці, при напрузі мережі 220В, визначаємо її потужність, що має світловий потік 2340 лм, найбільш близькій до розрахункового. При цьому фактична освітленість $E_ф$ буде:

$E_ф=E_{min}(Ф_л / Ф_р)=100*(2340/1631)=143$ лк.

Загальна потужність:

$P_{общ}=P_л*N=40*6=240$ Вт.

6.4 Розробка заходів щодо забезпечення сприятливих умов праці при роботі з персональним комп'ютером в офісі комп’ютерної фірми

Сьогодні вже зрозуміло, що не можна знайти ідеальне положення, у якому можна перебувати і працювати протягом усього робочого дня. Для більшості людей комфортабельним робочим місцем повинне бути таке, котре можна пристосувати не менш чим для двох позицій, при цьому положення крісла, монітора повинні щораз відповідати виконуваній роботі і звичкам. Багато хто вважає, що для роботи на комп'ютері більше підходять вертикальне і злегка похиле положення.

Дисплей. Положення тіла звичайно відповідає напрямку погляду; дисплеї, розташовані занадто низько чи під неправильним кутом, є основними причинами сутулості. Відстань від дисплея до очей може варіюватися в залежності від характеру виконуваної роботи - 40-70 см. При роботі з текстом відстань від екрана до очей повинна лише небагато перевищувати відстань між книгою й очима і складати 40- 45 см.

Необхідно давати відпочинок очам і час від часу їх просто закривати. Не можна допускати, щоб очі увесь час були сфокусовані на одній відстані, працюючи з текстом, рекомендується установлювати великий шрифт. Дуже важливо, щоб екран монітора не мерехтів. Частота регенерації зображення повинна бути не менш 72 Гц, тому що при більш низькій частоті помітне мерехтіння, хоча люди з особливо чуттєвим зором можуть помітити його і при більш високій частоті. Їм доведеться підібрати графічну плату і монітор з частотою регенерації 85 Гц і вище.

Рекомендується встановлювати на екран монітора спеціальні скляні поляризаційні фільтри з заземленням. Необхідно уникати того, щоб термінал був звернений екраном убік вікна, оскільки інтенсивна освітленість області зору може затопити потоками світла очі і розмити зображення на сітківці. Якщо приходиться сидіти поруч з вікном, то треба розташуватися під прямим

кутом до нього, причому екран дисплея повинний бути перпендикулярний шибці – цим виключаються відблиски на екрані.

Для зниження впливу низькочастотного електромагнітного випромінювання монітора на ЕЛТ рекомендується вибирати монітор, що задовольняє стандарту MPR II, чи TCO 99. У противному випадку необхідно установити на екран захисний фільтр, найбільш сучасні моделі, затримують до 99 % електромагнітного випромінювання.

Крісло необхідно установити на такій висоті, щоб не почувався тиск на куприк (крісло розташоване занадто низько) на стегна (крісло розташоване занадто високо). Фахівці з ергономіки, вважали, що кут між стегнами і хребтом повинний складати 90°, однак недавно проведені дослідження показали, що більшість людей переважно сидять трохи відкинувшись.

Клавіатуру необхідно установити так, щоб не треба було до неї тягтися і нахилятися вперед. При зміні положення тіла (наприклад, з вертикального на похиле) обов'язково треба перемінити положення клавіатури і дисплея. Може виявитися корисною регульована підставка клавіатури, завдяки якій можна без усякої напруги працювати з маніпулятором "миша". Можна поставити клавіатуру і на коліна. Руки при роботі повинні бути прямими в зап'ястях і зігнуті в ліктях приблизно під прямим кутом. Пальці також повинні бути злегка зігнуті. Удари по клавішах не повинні бути занадто сильними.

Зручна висота столу особливо важлива в тому випадку, коли на ньому розташовується клавіатура. Якщо в клавіатури немає підставки, а висоту столу не можна змінити (і він занадто високий), то треба вище підняти сидіння крісла, а під ноги підставити лавочку: чи що-небудь інше. Якщо стіл занадто низький, необхідно підкласти що-небудь під його ніжки.

Якщо при роботі часто приходиться дивитися на документи, треба установити підставку з оригіналом документа в одній площині і на одній, висоті з екраном. Якщо треба частіше дивитися на оригінал, чим на екран, необхідно повернути крісло й екран таким чином, щоб оригінал розміщався

прямо перед очима, а комп'ютер ледве збоку.

Щогодини необхідно робити перерву в роботі. У цей час рекомендується робити вправи для зап'ясть. Нижче описаний можливий комплекс вправ:

- покласти руку на край столу долонею вниз. Взявшись, за пальці, іншою рукою відвести кисть назад і утримувати протягом 5 секунд. Повторити для іншої руки;
- злегка упертися рукою в стіл і на 5 секунд напружити пальці в зап'ястя. Повторити іншою рукою;
- сильно стиснути пальці, а потім розпрямити їх. Виконання приведених рекомендацій дозволяє скоротити вплив шкідливих факторів на організм людини.

Відповідно до «Закону про охорону праці» роботодавець зобов'язаний забезпечити:

- безпеку працівників при експлуатації устаткування;
- застосовування засобів індивідуального захисту працівників;
- відповідні вимоги охорони праці, умови праці на кожному робочому місці; дотримання режиму праці і відпочинку працівників;
- навчання безпечним методам і прийомам виконання робіт;
- інструктаж з охорони праці;
- організацію контролю за станом умов праці на робочих місцях;
- проведення атестації робочих місць за умовами праці;
- інформування працівників про умови й охорону праці на робочих місцях, про існуючий ризик ушкодження здоров'я і компенсаціях при ушкоджені та засобах індивідуального захисту.

Вимоги до освітлення. У офісі комп’ютерної фірми, де експлуатуються комп'ютери, штучне освітлення повинне бути загальним і рівномірним. Однак якщо співробітники переважно працюють з документами, то допускається застосувати комбіноване освітлення: крім загального установлюються світильники місцевого освітлення, які не повинні

створювати відблисків на поверхні екрана і збільшувати його освітленість більш 300 лк. Освітленість поверхні столу в зоні розміщення робочого документа повинна становити 300-500 лк.

Джерела освітлення варто встановлювати таким чином, щоб вони не осліплювали, при цьому яскравість світних поверхонь (вікна, світильники та ін.), які розташовуються в полі зору, повинна бути не більш 200 кд/м^2.

В якості джерела світла при штучному освітленні повинні застосовуватися переважно люмінесцентні лампи типу ЛБ. При пристрої відбитого освітлення допускається застосування металогалогенних ламп потужністю до 250 Вт, а у світильниках місцевого освітлення – ламп накалювання.

Для забезпечення нормованих значень освітленості в офісі необхідно не рідше двох разів на рік чистити скло, віконні рами і світильники та вчасно заміняти перегорілі лампи.

Робочі місця повинні розташовуватися таким чином, щоб природне світло падало збоку, переважно ліворуч.

Для внутрішньої обробки приміщень повинні використовуватися дифузно-відбиваючі матеріали, з коефіцієнтом відбиття від стелі – 0,7-0,8; для стін – 0,5-0,6; для підлоги – 0,3-0,5. Полімерні матеріали для внутрішньої обробки повинні бути дозволені для застосування органами й установами Держсанепіднагляду.

Поверхня підлоги в приміщеннях повинна бути рівною, без вибоїн, неслизькою, зручною для очистки і вологого прибирання, мати антистатичні властивості.

Вимоги до кондиціонування. У офісі комп’ютерної фірми мікроклімат повинен відповідати наступним санітарним нормам:

- температура повітря в теплий період року – не більш 230-250 С, у холодний – 220-240 С;
- відносна вологість повітря – 40-60%;
- швидкість руху повітря – 0,1м/с.

Для підвищення вологості повітря в приміщеннях варто застосовувати зволожувачі повітря, щодня заправляти їх дистильованою або кип'яченою водою.

Вимоги до розміщення, оснащення й організації робочих місць. Відстань між робочими столами з моніторами (у напрямку тилу поверхні одного монітора та екрана іншого) повинне бути не менш 2 м, а між бічними поверхнями моніторів – не менш 1,2 м.

Віконні отвори повинні бути обладнані регульованими жалюзями, завісами, зовнішніми навісами та ін.

Бажано, щоб висоту робочої поверхні столу можна було регулювати в межах 680-800 мм, а при відсутності такої можливості вона повинна дорівнювати 725 мм. Модульними розмірами робочої поверхні комп'ютерного столу, на базі яких розраховують конструктивні розміри, варто вважати: ширину 800, 1000, 1200 і 1400 мм; глибину 800 і 1000 мм.

Екран монітора повинен знаходитись від очей користувача на оптимальній відстані 600-700 мм, але не ближче 500 мм з урахуванням розмірів алфавітно-цифрових знаків і символів.

Вимоги до безпеки під час експлуатації й обслуговування ЕОМ. У приміщенні з комп'ютерами повинно проводитися щоденне вологе прибирання. Офіс повинен оснащуватися аптечкою першої допомоги та вуглекислотними вогнегасниками.

Режими праці і відпочинку. Режими праці і відпочинку при роботі на комп'ютерах залежать від виду і категорії трудової діяльності.

При восьмигодинній робочій зміні і роботі на комп'ютері регламентовані перерви варто встановлювати:

- для I категорії робіт – через 2 години від початку робочої зміни і через 2 години після обідньої перерви тривалістю по 15 хв.;
- для II категорії робіт – через 2 години від початку робочої зміни і через 1,5 - 2 години після обідньої перерви тривалістю по 15 хв. або через кожну годину роботи тривалістю по 10 хв.

Під час регламентованих перерв із метою зниження нервово-емоційної напруги, зменшення стомлення очей, усунення гіподинаміки і гіпокінезії доцільно виконувати комплекси вправ, викладених у санітарних нормах.

7 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

7.1 Основні положення охорони навколишнього середовища

Під охороною навколишнього середовища розуміють сукупність міжнародних, державних і регіональних правових актів, інструкцій і стандартів, які доводять загальні юридичні вимоги до кожного конкретного забруднювача і забезпечують його зацікавленість у виконанні цих вимог, конкретних природоохоронних заходів по втіленню в життя цих вимог.

Тільки якщо всі ці складові частини відповідають один одному за змістом і темпами розвитку, тобто складаються в єдину систему охорони навколишнього природного середовища, можна розраховувати на успіх.

Оскільки не була вирішена вчасно завдання охорони природи від негативного впливу людини, тепер все частіше постає завдання захисту людини від впливу змінилася природного середовища. Обидва ці поняття інтегруються в терміні «охорона навколишнього (людини) природного середовища».

Охорона навколишнього природного середовища складається з:

- правової охорони, що формулює наукові екологічні принципи у вигляді юридичних законів, обов'язкових для виконання;
- матеріального стимулювання природоохоронної діяльності з ціллю зробити її економічно вигідною для підприємств;
- інженерної охорони, що розробляє природоохоронну та ресурсозберігаючу технологію і техніку.

Основними принципами охорони навколишнього природного середовища є:

- пріоритет забезпечення сприятливих екологічних умов для життя, праці та відпочинку населення;
- науково обґрунтоване поєднання екологічних і економічних інтересів суспільства;

- облік законів природи і можливостей самовідновлення і самоочищення її ресурсів;

- недопущення незворотних наслідків для охорони природного середовища і здоров'я людини;

- право населення та громадських організацій на своєчасну і достовірну інформацію про стан навколишнього середовища та негативному впливі на неї і на здоров'я людей різних виробничих об'єктів;

- невідворотність відповідальності за порушення вимог природоохоронного законодавства.

7.2 Забруднення навколишнього середовища ПК. Заходи щодо зменшення забруднення навколишнього середовища

В останні 2-3 роки тема екологічної відповідальності виробників комп’ютерних технологій є однією з найбільш часто обговорюваних в ЗМІ, наукових і бізнес-конференціях, виставках.

На прикладі сучасного стану комп'ютерних технологій, їх активного раз-розвитку в контексті націленості на екологічну заклопотаність, можна проаналізувати, як реалізуються на практиці принципи екологічної етики.

Розробка та впровадження так званих «Зелених технологій» – Green IT направлена на вирішення цілого комплексу проблем, обумовлених екологічними проблемами і активною пропагандою пошуку оптимальних шляхів їх вирішення: від утилітарного зниження матеріальних витрат на виробництво і використання до індивідуальної відповідальності рядового споживача перед природою та майбутніми поколіннями.

Що таке зелені технології в комп'ютерній сфері? Сьогодні під « зеленим» розуміється такий комп'ютер, чий вплив на навколишнє середовище значно зменшено порівняно зі звичайними. В ідеалі «зелений» комп'ютер повинен забезпечувати цілий спектр завдань, що дозволяють знизити шкідливий вплив. Тут і відсутність серед комплектуючих токсичних

речовин, низький рівень електромагнітних випромінювань і шуму, забезпечення енергоефективності комп'ютера і зниження вироблюваного ним тепла, споживаної електроенергії, легкість утилізації та можливість переробки окремих компонентів.

Відповідно, «зелені» технології повинні забезпечити умови для випуску подібних пристроїв, а також взагалі дати можливість для зміни виробничого циклу з точки зору екологічності, оскільки саме виробництво може нашкодити навколишньому середовищу в більшій мірі, ніж випущені продукти .

Отже, розглянемо основні проблеми, що стимулюють виникнення і розвиток Green IT .

По-перше, найбільш важливою заклопотаністю розробників є створення нових продуктів, що дозволяють економити енергоспоживання, в більш кращому варіанті використовувати для своїх продуктів поновлювані джерела енергії.

Мотивація до розробки та впровадження енергозберігаючих комп'ютерних технологій спирається на прагматичний підрахунок витрачених коштів на оплату споживаної електроенергії. Як основний аргумент наводиться масштаб і кількість енергоємних установок, починаючи з персонального комп'ютера в будинку, закінчуючи великими дата-центрами, що споживають велику кількість електроенергії , при цьому виділяється велика кількість теплоти, на зниження якої також використовуються додаткові енергоресурси.

По-друге, при роботі будь-якого комп'ютера неминучий високий рівень шуму. Ця проблема, на перший погляд, не становить загрози для навколишнього середовища. Однак, необхідність створення і роботи великих серверів і глобальних дата-центрів є серйозною причиною безпосередньо негативного впливу на довкілля, тварин, людини.

У зв'язку з цим в даний час розробляються нові екологічні безпечні проекти по створенню великих серверів і дата-центрів, що завдають мінімальну шкоду людині і навколишньому середовищу.

Пошуковий гігант Google має намір розмістити свої дата-центри на баржах в 11 кілометрах від берегової лінії. Ініціатори проекту вважають, що настільки масштабне рішення дозволить не тільки заощадити величезні кошти, а й залучити екологічно чисті джерела енергії. Так, знаходження дата-центрів в офшорній зоні позбавить корпорацію від податків на нерухомість, а необхідна для них електроенергія вироблятиметься автономно за рахунок руху хвиль і сили вітру. Як заявляють представники компанії, природне ефективне охолодження вузлів дата-центрів забезпечать спеціальні насоси.

По-третє, існує ще одна проблема, суть якої зводиться до різних PR-акцій, інформаційним потокам і маркетингових ходам, пропагуючим Green IT. Кроки, що робляться виробниками комп'ютерів і комплектуючих, можна розцінювати як бутафорію, спрямовану на залучення покупців, яким не чуже почуття турботи про природу. Можливо, частково це дійсно так. Зараз немає достатніх підстав міркувати про те, помітний ефект екологічних проектів комп'ютерних виробників, мало часу пройшло з моменту впровадження перших стандартів і початку випуску перших екологічно чистих технологічних продуктів. Однак, без сумніву, очевидний той факт, що в гонитві за потужністю, швидкістю, ефективністю і дешевизною продукції багато виробників забули, що матеріальна вигода в цьому випадку небезпечна для здоров'я. Тому перший поштовх до розробки і просування Green IT – це констатація турботи про здоров'я споживачів, що в кінцевому підсумку виражається в турботі про природу .

По-четверте, ще одна проблема, обумовлена об'єктивними законами розвитку техніки – швидке старіння та утилізація комп'ютерів. Гордон Мур в 1965 році сформулював знаменитий закон (кількість транзисторів в кристалі мікропроцесора подвоюється кожен рік), який визначив бурхливий розвиток і зростання продуктивності комп'ютерів. Виробництва в гонитві за

досконалими технологіями та прибутками переходять на більш досконалі технологічні процеси, ціни знижуються, потужності і продуктивність промислових і персональних комп'ютерів збільшується, але неминуче прискорюється процес їх старіння за різними параметрами: від розмірів і ваги, до потужності і швидкості. Згідно з даними Інтернет-сайтів, середньостатистичний англієць за все своє життя викидає близько 3 тонн електронного мотлоху. Якщо розрахувати, скільки подібного сміття виробляє все населення Великобританії за середні 70 років життя, то вийде гора масою в 590 Лондонських Тауерів!

У цьому контексті виникає дилема – використовувати застаріваюче обладнання і не піднімати рівень забруднення навколишнього середовища, жертвуючи тим самим продуктивністю і ККД або продовжувати розвивати технології , викидати застарілі машини, створювати нове, ще більш досконале, не замислюючись про шкоду, вважаючи, що до того моменту, коли рівень забруднення виявиться вище допустимих норм, буде створена універсальна утилізуюча технологія .

По-п'яте, «отрута високих технологій», проблема зниження токсичності. Тут гострота проблеми становиться очевидною кожному користувачеві і ще більш очевидна фахівцеві. Локалізація технологічних відходів, під якою розуміють звалища, або полігони, – поняття дуже і дуже умовне, не вирішальна проблеми.

Як відомо, пластмаса, яка використовується у виробництві комп'ютерів, не «переробляється» природою відразу. Викинутий на смітник старий системний блок може пролежати до повного розкладання і сто, і двісті років. Скло електронно-променевої трубки дисплея протримається в сирій землі близько тисячоліття, а в більш сприятливих умовах і довше, до цілого мільйона років!

До того ж практично завжди у виробництві компонентів для ЕОМ використовуються важкі метали, такі як хром, ртуть і свинець, концентрацію таких речовин мікроскопічно велика. Наприклад, в корпусах 14 і 15 –

дюймових моделей ЕПТ – моніторів, які в масовому порядку відправляються на звалище, що витісняються своїми більш досконалими рідкокристалічними колегами, загальна кількість шкідливих речовин становить 25 відсотків від маси. Перебуваючи під сонцем, токсини починають випаровуватися в атмосферу (за останні 100 років вміст, наприклад, ртутних парів підвищилося в три рази), повільно, але вірно отруюють грунт, а значить, і грунтові води, які виносять шкідливі речовини в річки.

І останнє, шосте, у кожної проблеми, які б способи рішення не представлялися, є наслідки, часто негативні, що не дозволяють говорити про наявність універсальних методів.

Екологічна орієнтованість і етична обумовленість багатьох проблем у виробництві та впровадженні Green IT незважаючи ні на що має позитивні наслідки, створені завдяки розвитку інформаційних технологій. Загальна інформованість про проблеми екологічної загрози, пропаганда ідей соціальної відповідальності бізнесу та індивідуальної практики малих справ, маркетингові проекти , що спираються на потреби сучасного суспільства, дозволяють створити умови для поступового просування ідей і принципів екологічної етики.

Очевидно, що IT– галузь повинна знизити вплив швидкого зростання інформації на навколишнє середовище, забезпечивши більш високі рівні ефективності.

Три основні технологічні ініціативи допомагають вирішити цю задачу:

– Віртуалізація та консолідація
– Управління життєвим циклом інформації
– Дедуплікація даних

Віртуалізація та консолідація необхідні для скорочення енергоспоживання в ЦОД, в яких IT-менеджери встановлюють все більше систем для підвищення продуктивності, надмірності та доступності, не думаючи про ефективність живлення або охолодження. Віртуалізація вирішує проблему неефективності, відокремлюючи ПЗ від базового обладнання, що

забезпечує роботу декількох ОС і додатків на одному комп'ютері. У свою чергу, більш високий коефіцієнт використання серверів і сховищ означає розгортання меншої кількості машин, які споживають менше електроенергії для живлення та охолодження.

Управління життєвим циклом інформації (ILM) ґрунтується на тій передумові, що цінність інформації змінюється з часом. ILM використовує автоматичний аналіз для зберігання інформації в найбільш підходящому енергетично ефективному сховище на всіх етапах її життєвого циклу. Наприклад, оперативні дані і дані, важливі для роботи бізнесу, потребують систем з максимальною надійністю і продуктивністю, які вимагають більше ресурсів і потужності. Коли важливість інформації знижується, ILM переносить її в сховище з меншим енергоспоживанням.

Дедуплікація даних значно знижує обсяг зберігаються резервних копій, обумовлений тим, що користувачі зберігають по кілька копій і варіацій одного файлу в різних точках мережі. Дедуплікація зупиняє нестримне дублювання даних шляхом перетворення файлів у сегменти даних, які можуть зберігатися і використовуватися в декількох файлах. Резервне копіювання вихідного файлу на центральний сервер виконується тільки один раз. Зміни файлу відправляються на сервер у вигляді нових і унікальних сегментів даних , пов'язаних з оригіналом. Резервне копіювання виконується тільки для цих нових сегментів. Дедуплікація скорочує потреби в смузі пропускання мережі і обсязі резервного сховища в 300 разів.

Створення екологічної стійкості – масштабне підприємство. Воно вимагатиме глобального співробітництва і спільної роботи між найбільшими і впливовими зацікавленими сторонами – ООН, державними та приватними установами, компаніями, університетами та приватними особами. Воно вимагатиме системного погляду на вироблення електроенергії, енергоспоживання і зміна клімату, а також комбінації стимулів, стандартів, грантів, безперервних досліджень, уяви та інновацій, що сприяють глобальним змінам, які необхідні якомога швидше.

ВИСНОВКИ

Мета, яка ставилася на початку виконання кваліфікаційної роботи, була досягнута. Тобто була підвищена достовірність оцінювання трудомісткості розробки скраперів на Java за рахунок удосконалення нелінійної регресійної моделі.

Для досягнення поставленої мети були виконані наступні завдання:

- проведений аналіз існуючих методів та моделей оцінювання трудомісткості розробки програмного забезпечення;
- удосконалена математична модель для оцінювання трудомісткості розробки скраперів на Java;
- розроблене програмне забезпечення для реалізації удосконаленої моделі.

Програмне забезпечення, яке розроблене в ході виконання кваліфікаційної роботи, дозволить автоматизувати процес оцінювання трудомісткості розробки скраперів на Java та допоможе зменшити час, необхідний на ці підрахунки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Голованова, М.А. Оценка трудоемкости работ на ранних стадиях создания программного обеспечения [Текст] / М.А. Голованова, Є.В. Надін // Системи обробки інформації. – Харків, 2014. – № 8 (124). – С.151-156.

2. Boehm, B.W. Software engineering economics [Текст] / B.W. Boehm. – Prentice-Hall, 1981. – 320 p.

3. Boehm, B.W. The COCOMO 2.0 Software Cost Estimation Model / B.W. Boehm. – American Programmer, 2000. – 586 p.

4.Москаленко, А. А.Разработка приложения веб-скрапинга с возможностями обхода блокировок [Текст] / А.А. Москаленко, О.Р. Лапонина, В.А. Сухомлин // Современные информационные технологии и ИТ-образование. 2019. Т. 15, №2. – С. 413-420.

5. Кульдин, С.П. Генетический подход к проблеме оценки сроков и трудоемкости разработки программного обеспечения с заданными требованиями к качеству [Текст] / С.П. Кульдин // Прикладная информатика. – 2010. – №5(29). – С.30-42.

6. Coates, J. Technological Forecasting and Social Change [Текст] / J.Coates. – Elsevier Science Inc, 1999. – 235 p.

7. Johnson, Kim. Software cost estimation – Metrics and models [Текст] / Johnson Kim. – University of Calgary, 2001. – 115 p.

8. Макконнелл, С. Сколько стоит программный проект [Текст] / С. Макконнелл. – СПб.: Питер, 2007. – 297 с.

9. Цуман, Д.С. Перевірка якості регресійної моделі оцінювання трудомісткості розробки веб-скраперів на Java [Текст] / Д.С. Цуман, Л.О. Латанська // Всеукраїнська науково-практична інтернет конференція «ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ: МОДЕЛІ, АЛГОРИТМИ, СИСТЕМИ (ITMAS – 2020)» (м. Миколаїв, 26-28 жовтня 2020). – С. 46-47.

10 Malathi, S. et al (2012). Analysis of size metrics and effort performance criterion in software cost estimation. Indian Journal of Computer Science and Engineering, 3 (1). – P.24-31.

11 Гмурман, В.Е. (2010). Теория вероятностей и математическая статистика. Учеб. пособие. М.: Высшее образование.

12. Прикладна економетрика : навч. посіб. : у двох частинах. Частина 1 / Л. С. Гур'янова, Т. С. Клебанова, С. В. Прокопович та ін. – Харків: ХНЕУ ім. С. Кузнеця, 2016. – 235 с.

13. Регресійний аналіз [Електронний ресурс]. – Режим доступу: https://pidruchniki.com/17280924/ekonomika/regresiyniy_analiz – Назва з екрану

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

15. Расстояние Махаланобиса [Електронний ресурс]. – Режим доступу: http://statistica.ru/theory/rasstoyanie-makhalonobisa/ – Назва з екрану

16. Приходько С.Б. Методичні вказівки до виконання лабораторних робіт з дисципліни "Обробка експериментальних даних на ЕОМ" [Текст] / С.Б. Приходько. – Миколаїв: НУК , 2005. – 52 с.

17. Дрейпер, Н. Прикладной регрессионній 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.

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

19. 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.

ДОДАТОК А

Технічне завдання на розробку програмного забезпечення для оцінювання трудомісткості розробки скраперів на Java

Вступ

Назва програми: «Trydoemkost».

Область застосування:

Програма призначена для оцінювання трудомісткості розробки скраперів на Java.

1 Підстави для розробки

Підставою для розробки програмного продукту є завдання на кваліфікаційну роботу «Математична модель для оцінювання трудомісткості розробки скраперів на Java та створення програмного забезпечення для її реалізації»

2 Призначення програми

2.1 Функціональне призначення

Даний програмний продукт призначений для оцінювання трудомісткості розробки скраперів на Java.

2.2 Експлуатаційне призначення

Програма може використовуватися для оцінювання трудомісткості проектів в компаніях та командах програмістів, що займаються розробкою скраперів на Java.

3 Вимоги до програми

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу функцій, що виконуються

Програма повинна забезпечувати можливість виконання наведених нижче функцій:

– внесення даних про проекти (кількість строк коду, трудомісткість);

– виконання нормалізації внесених даних за допомогою натурального логарифму;

– побудова лінійної регресії для нормалізованих даних;

– підрахунок довірчого інтервалу для нормалізованих даних;

– підрахунок інтервалу передбачення для нормалізованих даних;

– отримання нелінійного регресійного рівняння за допомогою зворотного перетворення;

– підрахунок довірчого інтервалу для нелінійного рівняння;

– підрахунок інтервалу передбачення для нелінійного рівняння;

– побудова лінійної регресії для ненормалізованих даних при припущенні про нормальність вихідних даних;

– підрахунок довірчого інтервалу для лінійної регресії;

– підрахунок інтервалу передбачення для лінійної регресії;

– виведення графіків довірчих інтервалів і інтервалів передбачення;

– підрахунок метрик якості.

3.1.2 Вимоги до організації вхідних та вихідних даних

Вхідні дані мають бути організовані у текстових файлах. Внесення вхідних даних виконується шляхом вибору шляху до відповідного текстового файлу у спеціальному вікні.

Вихідні дані (довірчі інтервали, інтервали передбачення та метрики якості) повинні виводитися в відповідних місцях на створеній програмою формі.

3.2 Вимоги до надійності

3.2.1 Вимоги до забезпечення надійності функціонування програми

Надійне (стійке) функціонування програми має бути забезпечено надійним (стійким) функціонуванням операційної системи.

3.2.2 Відмова виконання програмного продукту через некоректні дії користувача.

Відмова програми можлива внаслідок некоректних дій користувача при взаємодії з операційною системою. Щоб уникнути виникнення відмов програми за вказаною вище причини необхідно забезпечити стабільне функціонування операційної системи.

3.4 Умови експлуатації

Кліматичні умови експлуатації програмного продукту відповідають умовам експлуатації апаратної частини персонального комп'ютера.

3.5 Вимоги до інформаційної та програмної сумісності

Системні програмні засоби (не вільно поширювані), використовувані програмою, повинні бути представлені ліцензійною версією ОС Windows.

3.6 Вимоги до захисту інформації та програм

Вимоги до захисту інформації та програм відсутні.

3.7 Вимоги до маркування та упаковки

3.7.1 Вимоги до маркування

Вимоги до маркування відсутні.

3.7.2 Вимоги до упаковки

Вимоги до упаковки відсутні.

3.8 Вимоги до транспортування та зберігання

Вимоги до транспортування програми відсутні.

4 Вимоги до програмної документації

Склад програмної документації:

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

5 Техніко-економічні показники

Економічна ефективність впровадження програмного забезпечення розрахована у розділі 5. Згідно з отриманими результатами впровадження даного ПЗ є доцільним, а термін його окупності становить приблизно 4 місяці.

6 Стадії та етапи розробки програмного забезпечення

Стадії та етапи розробки програми для оцінювання трудомісткості розробки скраперів на Java, зазначені в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки програмного забезпечення

Стадії розробки	Етапи робіт	Вміст робіт	Контрольні точки
1	2	3	4
Технічне завдання	Обумовлення необхідності розробки	Постановка задачі. Збір початкових матеріалів.	З 08.09.2020 до 09.09.2020
	Розробка та затвердження технічного завдання	Визначення вимог до ПЗ. Обговорення та затвердження технічного завдання	З 10.09.2020 до 01.10.2020

Продовження табл. А.1

1	2	3	4
Ескізний проект	Розробка ескізного проекту	Попередня розробка структури вхідних та вихідних даних, уточнення методів рішення завдань, загальний опис алгоритму	З 02.10.2020 до 10.11.2020
	Затвердження ескізного проекту	Розробка пояснювальної записки. Узгодження і затвердження ескізного проекту	З 11.11.2020 до 18.11.2020
Технічний проект	Розробка технічного проекту	Уточнення структури вхідних та вихідних даних, розробка алгоритму рішення задачі, розробка структури ПЗ	З 19.11.2020 до 25.11.2020
	Затвердження технічного проекту	Розробка пояснювальної записки. Узгодження і затвердження технічного проекту	З 26.11.2020 до 28.11.2020
Робочий проект	Розробка ПЗ	Програмування та налагодження програми	З 29.11.2020 до 10.12.2020
	Розробка програмної документації	Розробка програмної документації відповідно до вимог ГОСТ 19.101-77	З 11.12.2020 до 14.12.2020
	Випробування програми	Розробка, узгодження та затвердження програми і методики випробувань, коригування програми і програмної документації за результатами випробувань	З 15.12.2020 до 17.12.2020

7 Порядок контролю та приймання

Контроль здійснюється замовником на кожній стадії розробки ПЗ. Приймання здійснюється замовником за програмою та методикою випробувань. Випробування проводиться в зазначений термін.

Хід проведення приймально-здавальних випробувань документується за допомогою протоколу проведення випробувань. На підставі протоколу проведення випробувань виконувач сумісно з замовником підписують акт приймання-здачі програми в експлуатацію.

ДОДАТОК Б

Опис програми для оцінювання трудомісткості розробки скраперів на Java

1 Загальні відомості

1.1 Позначення і найменування програми

Найменування: Програмне забезпечення для оцінювання трудомісткості розробки програмного забезпечення на Java «Trydoemkost».

Позначення: «Trydoemkost».

1.2 Програмне забезпечення, необхідне для функціонування програми

Програмне забезпечення «Trydoemkost» є крос-платформним і основною необхідною вимогою для функціонування виробу є наявність .NET Framework (4.5 і вище версії).

1.3 Мови програмування

ПЗ «Trydoemkost» реалізовано на мові високого рівня C#. Модулі, реалізовані на інших мовах програмування, не використовуються.

2 Функціональне призначення

2.1 Класи вирішуваних завдань і призначення програми

ПЗ «Trydoemkost» повинно забезпечувати можливість виконання нижче перерахованих функцій:

– внесення даних про проекти (кількість строк коду, трудомісткість);

– виконання нормалізації внесених даних за допомогою натурального логарифму;

– побудова лінійної регресії для нормалізованих даних;

– підрахунок довірчого інтервалу для нормалізованих даних;

– підрахунок інтервалу передбачення для нормалізованих даних;

– отримання нелінійного регресійного рівняння за допомогою зворотного перетворення;

– підрахунок довірчого інтервалу для нелінійного рівняння;

– підрахунок інтервалу передбачення для нелінійного рівняння;

– побудова лінійної регресії для ненормалізованих даних при припущенні про нормальність вихідних даних;

– підрахунок довірчого інтервалу для лінійної регресії;

– підрахунок інтервалу передбачення для лінійної регресії;

– виведення графіків довірчих інтервалів і інтервалів передбачення;

– підрахунок метрик якості.

2.2 Функціональні обмеження

Файли повинні бути формату EXE.

3 Використовувані технічні засоби

ПЗ «Trydoemkost» призначене для використання на персональних комп'ютерах, що працюють під управлінням наступних операційних систем: ОС MS Windows XP/7/8/8.1/10

Для ПЗ «Trydoemkost» пред'являються такі апаратні вимоги до ПЕОМ:

– оперативна пам'ять 512 МБ;

– дисковий простір - не менше 512 МБ вільного місця на диску;

– роздільна здатність екрану - 1280 x 1024 пікселів;

– клавіатура 101/102-х клавішна рус / лат;

Швидкість роботи ПЗ «Trydoemkost» на конкретному комп'ютері залежить також від характеристик окремих його комплектуючих (процесора, оперативної пам'яті і ін.).

4 Виклик і завантаження

Запуск ПЗ «Trydoemkost» в графічному режимі здійснюється двома способами:

1) за допомогою файлу запуску: Trydoemkost.exe;

2) з використанням командного рядку: Trydoemkost.exe.

5 Вхідні і вихідні дані

Вхідні дані програми(строки коду та трудомісткість) повинні знаходитись в окремому текстовому файлі у два стовбці. Вихідні дані програми мають відображатися на екрані користувача.

ДОДАТОК В

Програма і методика випробувань програмного забезпечення для оцінювання трудомісткості розробки скраперів на Java

1 Об’єкт випробувань

Об’єктом випробувань є програмне забезпечення для оцінювання трудомісткості розробки скраперів на Java.

2 Мета випробувань

Перевірка функціональних можливостей програмного забезпечення на відповідність вимогам технічного завдання.

3 Вимоги до програми

В ході тестування програмного продукту необхідно перевірити, що розроблене ПЗ реалізує наступні функції:

– внесення даних про проекти (кількість строк коду, трудомісткість);

– виконання нормалізації внесених даних за допомогою натурального логарифму;

– побудова лінійної регресії для нормалізованих даних;

– підрахунок довірчого інтервалу для нормалізованих даних;

– підрахунок інтервалу передбачення для нормалізованих даних;

– отримання нелінійного регресійного рівняння за допомогою зворотного перетворення;

– підрахунок довірчого інтервалу для нелінійного рівняння;

– підрахунок інтервалу передбачення для нелінійного рівняння;

– побудова лінійної регресії для ненормалізованих даних при припущенні про нормальність вихідних даних;

– підрахунок довірчого інтервалу для лінійної регресії;

– підрахунок інтервалу передбачення для лінійної регресії;

– виведення графіків довірчих інтервалів і інтервалів передбачення;

– підрахунок метрик якості.

4 Вимоги до програмної документації

Склад програмної документації, що надається на випробування:

– технічне завдання;

– текст програми;

– опис програми;

– програма і методика випробувань;

– інструкція користувача.

5 Засоби і порядок випробувань

Випробування проводилось на платформі Windows 7 32bit.

Порядок випробувань:

– встановити і запустити програму;

– перевірити можливості додавання даних до програми;

– перевірити можливість продивитися довірчі інтервали та інтервали передбачення;

– перевірити можливість сховати інтервали;

– перевірити можливість побудови графіків довірчих інтервалів та інтервалів передбачення;

– перевірити можливість розрахувати метрики якості;

– перевірити неможливість ввести файл з даними неправильного формату;

6 Методика випробувань

– Додавання даних до програми:

- Натиснути на кнопку «Вибрати файл».

- Вибрати шлях до файлу.
- Файл відкривається і поблизу кнопки з’являється назва файлу

Очікуваний результат: після натискання кнопки «Вибрати файл» файл додається.

– Перегляд довірчих інтервалів та інтервалів передбачення

- Прочитати дані з файлу.
- Натиснути на кнопку «Показати інтервали».
- Інтервали зявляються на екрані у вигляді таблиці.
- З’являється текст «Сховати інтервали» замість «Показати інтервали».

Очікуваний результат: після натискання кнопки «Показати інтервали» на головній формі додається таблиця, яка містить ці інтервали.

– Введення файлу з даними неправильного формату

- Ввести дані неправильного формату.
- З’являється повідомлення про неправильний формат даних.

Очікуваний результат: програма видає повідомлення про неправильний формат даних.

ДОДАТОК Г

Інструкція користувача програмного забезпечення для оцінювання трудомісткості розробки скраперів на Java

1. Для того, щоб запустити програму, потрібно запустити файл «Trydoemkost.exe». При запуску програми з’являється початкова форма (рис. Д.1).

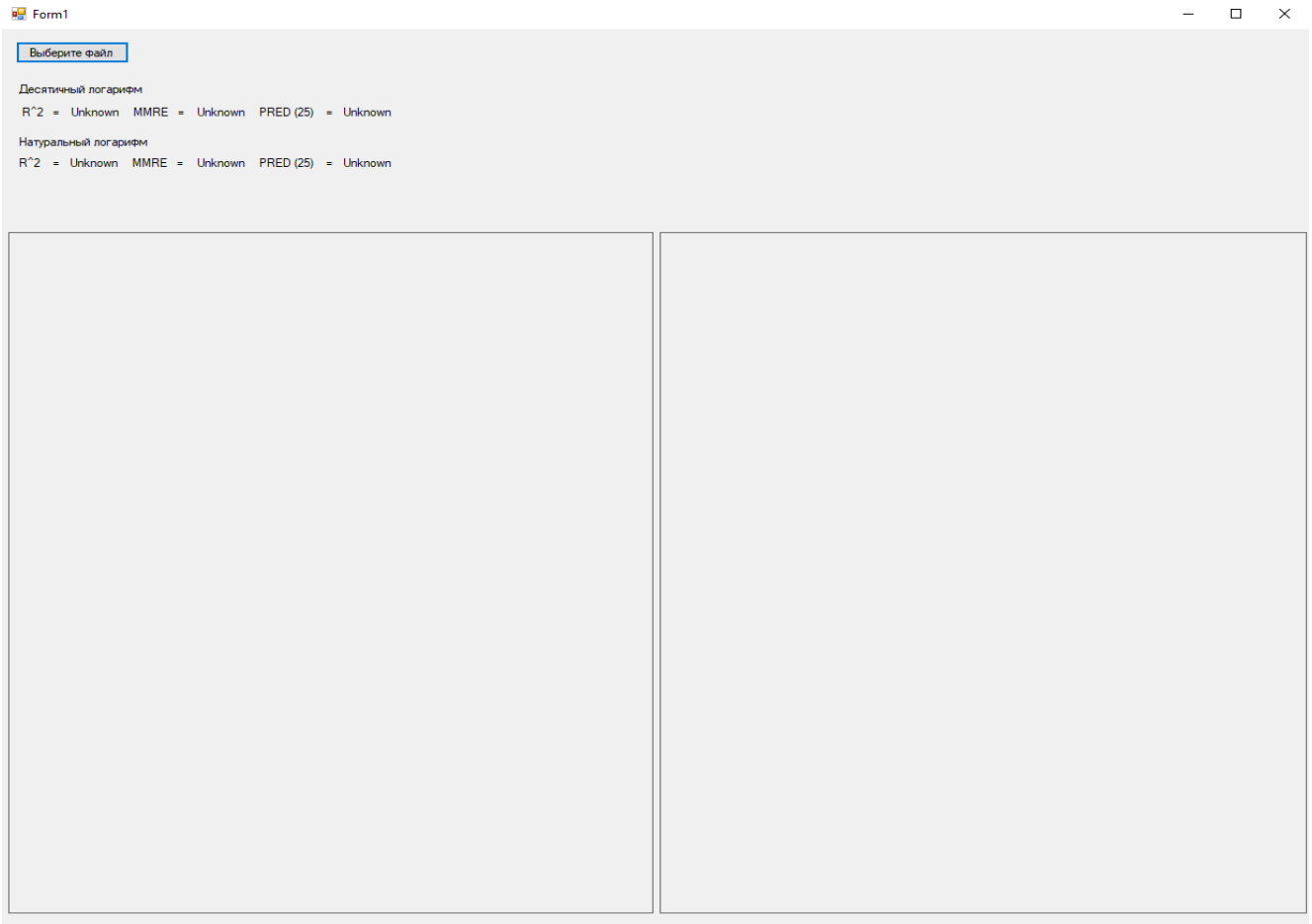

Рисунок Д.1 – Головна форма

2. Для внесення інформації про даних потрібно натиснути на кнопку «Вибрати файл» (рис. Д.2).

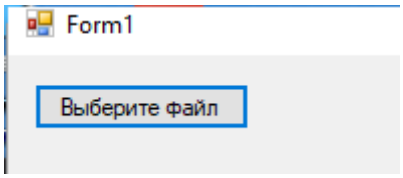

Рисунок Д.2 – Додавання даних

3. Потім необхідно у відкрившимуся вікні знайти файл з даними та запустити його.

4. Після цього на формі з’являються графіки побудовані на основі порахованих довірчих інтервалів і інтервалів передбачення (рис. Д.3)

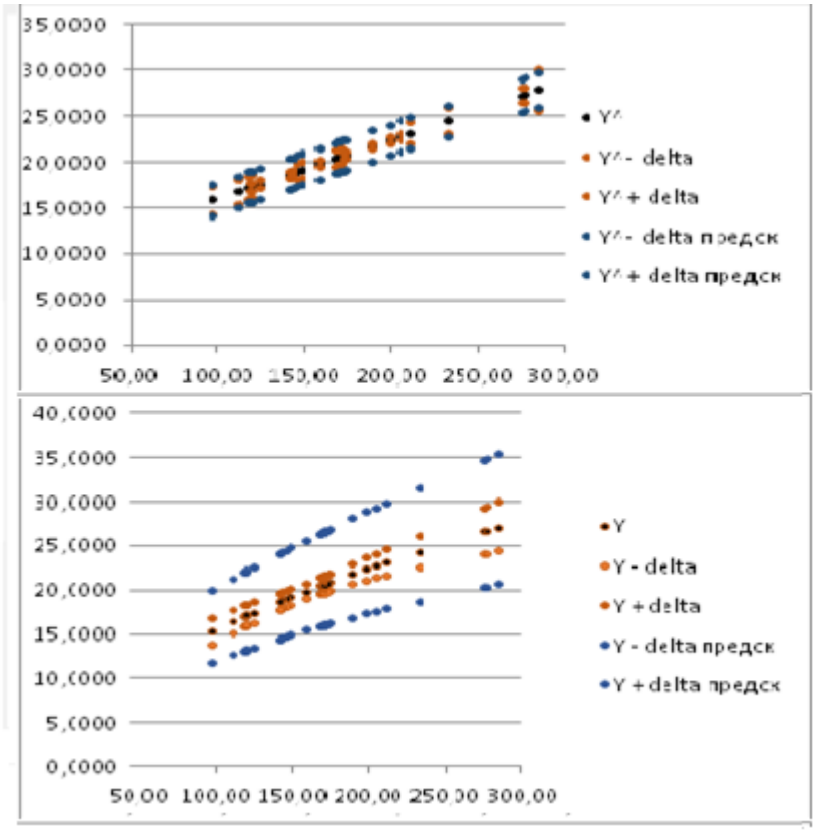

Рисунок Д.3 – Графіки інтервалів

5. Також з’являються пораховані метрики якості (рис. Д.4).

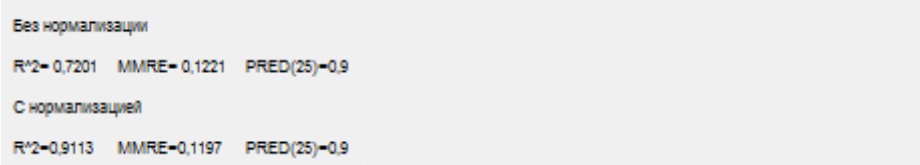

Рисунок Д.4 – Метрики якості

6. Для відображення порахованих інтервалів необхідно натиснути кнопку «Показати інтервали» (рис. Д.5).

Рисунок Д.5 – Кнопка «Показати інтервали»

7. Для того щоб сховати пораховані інтервали необхідно натиснути на кнопку «Сховати інтервали» (рис. Д.6)

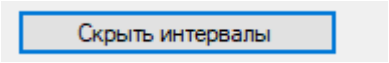

Рисунок Д.6 – Кнопка «Сховати інтервали»

8. При неправильному форматі введених даних програма показує повідомлення про помилку (рис. Д.7)

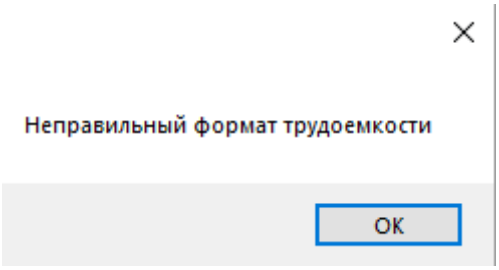

Рисунок Д.7 – Повідомлення про помилку

ДОДАТОК Д

Текст програми для оцінювання для оцінювання трудомісткості розробки скраперів на Java

Oproximator.cs

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WindowsFormsApp1
{
    class Regres
    {
        private double a;
        private double b;

        public Regres(double[] x, double[] y, bool check )
        {
            double sumx = 0;
            double sumy = 0;
            double sumx2 = 0;
            double sumxy = 0;

            int n = x.Length;

            for (int i = 0; i < n; i++)
            {
                sumx += x[i];
                sumy += y[i];
                sumx2 += x[i] * x[i];
                sumxy += x[i] * y[i];
            }

            a = (n * sumxy - sumx * sumy) / (n * sumx2 - sumx * sumx);
            b = (sumy - a * sumx) / n;
            if (check == true)
```

```
            {
                b += 0.025;
            }
        }

        public double getA()
        {
            return a;
        }

        public double getB()
        {
            return b;
        }

    }
}
```

FromFile.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Diagnostics;

namespace WindowsFormsApp1
{
    class FromFile
    {
        private double[] x;
        private double[] y;
        private string path;

        public void read()
        {
            using (var reader = new StreamReader(path))
            {
                string row;
                int i = 0;

                var lineCount = File.ReadLines(path).Count();

                x = new double[lineCount];
                y = new double[lineCount];

                while ((row = reader.ReadLine()) != null)
                {
                    try
                    {
```

```
                        x[i] = Double.Parse(row.Split(new[] { ' ' 
                    }, StringSplitOptions.RemoveEmptyEntries)[0]);

                }
                catch (Exception e)
                {
                        throw new Exception("Неправильный формат 
                    количества строк кода");
                }

                if (x[i] <= 0)
                {
                    throw new Exception("Количество строк ода меньше 
                      или равно 0");
                }

                try
                {
                         y[i] = Double.Parse(row.Split(new[] { ' ' 
                    },
                    StringSplitOptions.RemoveEmptyEntries)[1].TrimSt
                    art());

                }
                catch (Exception e)
                {
                    throw new Exception("Неправильный формат 
                     трудоемкости");
                }

                if (y[i] <= 0)
                {
                        throw new Exception("Значение трудоемкости 
                    меньше или равно 0");
                }

                i++;

            }

        }
    }

    public double[] GetLines()
    {
        return x;
    }

    public double[] GetTrudoemkost()
    {
        return y;
    }

    public FromFile(string path)
    {
        this.path = path;
        this.read();
    }
}
}
```

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp1
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Graph());
        }
    }
}
```

Graph.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;

namespace WindowsFormsApp1
{
    public partial class Graph : Form
    {
        private double[] x;
```

```
private double[] y;
private bool isShown = false;
private bool isFirst = true;
private LnNormalizer ln;

public Graph()
{
    InitializeComponent();
    this.button3.Visible = false;
}

private void Graph_Load(object sender, EventArgs e)
{

}

private void panel1_Paint(object sender, PaintEventArgs e)
{

}

private void panel4_Paint(object sender, PaintEventArgs e)
{

}

       private void buildGraph(double[] lnY, double[]
 lnMinusDelta, double[] lnPlusDelta, double[] x, bool
 doveritelniy)
{
    Chart mych = new Chart();

    ChartArea chA = new ChartArea();
    chA.AxisX.Title = "Строки кода";
    chA.AxisY.Title = "Трудоемкость";
    mych.ChartAreas.Add(chA);
    mych.Dock = DockStyle.Fill;
    mych.Visible = true;
    mych.Legends.Add(new Legend("Legend"));

    mych.Series.Add("Y");
    mych.Series["Y"].SetDefault(true);
```

```
                mych.Series["Y"].Enabled = true;
                mych.Series["Y"].Color = Color.Blue;
                if(doveritelniy)
                    mych.Series["Y"].ChartType = SeriesChartType.Point;
                else
                    mych.Series["Y"].ChartType = SeriesChartType.Line;
                mych.Series["Y"].LegendText = "y";

                mych.Series.Add("lnPlusDelta");
                mych.Series["lnPlusDelta"].SetDefault(true);
                mych.Series["lnPlusDelta"].Enabled = true;
                mych.Series["lnPlusDelta"].ChartType =
SeriesChartType.Point;
                mych.Series["lnPlusDelta"].Color = Color.Red;
                mych.Series["lnPlusDelta"].LegendText = "ln + delta";

                mych.Series.Add("lnMinusDelta");
                mych.Series["lnMinusDelta"].SetDefault(true);
                mych.Series["lnMinusDelta"].Enabled = true;
                mych.Series["lnMinusDelta"].ChartType =
SeriesChartType.Point;
                mych.Series["lnMinusDelta"].Color = Color.Red;
                mych.Series["lnMinusDelta"].LegendText = "ln - delta";

                for (int i = 0; i < lnY.Length; i++)
                {
                    mych.Series["Y"].Points.AddXY(x[i], lnY[i]);
                          mych.Series["lnMinusDelta"].Points.AddXY(x[i],
                     lnMinusDelta[i]);
                          mych.Series["lnPlusDelta"].Points.AddXY(x[i],
                     lnPlusDelta[i]);
                }

                if (doveritelniy)
                {
                    mych.Titles.Add("Доверительный интервал");
                    this.panel3.Controls.Add(mych);
                }
                else
                {
                    mych.Titles.Add("Интервал прогнозирования");
                    this.panel2.Controls.Add(mych);
```

```
    }
}


       private void putLabels(double lnRSquare, double lnMMRE,
 double lnPRED)
{
    this.label11.Text = Math.Round(lnRSquare, 4).ToString();
    this.label13.Text = Math.Round(lnMMRE, 4).ToString();
    this.label15.Text = Math.Round(lnPRED, 4).ToString();
    private void button1_Click(object sender, EventArgs e)
{
    if (isShown)
    {
        checkHide();
        isFirst = true;
    }
    FileReader filereader = null;

    var fd = new System.Windows.Forms.OpenFileDialog();
    if (fd.ShowDialog() == System.Windows.Forms.DialogResult.OK)
    {
        try
        {
            filereader = new FromFile (fd.FileName);
            this.label1.Text = fd.FileName;
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }

    if (filereader != null)
    {
        x = filereader.GetLines();
        y = filereader.GetTrudoemkost();

        ln = new LnNormalizer(x, y);

               this.buildGraph(ln.getY(),ln.getYMinusDelta(),
         ln.getYPlusDelta(), x, true);
```

```
                this.buildGraph(ln.getY(),
                ln.getYMinusDeltaPrognozirovaniya(),
          ln.getYPlusDeltaPrognozirovaniya(), x, false);
                this.putLabels(ln.getRSquare(),ln.getMMRE(),
                ln.getPRED());
          this.button3.Visible = true;
     }
}

private void panel3_Paint(object sender, PaintEventArgs e)
{

}

private void panel3_Paint_1(object sender, PaintEventArgs e)
{

}

private void panel2_Paint(object sender, PaintEventArgs e)
{

}

private void label11_Click(object sender, EventArgs e)
{

}

private void label8_Click(object sender, EventArgs e)
{

}

private void label12_Click(object sender, EventArgs e)
{

}
```