МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

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

Програмного забезпечення автоматизованих систем
(повна назва кафедри)

Пояснювальна записка

до кваліфікаційної (магістерської) роботи

за темою

«Удосконалення однофакторної регресійної моделі для оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java, та розробка програми для її реалізації»

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

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

м. Миколаїв – 2020 р.

Міністерство освіти і науки України
Національний університет кораблебудування
імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра програмного забезпечення автоматизованих систем
Освітній ступень Магістр
Галузь 12 «Інформаційні технології»
(шифр і назва)
Спеціальність 121 «Інженерія програмного забезпечення»
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри

" 26 " 10 2020 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ (МАГІСТЕРСЬКУ) РОБОТУ СТУДЕНТУ

Пічугін Максим Юрійович
(прізвище, ім'я, по батькові)

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

4.Зміст магістерської роботи (МР):
• Титульний аркуш, завдання на кваліфікаційну (магістерську) роботу, реферат (українською, англійською), зміст, перелік умовних позначень, символів, одиниць та термінів (за необхідності).
• Вступ (Актуальність теми. Зв'язок роботи з науковими програмами, планами, темами. Мета і завдання дослідження. Об'єкт дослідження. Предмет дослідження. Методи дослідження. Наукова новизна одержаних результатів. Практичне значення одержаних результатів. Особистий внесок здобувача. Апробація результатів досліджень. Публікації.)
• Огляд літератури за темою, обґрунтування необхідності проведення досліджень за обраною темою, вибір напрямків досліджень, мета дослідження, основні задачі дослідження
• Викладення результатів власних досліджень з висвітленням того нового, що пропонується
• Проект програмного забезпечення
• Результати досліджень та розробки проекту програмного забезпечення
• Організаційно-економічний розділ
• Розділи з охорони праці та охорони навколишнього середовища
• Висновки
• Список використаних джерел
• Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма і методика випробувань програмного забезпечення)

5. Перелік графічного матеріалу

_____
_____
_____
_____
_____
_____

6. Консультанти розділів магістерської роботи

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

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

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

Студент _____ Пічугін М.Ю.
(підпис) (прізвище та ініціали)

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

РЕФЕРАТ

Пічугін Максим Юрійович

«Удосконалення однофакторної регресійної моделі для оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java, та розробка програми для її реалізації»

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

Обсяг роботи: 122 стор., 12 табл., 16 рис., 28 використаних джерел, 5 додатків.

Актуальність теми роботи: визначається необхідністю підвищення достовірності оцінювання тривалості виконання задач з розробки веб-застосунків, реалізованих мовою Java, та розробкою відповідної програми.

Мета та завдання дослідження: метою роботи є підвищення достовірності оцінювання тривалості виконання задач з розробки веб-застосунків, реалізованих мовою Java.

Об’єкт дослідження: процес оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java.

Предмет дослідження: однофакторна регресійна модель для оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java.

Методи дослідження: методи теорії ймовірності та математичної статистики, регресійного аналізу побудови нелінійних регресійних моделей на основі нормалізуючих перетворень, об`єктно-орієнтованого програмування.

Наукова новизна одержаних результатів: полягає в удосконаленні однофакторної регресійної моделі для оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java, за рахунок використання нормалізуючого перетворення сім'ї S_B Джонсона, що дозволило підвищити достовірність оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java, в порівнянні з існуючими моделями COCOMO та ISBSG.

Практичне значення одержаних результатів: розробка програмного забезпечення для оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java, за рахунок використання нормалізуючого перетворення сім'ї S_B Джонсона.

Апробація результатів досліджень: основні положення і результати досліджень, викладені у кваліфікаційній роботі, пройшли апробацію на VI міжнародній науково-технічній конференції «Комп'ютерне моделювання та оптимізація складних систем» (м. Дніпро, 4 – 6 листопада 2020 р.).

Публікації: Основні результати кваліфікаційної роботи викладено в 1 науковій праці – тезах конференції.

Ключові слова: однофакторна регресійна модель, нелінійна регресія, нормалізуюче перетворення Джонсона, оцінювання тривалості виконання робіт, розробка веб-застосунків, Java.

ABSTRACT

Pichuhin Maksym Yuriiovych

«Improving the univariate regression model for estimating the duration of web applications implemented in Java development and developing the software for its implementation»

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

Volume of work: 122 pages of typewritten text, 12 tables, 16 figures, a list of references from 28 names, 5 appendices.

Relevance of the theme: determined by the need to increase the reliability of estimating the duration of tasks for the development of web applications implemented in Java, and the development of an appropriate software.

The goal and objectives of the study: the purpose of this work is to improve the reliability of estimating the duration of tasks for the development of web applications implemented in Java.

The object of the study: the process of estimating the duration of work on the development of web applications implemented in Java.

The subject of the study univariate regression model for estimating the duration of the work to develop web applications implemented in Java.

Research methods: methods of probability theory and mathematical statistics, regression analysis, construction of nonlinear regression models based on normalizing transformations, object-oriented programming.

The scientific novelty of the obtained results: is to improve the univariate regression model for estimating the duration of work on web applications implemented in Java, using the normalizing transformation of the S_B Johnson family, which increased the reliability of estimating the duration of work on web applications implemented in Java, compared to existing models COCOMO and ISBSG.

The practical significance of the obtained results: software development for estimating the duration of work on the development of web applications implemented in Java, using the normalizing transformation of the S_B Johnson family.

Approbation of research results: the main provisions and research results presented in the qualification work have been approbat at the VI International Scientific and Technical Conference "Computer Modeling and Optimization of Complex Systems" (Dnipro, November 4 – 6, 2020).

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

Keywords: univariate regression model, nonlinear regression, Johnson normalizing transformation, estimating the duration of work, web application development, Java.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД	База даних
ВВ	Випадкова величина
ЕД	Емпіричні дані
МНК	Метод найменших квадратів
ПЗ	Програмне забезпечення
CMM	Capability Maturity Model
COCOMO	COnstructive COst MOdel
ISBSG	International Software Benchmarking Standards Group
UI	User Interface

ВСТУП

Актуальність теми. Максимально точне оцінювання необхідних трудових витрат на розробку програмних продуктів є однією з основних проблем при управлінні проектами розробки ПЗ. Існуючі моделі та методи для оцінювання тривалості виконання робіт з розробки програмного забезпечення використовують різні фактори: кількість рядків коду, кількість функціональних точок, кількість сторінок програмно-технічної та користувацької документації, досвід розробників та інші [1 – 4].

Чи не найбільш популярними програмами нині являються веб-застосунки. Згідно зі статистикою, найбільш популярними веб-додатками є: Facebook показав найвищий рівень серед людей віком від 18 років, Youtube – другий, а третій – Facebook Messenger. Цікаво відзначити, що серед 10 найпопулярніших додатків, 3 додатки – Facebook, FB Messenger та Instagram, що належать Facebook і 5 додатків – Youtube, пошук Google, Google Maps, Google Play і Gmail, що належать Google. Серед десятки найпопулярніших також опинились Snapchat і Pandora [5].

Серед популярних технологій та мов програмування, котрі нині застосовують для розробки веб-додатків на серверній частині можна виокремити: PHP, Java, Python, а на клієнтській частині застосовують: HTML, CSS, Activex, Javascript.

Важко за короткий термін та в необхідному обсязі отримати інформацію для оцінювання тривалості розробки, оскільки ця інформація може бути недоступною на ранніх етапах розробки проекту. Крім того, більшість існуючих моделей є лінійними, які не враховують нелінійні зв’язки між факторами. Тому необхідність удосконалення математичної моделі для оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java, з використанням нормалізуючих перетворень, є актуальною задачею. Така удосконалена модель дасть можливість підвищити

достовірність оцінювання тривалості виконання робіт, з розробки веб-застосунків, реалізованих мовою Java

Мета і завдання дослідження. Метою роботи є підвищення достовірності оцінювання тривалості виконання задач з розробки веб-застосунків, реалізованих мовою Java.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1) Проаналізувати існуючі алгоритми та методи оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java, сформулювати постановку задачі;

2) Удосконалити однофакторну регресійну модель для оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java;

3) Розробити програму для оцінювання тривалості виконання задач з розробки веб-застосунків, реалізованих мовою Java.

Об’єктом дослідження є процес оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java.

Предметом дослідження є однофакторна регресійна модель для оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java.

Методи дослідження. Для вирішення поставлених задач були застосовані методи теорії ймовірностей та математичної статистики, регресійного аналізу, побудови нелінійних регресійних моделей на основі нормалізуючих перетворень, об`єктно-орієнтованого програмування.

Наукова новизна одержаних результатів полягає в удосконаленні однофакторної регресійної моделі для оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java, за рахунок використання нормалізуючого перетворення сім'ї S_B Джонсона, що дозволило підвищити достовірність оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java, в порівнянні з існуючими моделями COCOMO та ISBSG.

Практичне значення отриманих результатів полягає в розробці ПЗ для оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java, за рахунок використання нормалізуючого перетворення сім'ї S_B Джонсона.

Особистий внесок здобувача. Кваліфікаційна робота є самостійно виконаною працею. Усі результати, викладені у роботі, отримані автором особисто. У праці, яка опублікована у співавторстві з керівником кваліфікаційної роботи [6], здобувачеві належить: удосконалення однофакторної нелінійної регресійної моделі для оцінювання тривалості виконання робіт.

Апробація результатів роботи. Основні положення і результати досліджень, викладені у кваліфікаційній роботі, пройшли апробацію на VI міжнародній науково-технічній конференції «Комп'ютерне моделювання та оптимізація складних систем» (м. Дніпро, 4 – 6 листопада 2020 р.).

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

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ТА МОДЕЛЕЙ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ ВИКОНАННЯ РОБІТ З РОЗРОБКИ ВЕБ-ЗАСТОСУНКІВ, РЕАЛІЗОВАНИХ МОВОЮ JAVA

1.1 Веб-застосунки: сутність, особливості та технології створення

Веб-застосунок – це програма, яка призначена для користувача, головна частина якої знаходиться на вилученому сервері, а призначений для користувача інтерфейс UI користувач бачить у браузері у вигляді веб-сторінок.

Технологія «клієнт-сервер» яскраво виражена у веб-застосунках, бо вони складаються з двох частин: серверної та клієнтської. На сервер клієнт відправляє запит, сервер його одержує та здійснює необхідні обчислення, потім формує веб-сторінку і посилає її клієнту через мережу використовуючи HTTP. За допомогою клієнтської частини на екрані відображається інтерфейс, відбувається формування запитів до сервера та обробляється відповідь, отримана від нього. Іноді веб-застосунок може виступати в якості клієнта інших служб, таких як, бази даних або веб-додатку, розташованого на іншому сервері [7].

Для створення веб-застосунків на серверній частині використовуються різні технології та мови програмування, за допомогою яких можна здійснити вивід в стандартну консоль. Серед популярних технологій та мов програмування, котрі нині застосовують для розробки веб-додатків на серверній частині можна виокремити: PHP, Java, ASP.NET, Perl, Python, Ruby, Node.Js. На клієнтській частині у той час застосовують: для реалізації GUI: XHTML, HTML, CSS; для формування запитів, створення інтерактивного і незалежного від браузера інтерфейсу: Adobe Flash (Flex), Activex, Java, Javascript, Silverlight [8].

За своєю суттю запуск веб-застосунка нічим не відрізняється від завантаження звичайної веб-сторінки: вводимо посилання в браузер – і воно перед нами, точніше, верхня частина айсберга, якою є інтерфейс користувача. В цьому є кілька переваг. Перша з них – це те, що саме по собі додаток абсолютно не залежить від того, яка операційна система встановлена на комп'ютері користувача, тобто веб-застосунок, по суті, є крос-платформенним.

По-друге, що сам факт існування веб-застосунку повністю змінює спосіб поширення продукту. Тут розробники відходять від традиційних способів поширення програмних продуктів шляхом продажу копій і установки їх на кожен комп'ютер користувачів. Тепер все набагато простіше: єдина версія додатка розташована на сервері, а всі користувачі мають доступ до неї, вірніше, до її призначеного для користувача інтерфейсу з будь-якого місця. При цьому користувачеві навіть не потрібно встановлювати нову версію програми – відразу після своєї появи вона доступна всім, причому багато хто може і не помітити яких-небудь змін, тим більше якщо ці зміни не стосуються зовнішнього вигляду інтерфейсу. У всьому цьому явно видно і позитивний момент для розробників – їм не потрібно піклуватися про сумісність версій своїх застосунків, оскільки всі користувачі одноразово отримують доступ і працюють з самою останньою версією програми.

Третьою перевагою для користувача є те, що немає потреби встановлювати і налаштовувати ПЗ – все вже встановлено на серверах і налаштоване розробниками. Це великий плюс для користувачів, оскільки більша частина з них не любить мати справу з налаштуваннями і вважає за краще, щоб програмні продукти були повністю готові до використання відразу після їх інсталяції, хоча є й такі, які вважають за краще повністю налаштувати програму під свій смак і потреби. Однак в разі використання веб-застосунків користувачі позбавлені навіть процесу інсталяції.

Четвертою перевагою можна назвати те, що для роботи з додатком від користувача нічого не потрібно, окрім комп'ютера і встановленого браузера.

Інтернет-браузер є в будь-якій операційній системі, і для доступу до необхідного веб-застосунку досить просто завантажити його веб-адресу у браузер. Використання веб-застосунків багато в чому знімає обмеження, що накладаються на апаратну частину комп'ютера. Тобто певні системні вимоги до персонального комп’ютера все ж таки є, але їх рівень автоматично досягнутий комп'ютером, якщо на ньому вже запущені операційна система та браузер.

Наступний позитивний момент веб-застосунків стосується їх розробників. З огляду на те, що основна частина веб-застосунків сконцентрована на сервері в одному місці, його налаштування стає набагато простішим, не потрібно утримувати величезні команди фахівців технічної підтримки, що займаються консультаціями користувачів і налаштуванням програми на комп'ютерах. Це менш затратно в фінансовому плані і в той же час ефективніше. При цьому користувачеві невидима архітектура веб-застосунку, в будь-який момент можна додати будь-яку кількість серверів, на яких встановлена основна складова частина програми, додати обчислювальні потужності, і користувач цього навіть не помітить [7].

Таким чином, бачимо, що веб-застосунки мають велику кількість переваг при відсутності видимих недоліків, найбільшим і очевидним з яких є неможливість використання веб-застосунків при відсутності доступу до мережі Інтернет.

1.2 Аналіз існуючих моделей для оцінювання тривалості виконання робіт з розробки програмного забезпечення

COCOMO

Методика COCOMO Дозволяє оцінити трудомісткість та час розробки програмного продукту. Модель складається з ієрархії трьох послідовно

уточнюючих рівнів [2]. На кожному рівні всі проекти розбиваються на три групи за рівнем складності:

1) поширений тип (organic projects);

2) вбудований тип (embedded projects);

3) напівнезалежний тип (semidetached projects).

Поширений тип характеризується тим, що проект виконується невеликою групою фахівців, що мають досвід у створенні подібних проектів і досвід застосування технологічних засобів. Умови роботи стабільні, і проект має відносно невисоку складність.

Вбудований тип характеризується дуже жорсткими вимогами на програмний продукт, інтерфейси, параметри ЕОМ. Як правило, у таких виробів висока ступінь новизни і планування робіт здійснюється за недостатньої інформації, як про самому виробі, так і про умови роботи. Вбудований проект вимагає великих витрат на зміни і виправлення.

Напівнезалежний тип займає проміжне положення між поширеним і вбудованим – це проекти середньої складності. Виконавці знайомі лише з деякими характеристиками (або компонентами) створюваної системи, мають середній досвід роботи з подібними проектами, проект має елемент новизни. Тільки частина вимог до проекту жорстко фіксується, в інших аспектах розробки є ступені вибору.

Тип тієї чи іншої групи можна розглядати як один з параметрів моделі COCOMO.

Базовий рівень (Basic COCOMO)

Модель цього рівня – двопараметрична. В якості параметрів виступають тип проекту і обсяг програми (число рядків коду).

Рівняння базового рівня моделі мають вигляд:

$$PM = a_i * (SIZE)^{b_i}$$

$$TM = c_i * (PM)^{d_i},$$

$$SS = \frac{PM}{TM},$$

$$P = \frac{SIZE}{PM},$$

де *PM* (People × Month) – трудомісткість (чол. × міс.);

SIZE – обсяг програмного продукту в тисячах рядків вихідного тексту (Kilo of Source Line of Code – KSLOC);

TM (Time at Month) – час розробки в календарних місяцях;

SS – середня чисельність персоналу;

P – продуктивність.

Коеффіцієнти a_i, b_i, c_i та d_i вибираються з відповідних таблиць в залежності від типу проекту.

Модель цього рівня підходить для ранньої швидкої приблизної оцінки витрат, але її точність дуже низька, тому що не враховуються такі чинники, як кваліфікація персоналу, характеристики обладнання, досвід застосування сучасних методів розробки програмного забезпечення і сучасних інструментальних середовищ розробки та ін.

Середній рівень (Intermediate COCOMO)

На цьому рівні базова модель уточнена за рахунок введення додаткових 15 «Атрибутів вартості» (або факторів витрат) Cost Drivers (CD_k), які згруповані за чотирма категоріями. Значення кожного атрибута також вибирається з відповідних таблиць згідно з його ступенем значущості (рейтингом) в конкретному проекті.

Рівняння середнього рівня моделі має вигляд:

$$PM = EAF * a_i * (SIZE)^{b_i},$$

де *PM* (People × Month) – трудомісткість (чол. × міс.);

EAF (Effort Adjustment Factor) – добуток обраних атрибутів з відповідних таблиць;

SIZE – обсяг програмного продукту в тисячах рядків вихідного тексту (Kilo of Source Line of Code – KSLOC).

Коефіцієнти моделі a_i та b_i вибираються з відповідної таблиці в залежності від типу проекту.

Детальний рівень (Advanced COCOMO)

Підвищує точність оцінки за рахунок ієрархічної декомпозиції створюваного ПЗ і обліку вартісних факторів на кожному рівні ієрархії і за фазами робіт.

У 1997 році методика COCOMO була вдосконалена і отримала назву *COCOMO II*. Калібрування параметрів проводилася по 161 проекту розробки ПЗ. Розрізняються дві стадії оцінки проекту: попередня оцінка на початковій фазі (Early Design) і детальна оцінка після опрацювання архітектури (Post Architecture).

Формула оцінки трудомісткості проекту в чол.×міс. має вигляд:

$$PM = EAF * A * (SIZE)^E,$$

де *EAF* (Effort Adjustment Factor) – добуток обраних множників трудомісткості: $EAF = \prod_{k=1}^{n} EM_k$;

EMj – множники трудоемкості (Effort Multiplier), кількість і значення яких відрізняються для різних стадій оцінки проекту (сім множників для стадії попередньої оцінки та сімнадцять множників для стадії детальної оцінки після опрацювання архітектури) та вибирається з відповідних таблиць;

A=2,94 для попередньої оцінки та *A*=2,45 для детальної оцінки;

SIZE – обсяг програмного продукту в тисячах рядків вихідного тексту (KSLOC – Kilo of Source Line of Code);

$E = B + 0.01 * \sum_{j=1}^{5} SF_j$, B=0,91;

SF_j – фактори масштабу (Scale Factors), які вибираються з відповідної таблиці, у методиці COCOMO II використовуються п'ять факторів масштабу (прецедентність, наявність досвіду аналогічних розробок; гнучкість процесу розробки; архітектура і дозвіл ризиків; спрацьованість команди; зрілість процесів), а також шість рівнів впливу факторів (дуже низький; низький; середній; високий; дуже високий; критичний), які застосовуються на обох стадіях оцінки проекту.

Оцінка тривалості проекту

Час розробки проекту *TM* в методиці COCOMO II для обох рівнів розраховується за формулою:

$$TM = SCED * C * (PM_{NS})^{D+0.2*(E-B)},$$

де *SCED* – множник, що визначає стиснення розкладу;

C=3,67; *D*=0,28;

PM_{NS} – розрахована трудомісткість проекту без урахування множника *SCED*.

Інші параметри визначені вище.

З врахуванням того, що один людино-місяць складається з 152 людино-години, для попереднього оцінювання тривалості робіт в залежності від трудомісткості програмних проектів можуть бути використані наступні розрахункові формули

– для органічного типу проектів

$$D = 0{,}371 \cdot E^{0{,}38}, \quad (1.1)$$

– для напіврозділеного типу проектів

$$D = 0{,}431 \cdot E^{0{,}35}, \quad (1.2)$$

– для вбудованого типу проектів

$$D = 0{,}501 \cdot E^{0{,}32}, \quad (1.3)$$

де *D* – тривалість проекту (в місяцях);

E – трудомісткість проекту (в людино-годинах).

Модель ISBSG – регресійна модель для оцінювання тривалості робіт в проектах з розробки ПЗ. Як фактор в цих регресійних моделях було використано трудомісткість програмних проектів:

$$D = 0{,}662 \cdot E^{0{,}328}, \quad (1.4)$$

де *D* – тривалість проекту (в місяцях);

E – трудомісткість проекту (в людино-годинах).

Закон розподілу ЕД тривалості робіт в програмних проектах та трудомісткості цих проектів зазвичай не є нормальним. Крім того, регресійні моделі COCOMO та ISBSG не дозволяють виконувати оцінювання довірчих інтервалів тривалості робіт.

1.3 Удосконалення регресійної моделі для оцінювання тривалості виконання робіт з розробки програмного забезпечення

В загальному вигляді регресійна модель може бути представлена рівнянням:

$$y = \bar{y} + \varepsilon_t = f(x) + \varepsilon_t; \quad (1.5)$$

де y – залежна змінна, або результативна ознака;

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

x – незалежна змінна, або фактор;

ε_t – випадкова помилка, або збурення.

Однак при побудові подібних регресійних моделей виникає ряд труднощів:

– якщо випадкові величини, що входять до регресійної моделі, не підпорядковуються нормальному закону розподілу, це призводить до нелінійної регресії;

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

Для побудови нелінійних регресійних моделей існує кілька методів: метод простого перебору, метод лінеаризуючих перетворень, метод нормалізуючих перетворень [9].

Метод простого перебору, або метод усіх можливих регресій, вимагає завдання різних видів рівняння регресії і вибору найкращого наближення із заданих за певним критерієм. Це вимагає досить великої кількості обчислень і не завжди призводить до отримання найкращого рішення. Всі ці недоліки роблять даний метод неефективним [10].

Метою *методу лінеаризуючих перетворень* є перехід від нелінійної регресії до лінійної шляхом заміни вихідних змінних і коефіцієнтів, проте не завжди можливо підібрати таку заміну [11 – 13]. Крім того, така заміна призводить до спрощення регресійної моделі і деякої втрати інформації, пов'язаної з нелінійністю.

Метою *методу нормалізуючих перетворень* є пошук перетворень, за допомогою яких можна здійснити перехід від вихідних негаусівських

випадкових величин до гаусівських. Для отриманих гаусівських випадкових величин будують лінійне рівняння регресії, яке далі перетворять на нелінійне рівняння за допомогою раніше знайдених нормалізуючих перетворень [12].

У разі нормального закону розподілу випадкових величин можна побудувати лінійну регресійну модель:

$$z_y = b_1 z_x + b_0 + \varepsilon; \tag{1.6}$$

де b_1, b_0 – коефіцієнти лінійної регресії, які знаходяться методом найменших квадратів:

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

$$b_0 = \frac{\sum_{i=1}^{n} z_{yi} - b_1 \sum_{i=1}^{n} z_{xi}}{n};$$

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

Для оцінювання точності побудови рівняння регресії намагаються знайти його довірчий інтервал. У випадку нормального закону розподілу випадкових величин довірчий інтервал лінійного рівняння регресії можливо побудувати традиційним методом з використанням t-розподілу Стьюдента [14]:

$$y = \hat{y} \pm t(\alpha/2, n-2) \cdot S \cdot \sqrt{\frac{1}{n} + \frac{(x - \bar{x})^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2}}; \quad (1.7)$$

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

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

α – рівень значущості;

n – кількість значень випадкових величин у вибірці;

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

Інтервал прогнозування лінійного рівняння регресії у даному випадку можна побудувати за наступною формулою:

$$y = \hat{y} \pm t_{(\alpha/2, n-2)} \cdot S \cdot \sqrt{1 + \frac{1}{n} + \frac{(x - \bar{x})^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2}}, \quad (1.8)$$

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

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

α – рівень значущості;

n – кількість значень випадкових величин у вибірці;

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

Оскільки використання методу нормалізуючих перетворень для рішення аналогічних задач дає гарні результати [14 – 16], це дозволяє застосувати даний метод для удосконалення регресійної моделі для оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java.

1.4 Перевірка вихідних емпіричних даних на викиди

Для знаходження викидів у вихідних ЕД використаємо квадрат відстані Махаланобіса:

$$d_i^2 = (z_i - \bar{z})^T S_n^{-1} (z_i - \bar{z}) \tag{1.9}$$

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

S_N – матриця кореляції вибірки, яка визначається за формулою:

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

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

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

$$Ts = \frac{(N-m)Nd_i^2}{(N^2-1)m}, \tag{1.10}$$

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

У данній роботі будемо використовувати критерій Фишера *F*. Тестова статистика для квадрату відстані Махаланобіса порівнюється з квантилем розподілу *F*, який позначається як $F_{m,N-m,\alpha}$. Тут α – це рівень значущості, який у нашому випадку дорівнюватиме 0,05. Точки, для яких значення T_S,

розраховане за формулою (1.10), буде більше, ніж квантиль розподілу F вважаються викидами, і ці значення видаляються з набору даних.

Після усунення викидів отриманий новий набір значень знову нормалізується та для нього знову знаходиться квадрат відстані Махаланобіса за формулою (1.9) та тестова статистика за формулою (1.10). Процедура повторюється до тих пір, поки всі значення T_S не будуть менше або дорівнюватимуть квантилю розподілу F.

2 УДОСКОНАЛЕННЯ ОДНОФАКТОРНОЇ РЕГРЕСІЙНОЇ МОДЕЛІ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ ВИКОНАННЯ РОБІТ З РОЗРОБКИ ВЕБ-ЗАСТОСУНКІВ, РЕАЛІЗОВАНИХ МОВОЮ JAVA

2.1 Однофакторна регресійна модель для оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java

Розподіл ЕД трудомісткості та часу виконання робіт з розробки веб-застосунків, реалізованих мовою Java, відрізняється від нормального закону розподілу. Тому для побудови нелінійної регресійної моделі буде використано нормалізовані значення трудомісткості та часу виконання робіт з розробки веб-застосунків, реалізованих мовою Java.

Для нормалізації ЕД будемо використовувати чотирьохпараметричне нормалізуючи перетворення Джонсона. Сім'ї розподілів Джонсона отримані шляхом перетворення нормованої нормально розподіленої випадкової величини. У загальному випадку перетворення має вигляд [17]:

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

де z – нормована нормально розподілена випадкова величина;

γ, η, φ, λ – параметри перетворення Джонсона, з яких γ і η – параметри форми, φ – параметр зсуву, λ – параметр масштабу;

x – випадкова величина, яка нормалізується;

h – функція певної сім'ї:

$$h_1(x, \varphi, \lambda) = \ln(\tilde{x}), x > \varphi; \tag{2.2}$$

$$h_2(x, \varphi, \lambda) = \ln\left(\frac{\tilde{x}}{1-\tilde{x}}\right), \varphi < x < \varphi + \lambda; \tag{2.3}$$

$$h_3(x, \varphi, \lambda) = Arsh(\tilde{x}), -\infty \le x \le +\infty. \tag{2.4}$$

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

Перетворення (2.1) має зворотне перетворення:

$$x = \varphi + \lambda h^{-1}(z, \gamma, \eta);\ \eta > 0; -\infty < \gamma < \infty; \lambda > 0; -\infty < \varphi < \infty; \tag{2.5}$$

де x – випадкова величина з розподілом Джонсона;

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

z – нормально розподілена випадкова величина;

h^{-1} – функції певної сім'ї:

$$h_1^{-1}(z, \gamma, \eta) = e^{\zeta}; \tag{2.6}$$

$$h_2^{-1}(z, \gamma, \eta) = \frac{1}{1 + e^{-\zeta}}; \tag{2.7}$$

$$h_3^{-1}(z, \gamma, \eta) = \frac{e^{\zeta} - e^{-\zeta}}{2}. \tag{2.8}$$

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

Для вибору конкретної сім'ї розподілу Джонсона використовують або діаграму в площині ексцес – асиметрія в квадраті $\varepsilon - A^2$, або аналітичну залежність $\varepsilon(A^2)$. Для автоматизації розрахунків та при розробці ПЗ зручніше використовувати аналітичну залежність, наведену в [18]:

$$\varepsilon(A^2) = 3{,}59 \cdot 10^{-6} A^8 - 4{,}8805 \cdot 10^{-4} A^6 + 4{,}1655 \cdot 10^{-2} A^4 + \\ + 1{,}8203 A^2 + 2{,}9658 \quad (2.9)$$

Якщо точка з координатами (A^2, ε) знаходиться біля лінії S_L, то можна використовувати розподіл з сім'ї S_L. Якщо точка з координатами (A^2, ε) знаходиться вище лінії S_L, то можна використовувати розподіл з сім'ї S_U, а якщо нижче лінії S_L до лінії критичної області – то розподіл з сім'ї S_B. Якщо ж точка потрапляє у критичну область, то використовувати для апроксимації сім'ї розподілів Джонсона не можна [9].

2.2 Оцінка адекватності регресійної моделі

Для перевірки адекватності рівняння регресії використаємо коефіцієнт детермінації R^2:

$$R^2 = 1 - \left(\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \right), \quad (2.10)$$

де y_i – емпіричне значення y;

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

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

При значенні $R^2 \geq 0{,}5$ можна вважати, що дана модель є прийнятною. Достатньо ефективною та результативною можна вважати модель з показником детермінації $R^2 \geq 0{,}8$. Якщо ж $R^2 = 1$, то лінія регресії точно відповідає усім спостереженням та вимогам, а модель можна вважати адекватною та достовірною.

Величину відносної похибки MRE (Magnitude of Relative Errors) для лінійного та нелінійного рівнянь регресії можна знайти за формулою:

$$MRE_i = \left| \frac{y_i - \hat{y}_i}{y_i} \right|, \tag{2.11}$$

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

y_i – фактичне значення ВВ y.

MMRE (Mean of Magnitude of Relative Errors – середня величина відносної похибки) можна розрахувати за формулою:

$$MMRE = \frac{1}{N} \sum_{i=1}^{N} MRE_i \, . \tag{2.12}$$

Рівень прогнозування PRED(l) можна розрахувати за формулою:

$$PRED(l) = \frac{k}{N}, \tag{2.13}$$

де k – кількість значень з $MRE \leq l$.

Таким чином, маємо ряд показників для перевірки адекватності регресійних моделей та порівняння їх між собою.

2.3 Побудова удосконаленої однофакторної регресійної моделі для оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java

Для побудови удосконаленої однофакторної регресійної моделі в якості вихідних ЕД були розглянуті дані 499 проектів з розробки програмного забезпечення, які взяті з відкритого джерела – репозиторію tera-PROMISE. В якості випадкової величини x взята трудомісткість в людино-годинах, випадкової величини y – час виконання робіт в місяцях (проведено перетворення годин в місяці для порівняння розрахунків з моделями COCOMO та ISBSG).

Для продовження роботи з ЕД необхідно переконатися в якості їх збору та відсутності викидів, так як наявності викидів може бути причиною того, що дані не розподіляються за нормальним законом. Перевірка на наявність викидів проводиться ітераційно.

Одним з варіантів перевірки даних на викиди є перевірка за допомогою квадрата відстані Махаланобіса (Mahalanobis squared distance), формули (1.9), (1.10). У таблиці 2.1 наведено фрагмент даних, значення вихідних ЕД X та Y, нормалізовані значення Z_X та Z_Y, значення T_S для d_i^2 на першій ітерації, для якої F=3,014, m=2, α=0,05.

Таблиця 2.1 – Фрагмент даних

№ п/п	X	Y	Z_X	Z_Y	T_S для d_i^2
1	2	3	4	5	6
51	1464	6	0,000002	-0,197635	0,0331
52	1636	2	0,000002	-1,592412	1,5717
53	281	7	0,000000	0,016166	1,6643
54	2796	5	0,000003	-0,445549	0,2716
55	4884	13	0,000003	0,910683	0,4237
56	**2400**	**84**	**0,000002**	**4,162215**	**11,5840**
57	5650	13	0,000003	0,910683	0,4664
58	1837	2	0,000002	-1,592412	1,6684
59	575	2	0,000001	-1,592412	1,2193
60	606	2	0,000001	-1,592412	1,2149
61	10205	10	0,000004	0,524659	0,8695

Продовження табл. 2.1

1	2	3	4	5	6
62	937	3	0,000001	-1,107735	0,5870
63	2597	7,5	0,000002	0,113046	0,0187
64	950	3	0,000001	-1,107735	0,5873
65	5727	13	0,000003	0,910683	0,4714
66	2458	4	0,000002	-0,741073	0,4855
67	4430	8	0,000003	0,204322	0,2020
68	687	2	0,000001	-1,592412	1,2139
69	4530	8	0,000003	0,204322	0,2151
70	1084	4	0,000002	-0,741073	0,2668

На першій ітерації викидами є 25 наборів даних. Видаляємо їх з загального набору ЕД та повторюємо перевірку. Загалом за три ітерації було видалено 73 набори даних. Скоригована таким чином вибірка ЕД для подальшого аналізу склала 426 наборів даних.

Нормалізацію ЕД виконуємо за допомогою перетворення Джонсона, формула (2.1). За допомогою формули (2.9) визначаємо сім'ю розподілів Джонсона. І для ВВ X, і для ВВ Y це сім'я S_B Джонсона. Параметри перетворення Джонсона для ВВ X та Y наведені в табл. 2.2.

Таблиця 2.2 – Параметри перетворення Джонсона для ВВ X та Y

Параметр	ВВ X	ВВ Y
γ	1,7250	0,9744
η	0,7467	0,9343
φ	135,49	1,0104
λ	19362,52	22,2027

Отримані нормалізовані вибірки Z_X та Z_Y відповідають нормальному закону розподілу: χ^2_{Zx}=8,14 та χ^2_{Zy}=4,72 при критичному значенні χ^2=12,59 для довірчої ймовірності 0,95.

Гістограми розподілу нормалізованих значень для вибірок Z_X та Z_Y зображені на рис. 2.1 та 2.2 відповідно.

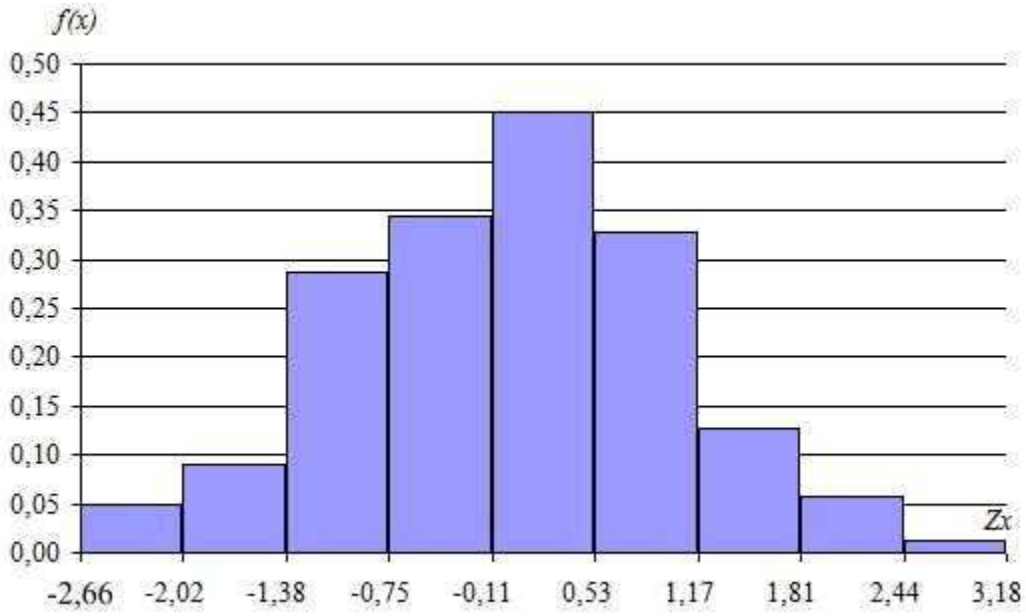

Рисунок 2.1 – Гістограма розподілу нормалізованих значень для вибірки Z_X

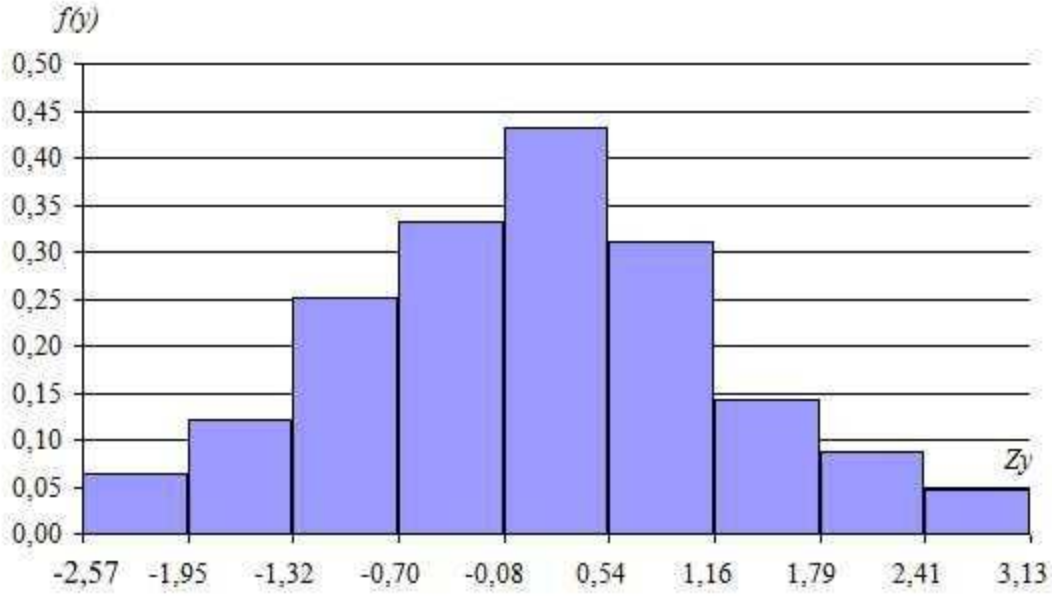

Рисунок 2.2 – Гістограма розподілу нормалізованих значень для вибірки Z_Y

Тепер будуємо лінійне рівняння регресії для нормалізованих даних згідно з (1.6) та за допомогою методу найменших квадратів знаходимо його коефіцієнти: b_1=0,5105, b_0=0,0002. Лінійна регресійна модель має вигляд $Z_y = 0{,}5105 Z_x + 0{,}0002 + \varepsilon$.

Довірчий інтервал лінійного рівняння регресії знаходимо згідно з (1.7). Інтервал прогнозування лінійного рівняння регресії знаходимо згідно з (1.8). Нормалізовані дані, лінійне рівняння регресії та відповідні інтервали зображено на рис. 2.3.

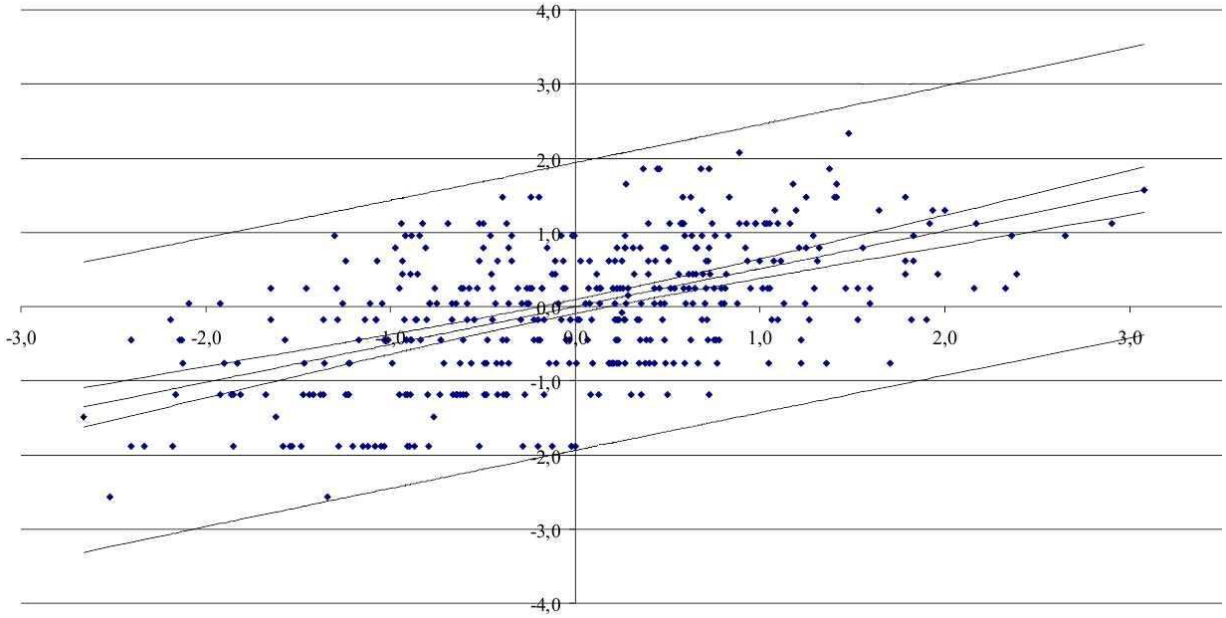

Рисунок 2.3 – Нормалізовані дані, лінійне рівняння регресії, довірчий інтервал та інтервал прогнозування лінійного рівняння регресії

Як видно з рис. 2.3, усі нормалізовані дані знаходяться в середині інтервалу прогнозування. Тобто, усі викиди на етапі первинної обробки даних були видалені правильно.

Для побудови нелінійного рівняння регресії використаємо вже побудоване лінійне рівняння регресії та зворотне нормалізуюче перетворення Джонсона (2.5) з відповідною сім'єю S_B:

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

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

(1-α)% довірчий інтервал нелінійного рівняння регресії можна побудувати, використовуючи лінійне рівняння регресії, *t*-розподіл Стьюдента та зворотне нормалізуюче перетворення Джонсона (2.5) з відповідною сім'єю S_B [9]:

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

де:

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

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

Аналогічно знайдемо і інтервал прогнозування нелінійного рівняння регресії. Перехід до (1-α)% інтервалу прогнозування нелінійного рівняння регресії можна здійснити, використовуючи побудований (1-α)% інтервал прогнозування та зворотне перетворення Джонсона (2.5) з відповідною сім'єю S_B.

Вихідні ЕД, нелінійне рівняння регресії та відповідні інтервали зображено на рис. 2.4.

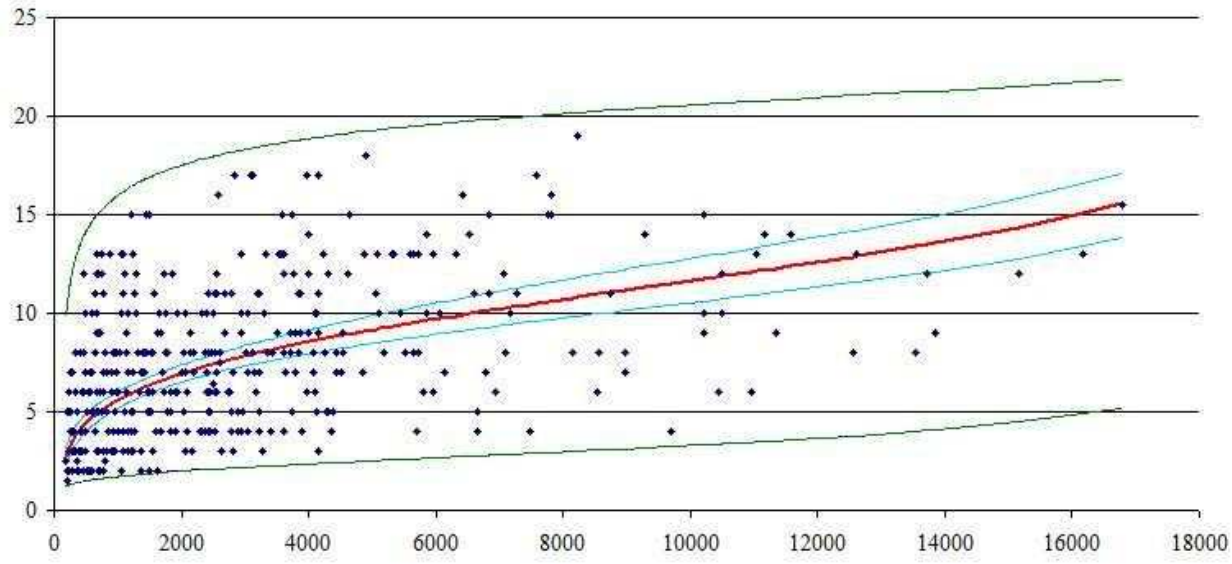

Рисунок 2.4 – Вихідні ЕД, нелінійне рівняння регресії, довірчий інтервал та інтервал прогнозування нелінійного рівняння регресії

Як видно з рис. 2.4, усі ЕД знаходяться в середині інтервалу прогнозування, аналогічно із рис. 2.3 для нормалізованих даних. Тобто, усі викиди на етапі первинної обробки даних були видалені правильно.

Вихідні ЕД, нелінійне рівняння регресії, довірчий інтервал та інтервал прогнозування для моделей COCOMO та ISBSG зображені на рис. 2.5 та 2.6 відповідно.

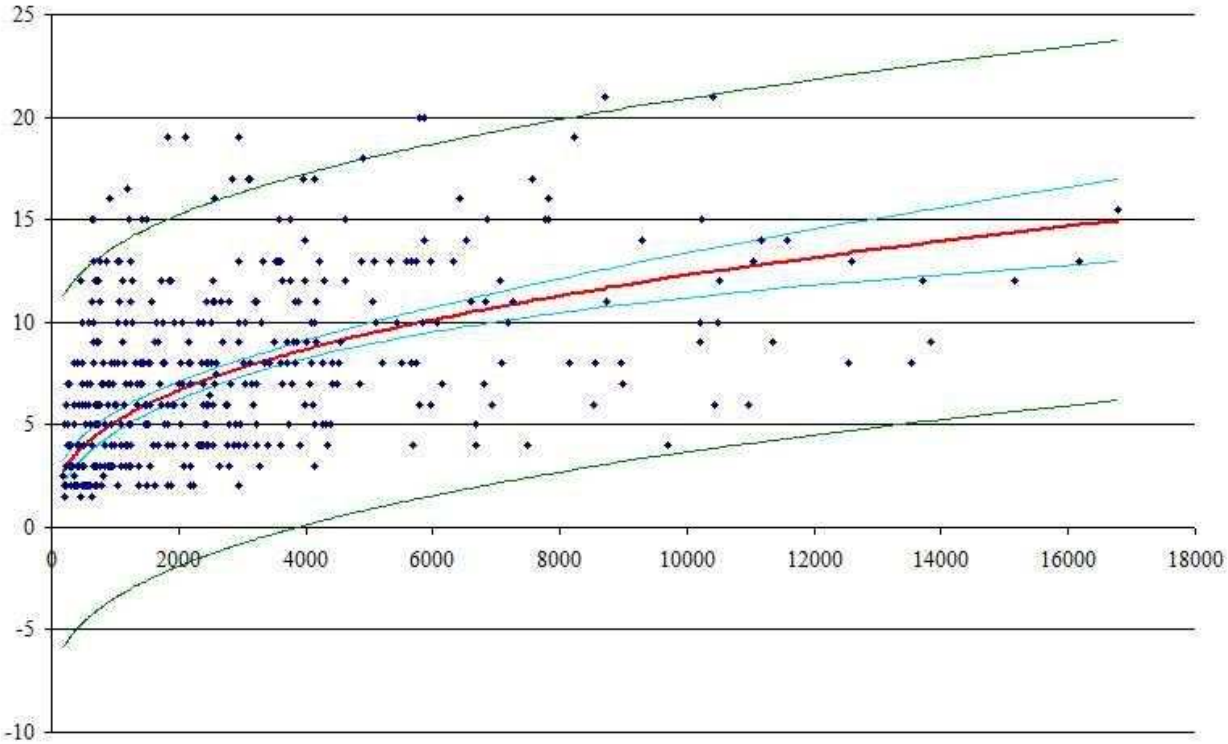

Рисунок 2.5 – Вихідні ЕД, нелінійне рівняння регресії, довірчий інтервал та інтервал прогнозування нелінійного рівняння регресії для моделі COCOMO

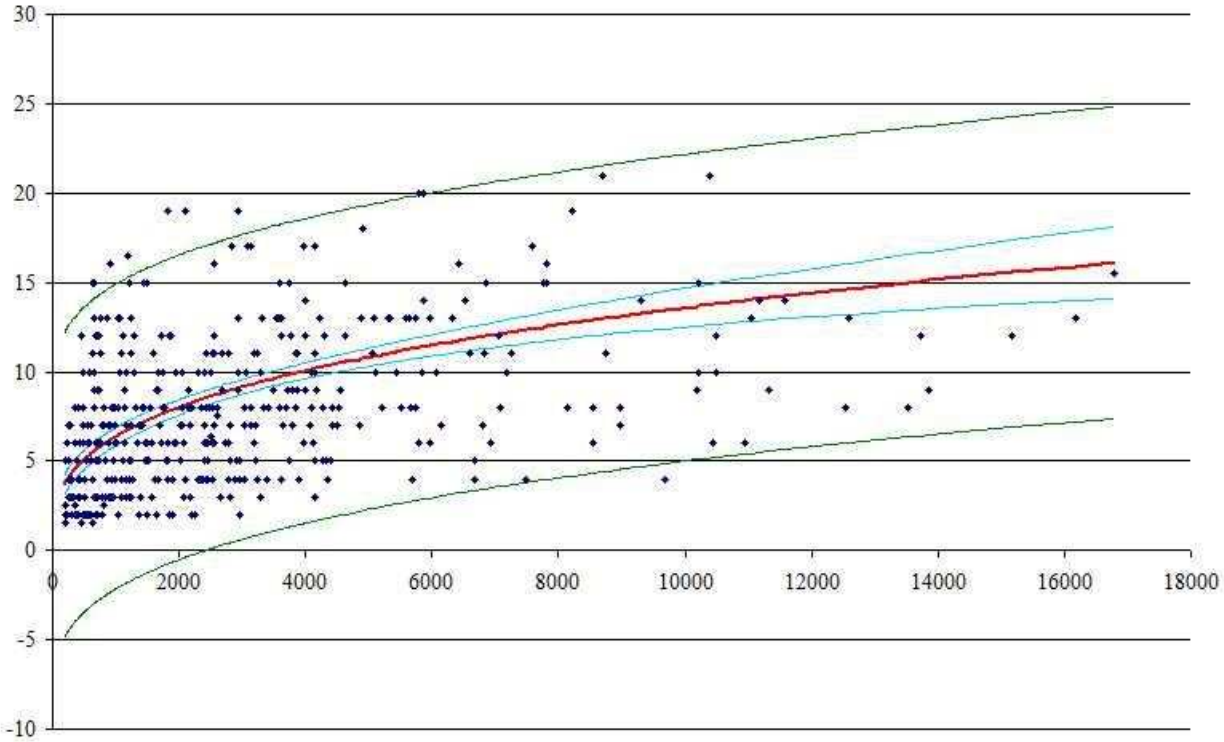

Рисунок 2.6 – Вихідні ЕД, нелінійне рівняння регресії, довірчий інтервал та інтервал прогнозування нелінійного рівняння регресії для моделі ISBSG

Як видно з рис. 2.5 та 2.6, не всі ЕД знаходяться в середині інтервалів прогнозування. Тобто, ці ЕД не підпорядковуються нормальному закону розподілу та містять викиди, на відміну від рис. 2.4.

Порівняємо побудовані регресійні моделі: модель на основі нормалізуючого перетворення Джонсона, модель COCOMO та модель ISBSG. Для порівняння моделей використаємо коефіцієнт детермінації R^2, формула (2.10), значення MMRE, формула (2.12) та значення Pred(0,25), формула (2.13). Всі ці значення наведено в табл. 2.3.

Таблиця 2.3 – Порівняння параметрів якості побудованих регресійних моделей

Параметр	Модель на основі нормалізуючого перетворення Джонсона	Модель COCOMO	Модель ISBSG
R^2	0,7058	0,5812	0,5896
MMRE	0,2019	0,4562	0,5691
Pred(0,25)	0,7592	0,3662	0,3732

Як видно із наведених даних, модель на основі нормалізуючого перетворення Джонсона, має кращі параметри, ніж модель COCOMO або модель ISBSG.

Порівняємо отримані значення для довірчих інтервалів та інтервалів прогнозування побудованих нелінійних моделей. Зведені результати для довірчих інтервалів наведено у таблиці 2.4, у якій приведено фрагмент даних.

Таблиця 2.4 – Довірчі інтервали рівнянь регресії побудованих нелінійних моделей (фрагмент даних)

№ п/п	Модель на основі нормалізуючого перетворення Джонсона			Модель COCOMO			Модель ISBSG		
	нижня границя	верхня границя	довжина	нижня границя	верхня границя	довжина	нижня границя	верхня границя	довжина
51	4,2943	5,1871	0,8928	3,7035	4,7637	1,0602	4,8865	5,9413	1,0547
52	10,6046	12,8784	2,2738	11,2629	13,4973	2,2344	12,5580	14,7808	2,2228
53	5,0593	5,9017	0,8424	4,4941	5,4982	1,0041	5,7463	6,7453	0,9989
54	7,0223	7,9812	0,9589	6,9428	7,7770	0,8342	8,3108	9,1408	0,8299
55	5,0842	5,9254	0,8411	4,5214	5,5234	1,0020	5,7757	6,7725	0,9968
56	8,8076	10,3400	1,5324	9,3644	10,5155	1,1511	10,7373	11,8825	1,1451
57	6,9099	7,8465	0,9366	6,7879	7,6275	0,8396	8,1521	8,9874	0,8353
58	8,1807	9,4749	1,2942	8,5492	9,4826	0,9334	9,9320	10,8606	0,9286
59	4,5101	5,3874	0,8774	3,9174	4,9633	1,0460	5,1210	6,1616	1,0406
60	8,2328	9,5457	1,3129	8,6192	9,5661	0,9468	10,0017	10,9437	0,9419
61	5,3252	6,1562	0,8309	4,7901	5,7712	0,9811	6,0636	7,0396	0,9760
62	3,8524	4,7748	0,9224	3,2887	4,3742	1,0855	4,4270	5,5069	1,0799
63	8,6209	10,0797	1,4588	9,1285	10,2023	1,0739	10,5055	11,5739	1,0683
64	3,3560	4,3010	0,9449	2,8637	3,9718	1,1081	3,9493	5,0517	1,1024
65	3,4579	4,3996	0,9417	2,9470	4,0510	1,1040	4,0436	5,1418	1,0983
66	8,0935	9,3569	1,2635	8,4311	9,3440	0,9129	9,8144	10,7225	0,9081
67	3,7572	4,6850	0,9279	3,2036	4,2939	1,0903	4,3319	5,4166	1,0847
68	5,7519	6,5775	0,8256	5,2886	6,2297	0,9411	6,5929	7,5291	0,9362
69	6,9712	7,9197	0,9485	6,8722	7,7087	0,8365	8,2386	9,0707	0,8322
70	10,9315	13,3348	2,4032	11,5287	13,9870	2,4583	12,8059	15,2515	2,4456

Як видно із таблиці 2.4, усі нижні границі довірчих інтервалів більші 0. Однак довжини довірчих інтервалів для моделі на основі нормалізуючого перетворення Джонсона менші, ніж для моделей COCOMO та ISBSG для 223 наборів даних, або приблизно 52% даних.

Зведені результати для інтервалів прогнозування приведені у таблиці 2.5. У таблиці 2.5 наведено фрагмент даних.

Таблиця 2.5 – Інтервали прогнозувань рівнянь регресії побудованих нелінійних моделей (фрагмент даних)

№ п/п	Модель на основі нормалізуючого перетворення Джонсона			Модель COCOMO			Модель ISBSG		
	нижня границя	верхня границя	довжина	нижня границя	верхня границя	довжина	нижня границя	верхня границя	довжина
51	1,5540	14,6822	13,1282	-4,3378	12,8051	17,1430	-3,1133	13,9410	17,0543
52	3,3249	20,6036	17,2788	3,7524	21,0078	17,2554	5,0863	22,2525	17,1662
53	1,6877	15,8189	14,1312	-3,5736	13,5659	17,1396	-2,2797	14,7713	17,0509
54	2,1020	18,0332	15,9311	-1,2053	15,9252	17,1305	0,2049	17,2467	17,0419
55	1,6923	15,8525	14,1602	-3,5474	13,5921	17,1395	-2,2513	14,7995	17,0508
56	2,6239	19,5160	16,8921	1,3656	18,5144	17,1488	2,7798	19,8399	17,0601
57	2,0744	17,9247	15,8503	-1,3577	15,7730	17,1307	0,0487	17,0908	17,0421
58	2,4223	19,0455	16,6231	0,4481	17,5837	17,1356	1,8728	18,9198	17,0469
59	1,5907	15,0260	13,4353	-4,1307	13,0114	17,1421	-2,8854	14,1680	17,0534
60	2,4383	19,0866	16,6483	0,5245	17,6608	17,1363	1,9489	18,9965	17,0477
61	1,7368	16,1677	14,4308	-3,2885	13,8497	17,1383	-1,9732	15,0764	17,0496
62	1,4805	13,9057	12,4253	-4,7408	12,4037	17,1445	-3,5610	13,4949	17,0559
63	2,5617	19,3813	16,8196	1,0935	18,2373	17,1438	2,5121	19,5673	17,0551
64	1,4002	12,8874	11,4872	-5,1552	11,9908	17,1460	-4,0281	13,0292	17,0573
65	1,4165	13,1114	11,6949	-5,0738	12,0719	17,1457	-3,9358	13,1212	17,0570
66	2,3959	18,9758	16,5799	0,3203	17,4548	17,1345	1,7455	18,7914	17,0458
67	1,4649	13,7237	12,2588	-4,8237	12,3212	17,1449	-3,6538	13,4024	17,0562
68	1,8194	16,6890	14,8696	-2,8088	14,3272	17,1360	-1,4626	15,5847	17,0474
69	2,0894	17,9841	15,8947	-1,2748	15,8558	17,1306	0,1337	17,1756	17,0420
70	3,4745	20,7660	17,2916	4,1149	21,4008	17,2858	5,4305	22,6269	17,1964

Як видно із таблиці 2.5, нижні границі інтервалів прогнозування для моделей COCOMO та ISBSG мають від`ємні значення: для 320 наборів даних або приблизно 75% даних для моделі COCOMO та для 238 наборів даних або приблизно 56% даних для моделі ISBSG.

Довжини інтервалів прогнозування для моделі на основі нормалізуючого перетворення Джонсона менші, ніж для моделей COCOMO та ISBSG: для 410 наборів даних, або приблизно 96% даних для моделі COCOMO та для 395 наборів даних або приблизно 93% даних для моделі ISBSG.

3 ПРОЕКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ ВИКОНАННЯ РОБІТ З РОЗРОБКИ ВЕБ-ЗАСТОСУНКІВ, РЕАЛІЗОВАНИХ МОВОЮ JAVA

3.1 Ескізний проект програмного забезпечення

3.1.1 Моделювання як засіб представлення процесу розробки ПЗ

Моделювання часто використовується при розробці складних інженерних систем. Велика кількість програм нині не поступаються своєю складністю інженерним спорудам, що підтверджує необхідність моделювання програмних систем.

В таких концепціях, як MDA (Model Driven Architecture) та MDD (Model Driven Development) моделям належить провідна роль у процесі розробки ПЗ. Їх головною метою є відображення процесу створення ПЗ таким чином, щоб можна було простежити послідовність трансформацій від початкової моделі до завершеної програмної системи.

Мова UML має місце у більшості інструментальних засобах, які засновані на принципах MDD. UML розшифровується як Unified Modeling Language, що означає «уніфікована мова моделювання».

UML – це мова, призначена для візуалізації, специфікації, конструювання й документування програмних систем. Крім того її можна використати для моделювання різного роду програм, починаючи із вбудованих систем і систем реального часу до розподілених web-додатків. Чіткі засоби UML дають можливість зобразити систему з усіх точок зору, які пов’язані з процесом розробки та подальщої експлуатації [19].

Мова UML призначена для рішення наступних завдань:

– надати в розпорядження користувачів готову до використання виразну потужну мову візуального моделювання, що дозволяє розробляти осмислені моделі й обмінюватися ними;

– передбачити внутрішні механізми розширюваності й спеціалізації базових концепцій мови;

– забезпечити максимальну незалежність проекту створення програмного забезпечення від конкретних мов програмування й процесів розробки;

– забезпечити формальну основу для однозначної інтерпретації мови;

– стимулювати розширення ринку об’єктно-орієнтованих інструментальних засобів створення програмного забезпечення;

– інтегрувати кращий практичний досвід використання мови й реалізації програмних засобів його підтримки.

3.1.2 Розробка моделі варіантів використання

Для графічного відображення майбутнього ПЗ побудуємо UML діаграму, а саме – діаграму варіантів використання (з англ. – Use Case Diagram). В ній зобразимо вимоги до майбутньої системи, опишемо її внутрішній устрій.

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

Актором називається деякий об'єкт, суб'єкт чи система, яка взаємодіє з модельованої бізнес-системою з метою досягнення своїх цілей чи вирішення певних завдань. Це може бути людина, технічний пристрій, програма або будь-яка інша система, яка служить джерелом впливу на модельовану систему [20].

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

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

Модель варіантів використання представлена на рисунку 3.1. На цій діаграмі варіантів використання виділено 1 актора та відповідні прецеденти.

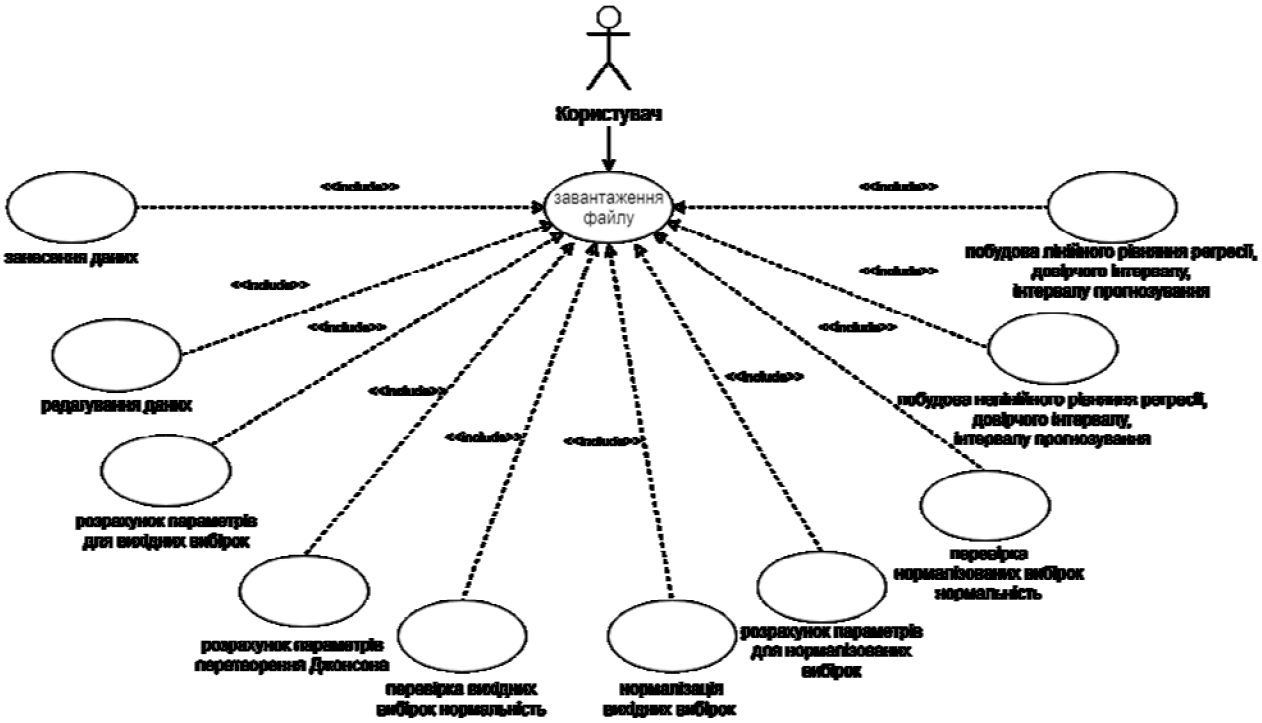

Рисунок 3.1 – Діаграма варіантів використання ПЗ

Функції користувача: завантажує файл зі статистичними даними у програму, заносить дані, редагує дані, розраховує параметри для вихідних та нормалізованих вибірок, розраховує параметри перетворення Джонсона, нормалізує вихідні вибірки, перевіряє вихідні та нормалізовані вибірки на нормальність розподілу, будує лінійне та нелінійне рівняння регресії, довірчий інтервал та інтервал прогнозування для них.

3.1.3 Розробка специфікації варіантів використання

Потік подій варіанта використання має такі складові:

- короткий опис;
- передумови;
- основний потік подій;
- альтернативний потік подій;
- постумови.

Опис варіантів використання системи наведено у табл. 3.1.

Таблиця 3.1 – Опис варіантів використання системи

Опис	Складова
1	2
Завантаження файлу	
Короткий опис	Дана функція дозволяє особі завантажити файл із вихідними статистичними даними.
Передумови	Система запущена та виводить початкову форму програми.
Основний потік подій	1. Система виводить форму для завантаження файлу. 2. Користувач вибирає файл та підтверджує введення; 3. Система перевіряє допустимість формату файлу; 4. Якщо формат вірний то зчитуються дані вихідних вибірок.
Альтернативний потік подій	1. Система виводить форму для завантаження файлу; 2. Користувач вибирає файл та підтверджує введення; 3. Система перевіряє допустимість формату файлу; 4. Повідомлення про те, що формат не вірний; 5. Надання можливості повторного завантаження файлу.
Постумови	Зчитуються дані вихідних вибірок із файлу.
Занесення даних	
Короткий опис	Дана функція дозволяє особі змінити допустимі поля з даними
Передумови	Система запущена, поле для вводу доступне для редагування
Основний потік подій	1. Користувач вводить дані 2. Система обробляє нові дані 3. Система перевіряє коректність введених даних; 4. Система приймає нові дані

Продовження табл. 3.1

1	2
Альтернативний потік подій	1. Користувач вводить дані 2. Система обробляє нові дані. 3. Система перевіряє коректність введених даних; 4. Повідомлення про те, що формат не вірний; 5. Надання можливості повторного введення
Постумови	Оновлення даних
Редагування даних.	
Короткий опис	Дана функція дозволяє особі змінити допустимі поля з даними
Передумови	Система запущена, поле для вводу доступне для редагування
Основний потік подій	1. Користувач вводить дані 2. Система обробляє нові дані 3. Система перевіряє коректність введених даних; 4. Система приймає нові дані
Альтернативний потік подій	1. Користувач вводить дані 2. Система обробляє нові дані. 3. Система перевіряє коректність введених даних; 4. Повідомлення про те, що формат не вірний; 5. Надання можливості повторного введення
Постумови	Оновлення даних
Розрахунок параметрів для вихідних даних.	
Короткий опис	Дана функція розраховує такі параметри для вихідних вибірок: (кількість значень вибірки; вибіркове середнє; дисперсія; середньоквадратичне відхилення; асиметрія; квадрат асиметрії;ексцес).
Передумови	Користувач вибрав коректний формат файлу та натиснув кнопку розрахувати
Основний потік подій	1. Програма розраховує параметри; 2. Система виводить розраховані параметри;
Альтернативний потік подій	Відсутній.
Постумови	Вивід розрахованої інформації, запис інформації у вихідний файл
Перевірка вихідних вибірок на нормальність.	
Короткий опис	Призначений для перевірки вихідних даних на нормальність розподілу
Передумови	Користувач вибрав коректний формат файлу.
Основний потік подій	1. Користувач натиснув кнопку «Перевірити на нормальність» 2. Програма аналізує нормальність розподілу; 3. Система виводить повідомлення про нормальність розподілу та відображає розраховані параметри.
Постумови	Вивід розрахованої інформації.
Розрахунок параметрів перетворення Джонсона. Нормалізація вихідних вибірок.	
Короткий опис	Призначений для пошуку оптимальних параметрів перетворення Джонсона та нормалізації вихідних даних.

Продовження табл. 3.1

1	2
Передумови	Користувач вибрав коректний формат файлу, розрахував початкові параметри вихідної вибірки
Основний потік подій	1. Користувач натискає кнопку «Нормалізувати» 2. Система виводить нове вікно роботи з програмою; 3. Користувач вводить з клавіатури параметри γ, η, ϕ, λ для початкового приближення; 4. Користувач натискає кнопку «Розрахувати»; 5. Система обробляє інформацію, розраховує оптимальні параметри γ, η, ϕ, λ. 6. Система виводить параметри нормалізації γ, η, ϕ, λ та нормалізовані значення, записує у вихідний файл.
Альтернативний потік подій	1. Користувач натискає кнопку «Нормалізувати» 2. Система виводить нове вікно роботи з програмою; 3. Користувач не ввів з клавіатури параметри γ, η, ϕ, λ для початкового приближення; 4. Користувач натискає кнопку «Розрахувати»; 5. Система виводить повідомлення про те, що потрібно ввести параметри для початкового приближення; 6. Надання можливості повторного введення.
Постумови	Вивід розрахованої інформації, запис інформації у вихідний файл
Перевірка нормалізованих вибірок на нормальність	
Короткий опис	Призначений для перевірки нормалізованих вибірок на нормальність розподілу
Передумови	Користувач вибрав коректний формат файлу.
Основний потік подій	4. Користувач натиснув кнопку «Перевірити на нормальність»; 5. Програма аналізує нормальність розподілу; 6. Система виводить повідомлення про нормальність розподілу та відображає розраховані параметри.
Постумови	Вивід розрахованої інформації.
Побудова лінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування.	
Короткий опис	Призначений для побудови лінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування.
Передумови	Користувач вибрав коректний формат файлу, провів нормалізацію вихідних вибірок
Основний потік подій	1. Користувач натиснув кнопку «Перейти до регресії»; 2. Відкривається нове вікно програми; 3. Користувач натиснув кнопку «Розрахувати» на вкладці «Лінійне рівняння регресії», а потім натиснув «Побудувати графік»; 4. Програма розраховує коефіцієнти рівняння регресії; 5. Система виводить параметри рівняння регресії, будує графік для рівняння регресії, довірчого інтервалу та інтервалу прогнозування.
Постумови	Вивід розрахованої інформації та графіка, запис інформації у вихідний файл

Продовження табл. 3.1

1	2
Побудова нелінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування	
Короткий опис	Призначений для побудови нелінійного рівняння регресії, довірчого інтервалу та інтервалу прогнозування.
Передумови	Користувач вибрав коректний формат файлу, провів нормалізацію вихідних вибірок
Основний потік подій	1. Користувач натиснув кнопку «Перейти до регресії»; 2. Відкривається нове вікно програми; 3. Користувач натиснув кнопку «Розрахувати» на вкладці «Нелінійне рівняння регресії», а потім натиснув «Побудувати графік»; 4. Програма розраховує коефіцієнти рівняння регресії; 5. Система виводить параметри рівняння регресії, будує графік для рівняння регресії, довірчого інтервалу та інтервалу прогнозування.
Постумови	Вивід розрахованої інформації та графіка, запис інформації у вихідний файл

3.1.4 Побудова концептуальної моделі даних

Концептуальна модель – модель предметної області, що складається з переліку взаємопов'язаних понять, що використовуються для опису цієї області, разом з властивостями й характеристиками, класифікацією цих понять, за типами, ситуацій, ознаками в даній області і законів протікання процесів в ній.

Основні елементи концептуальної моделі:

– умови функціонування об'єкта, визначені характером взаємодії між об'єктом і його оточенням, а також між елементами об'єкта;

– мета дослідження об'єкта та напрямок покращення його функціонування;

– можливості керування об'єктом, визначення складу керованих змінних об'єкта.

Діаграма «сутності – зв’язки» виступає засобом опису схеми бази даних на концептуальному рівні проектування. На діаграмах концептуального рівня сутності зображуються прямокутниками, атрибути – еліпсами, зв’язки – ромбами.

Але у нашому випадку ПЗ є програмою такого типу структури, яке не потребує функціональності бази даних. Усі вхідні дані програма буде отримувати за допомогою завантаження файлу та зчитування з нього.

Після розрахунків усі отримані параметри будуть записуватись у вихідний файл у форматі CSV(Character-separated values).

CSV – це текстовий файл, в якому міститься інформація. Кожен рядок – це окремий рядок таблиці, а стовпчики відокремлені один від іншого спеціальними символами-роздільниками (наприклад, коми). Останнім часом роздільником може бути не тільки кома, а й інші символи (пропуск, крапка з комою, табуляція та ін.).

Перевагою цього формату є те, що не дивлячись на те, що файл CVS – звичайний текстовий файл, він може використовуватись як вхідний файл у будь-яку БД, зазвичай застосовується для перенесення даних між базами даних та програмами – редакторами електронних таблиць.

3.1.5 Розробка сценаріїв варіантів використання

Поведінка об'єкта відбивається в його станах, щоб зобразити це, використовується два види діаграм: Statechart diagram (діаграма станів) і Activity diagram (діаграма активності) [21].

Нижче наведено декілька прикладів сценаріїв варіантів використання. На рисунках 3.2 – 3.3 зображено декілька сценарій варіантів використання для прецеденту «Завантаження файлу» та «Розрахунок параметрів перетворення Джонсона».

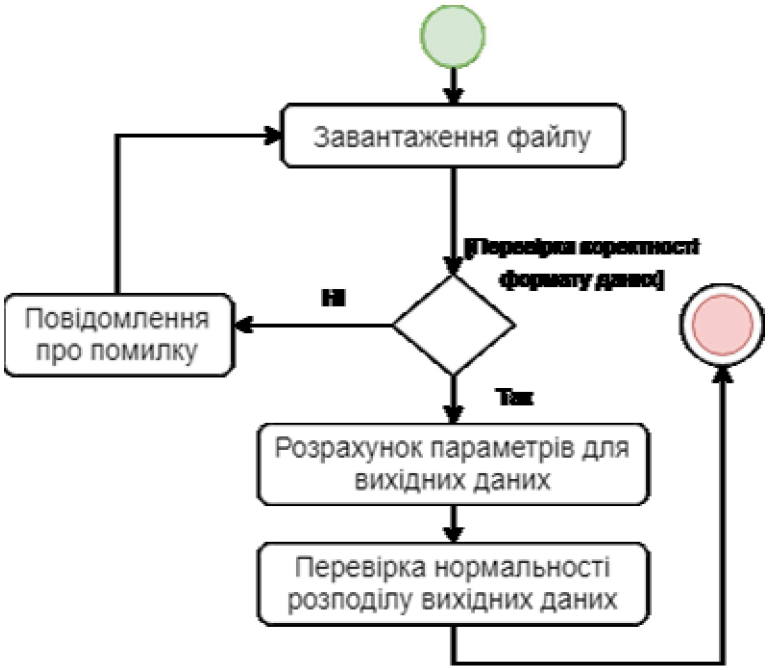

Рисунок 3.2 – Діаграма діяльності для прецеденту «Завантаження файлу»

Рисунок 3.3 – Діаграма діяльності для прецеденту «Розрахунок параметрів перетворення Джонсона»

Наведені вище діаграми сценаріїв діяльності для прецедентів гарно показують послідовність дій та події, що ініціюють дії або є кінцевим результатом.

3.1.6 Розробка прототипу користувацького інтерфейсу

Користувальницький інтерфейс – це сукупність програмних і апаратних засобів, що забезпечують взаємодію користувача з комп'ютером. Основу такої взаємодії становлять діалоги. Під діалогом у цьому випадку розуміють регламентований обмін інформацією між людиною та комп'ютером, здійснюваний в реальному масштабі часу та спрямований на спільне рішення конкретної задачі: обмін інформацією та координацію дій. Кожний діалог складається з окремих процесів введення – виведення, які фізично забезпечують зв'язок користувача і комп'ютера [22].

Отже, розробимо графічний інтерфейс нашого майбутнього ПЗ.

На рисунках 3.4 – 3.6 представлено прототип користувацького інтерфейсу ПЗ. На рисунку 3.4 зображено прототип початкового вікна програми.

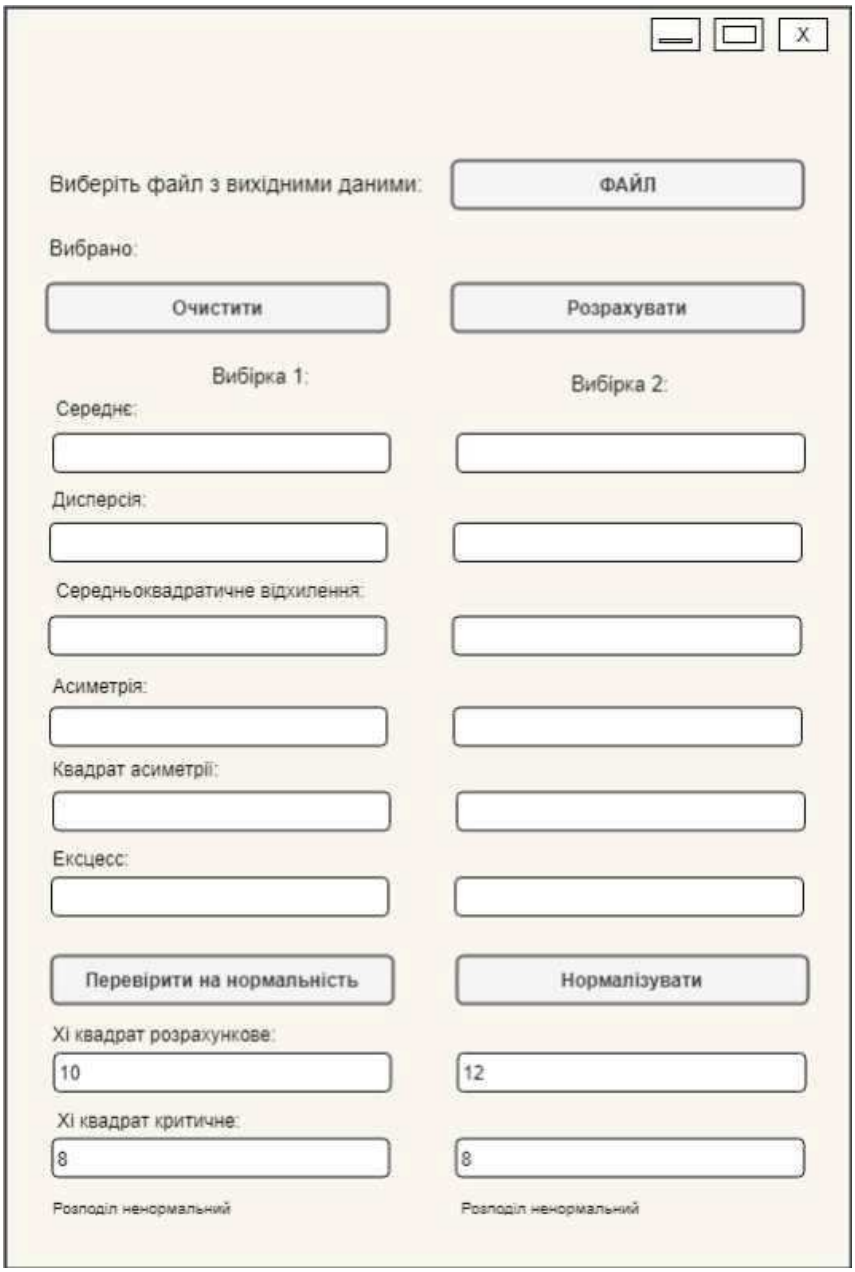

Рисунок 3.4 – Прототип початкового вікна програми

Після розрахунку початкових параметрів буде можливість перевірити нормальність розподілу та перейти далі до нормалізації, ескіз відповідної форми наведено на рис. 3.5.

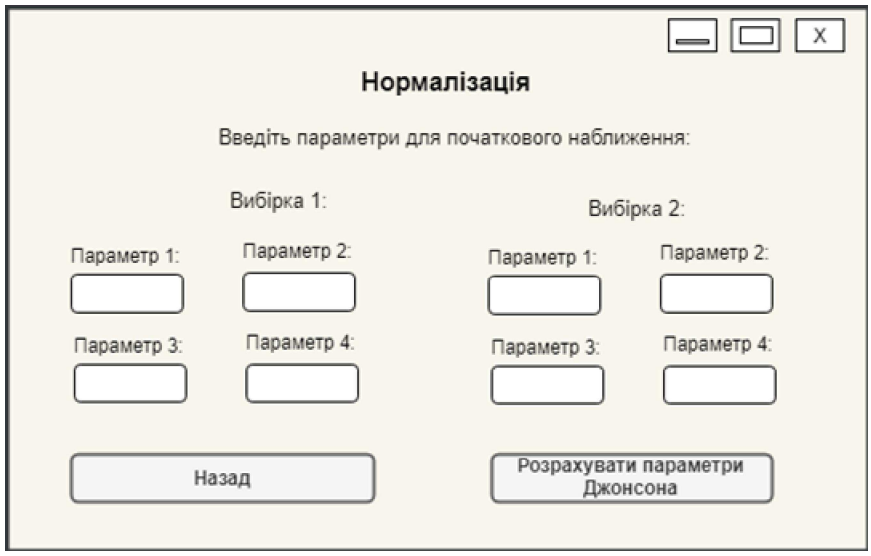

Рисунок 3.5 – Ескіз форми «Розрахунок параметрів нормалізації»

Після нормалізації переходимо до регресії та відкривається нове вікно з двома вкладками для рівнянь регресії: лінійного та нелінійного. Прототип користувацького інтерфейсу для вкладки «Лінійне рівняння» наведено на рисунку 3.6.

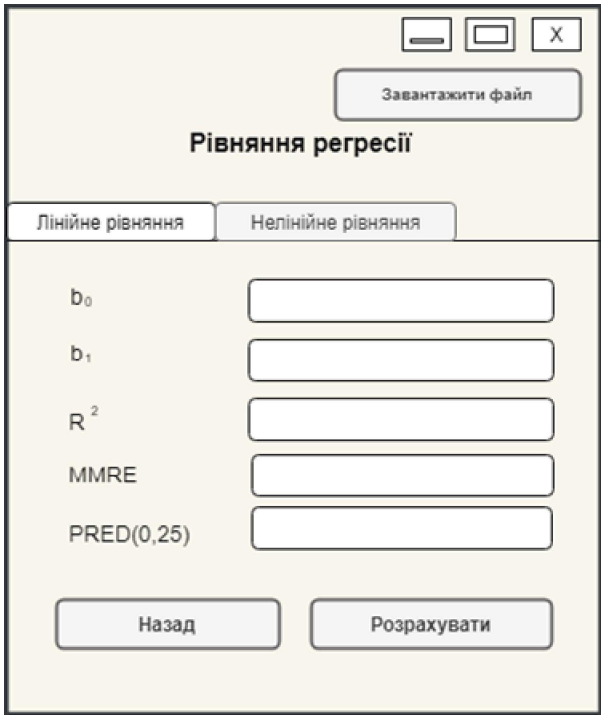

Рисунок 3.6 – Ескіз форми «Розрахунок параметрів лінійного рівняння регресії»

3.2 Технічний проект програмного забезпечення

3.2.1 Розробка статичної моделі програмного забезпечення

Діаграма класів – статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення. Діаграма класів може містити пакети та вкладені пакети [23]. Діаграма класів ПЗ, що розробляється, представлена на рисунку 3.7.

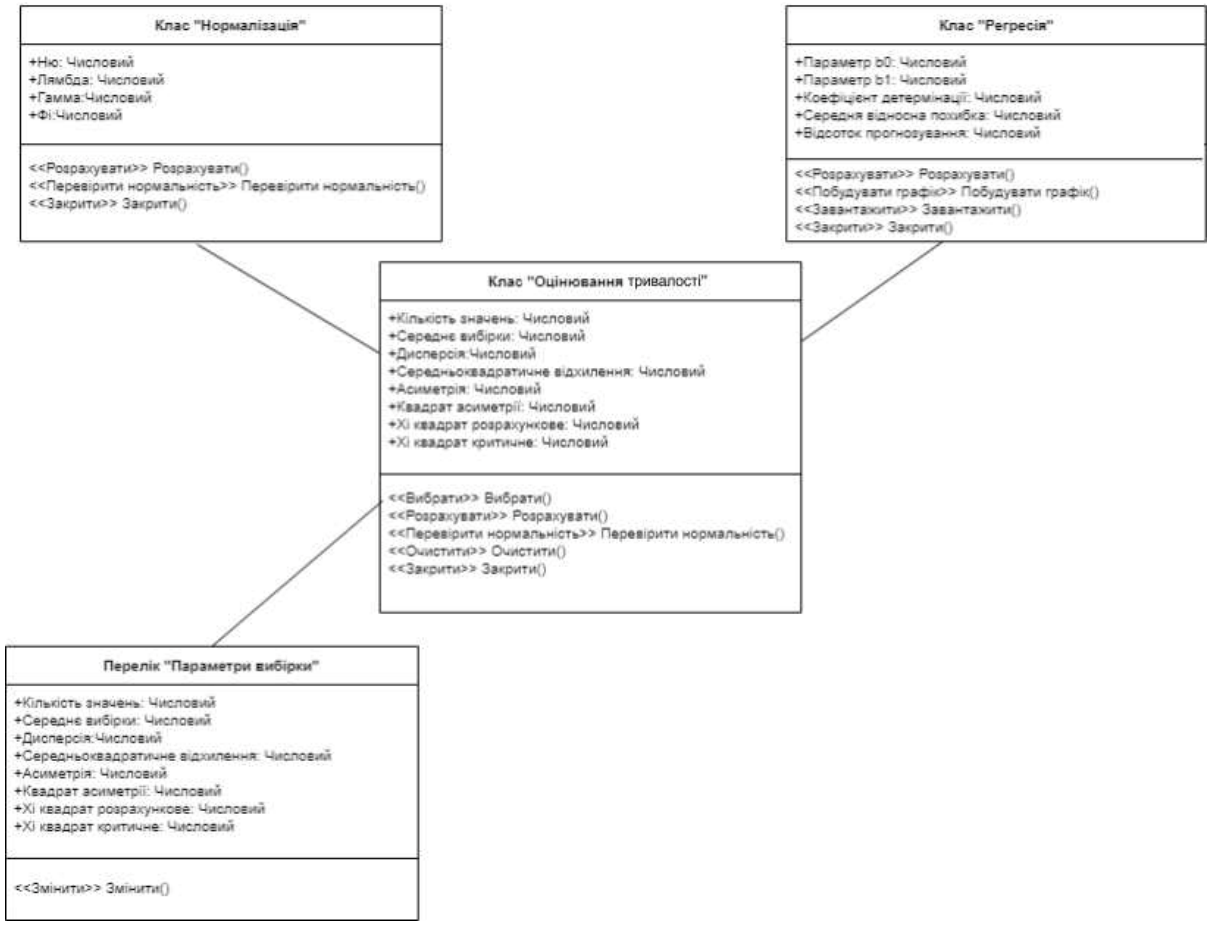

Рисунок 3.7 – Діаграма класів програмного забезпечення

На діаграмі класів відображені наступні елементи: клас «Оцінювання тривалості», клас «Нормалізація», клас «Регресія», перелік «Параметри вибірки». Вказано атрибути та їх типи, а також методи.

3.2.2 Розробка специфікації класів програмного забезпечення

Специфікація класу для об'єкта – це визначення його властивостей (які характеризують стан об'єкта) і методів (які є способом реалізації його поведінки). Специфікація класів ПЗ представлена в таблиці 3.2.

Таблиця 3.2 – Специфікація класів ПЗ

Опис	Складова
Оцінювання розміру	
Відповідальність	Даний клас відповідає за оцінювання розміру вибірок.
Атрибут	Кількість значень, середнє вибірки, дисперсія, середньоквадратичне відхилення, асиметрія, квадрат асиметрії, хі квадрат розрахункове, хі квадрат критичне
Операції зі специфікаціями нетривіальних операцій	Вибрати(): вибрати файл Розрахувати(): розрахунок параметрів вибірки Перевірити нормальність(): перевірка на нормальність розподілу Закрити(): закрити Очистити(): очистити дані
Параметри вибірки	
Відповідальність	Даний клас відповідає за інформацію про параметри вибірки.
Атрибут	Кількість значень, середнє вибірки, дисперсія, середньоквадратичне відхилення, асиметрія, квадрат асиметрії, хі квадрат розрахункове, хі квадрат критичне.
Операції зі специфікаціями нетривіальних операцій	Змінити(): змінити параметр.
Нормалізація	
Відповідальність	Даний клас відповідає за інформацію нормалізації вихідної вибірки.
Атрибут	ню, гамма, фі, лямбда.
Операції зі специфікаціями нетривіальних операцій	Розрахувати(): розрахунок параметрів Перевірити нормальність(): перевірка на нормальність розподілу. Закрити(): закрити.
Регресія	
Відповідальність	Даний клас відповідає за інформацію про лінійне та нелінійне рівняння регресії.
Атрибут	Параметр b_0, параметр b_1, коефіцієнт детермінації, середня відносна похибка, відсоток прогнозування.
Операції зі специфікаціями нетривіальних операцій	Розрахувати(): розрахунок параметрів. Побудувати графік(): побудова графіку рівняння регресії, довірчого інтервалу та інтервалу прогнозування. Завантажити(): завантажити вихідний файл.

3.2.3 Побудова динамічної моделі програмного забезпечення

Динамічна модель програми може бути представлена:

- діаграмами діяльності;
- діаграмами послідовності;
- діаграмами взаємодії;
- діаграмам станів.

Динамічна модель програми представлена у вигляді діаграми послідовності. У роботі представлені діаграми послідовності по деяким варіантам використання.

Діаграма послідовності (sequence diagram) – діаграма, на якій показані взаємодії об'єктів, упорядковані за часом їхнього прояву [24].

Нижче наведено декілька діаграм послідовності для розроблюванного ПЗ на рисунках 3.8 – 3.9

У варіанті використання «Завантаження файлу» користувач спочатку вибирає файл. Система перевіряє розширення файлу на допустимість та повертає результат. Діаграма послідовності для варіанту використання «Завантаження файлу» представлена на рисунку 3.8.

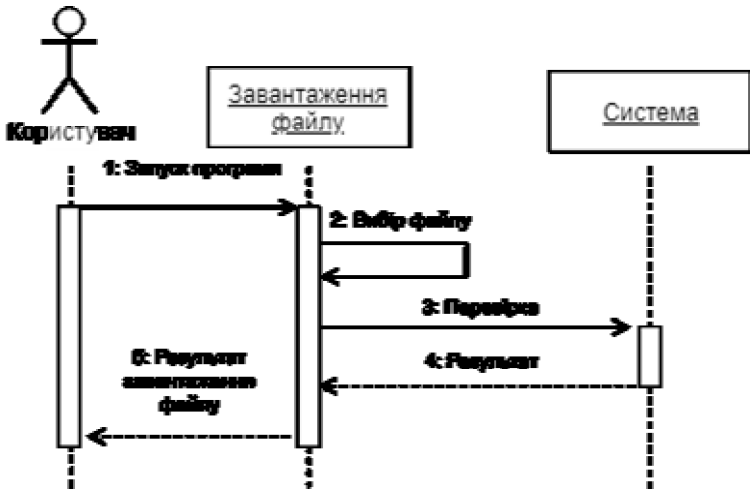

Рисунок 3.8 – Діаграма послідовності для варіанту використання «Завантаження файлу»

У варіанті використання «Розрахунок параметрів Джонсона» користувач спочатку проходить етап розрахунку параметрів вихідних вибірок. Потім вводить параметри початкового наближення. Система перевіряє наявність та допустимість введених даних. Якщо дані коректні, повертає розраховані оптимальні параметри Джонсона. Діаграма послідовності для варіанту використання «Розрахунок параметрів Джонсона» представлена на рисунку 3.9.

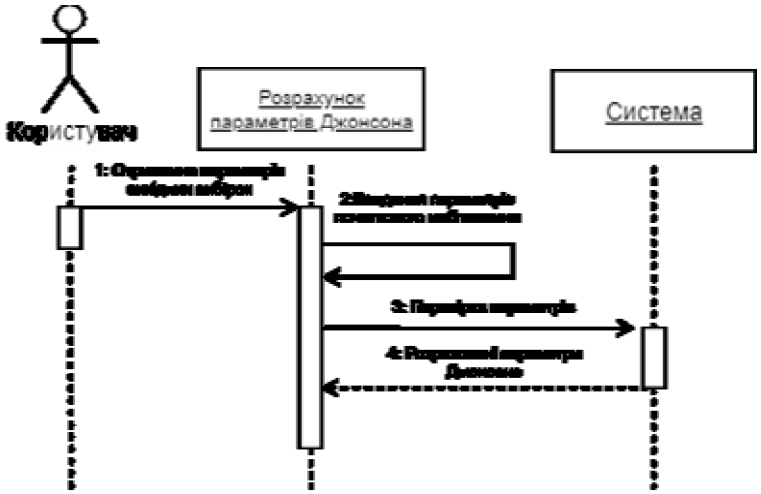

Рисунок 3.9 – Діаграма послідовності для варіанту використання «Розрахунок параметрів Джонсона»

3.3 Робочий проект програмного забезпечення

3.3.1 Обґрунтування вибору мови програмування та системи програмування

Для розробки ПЗ для оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java, було обрану мову C++ та середовище розробки Microsoft Visual Studio.

C++ – універсальна мова програмування високого рівня з підтримкою декількох парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної. Починаючи з 1990-х років і по наш день C++ є однією з найуживаніших та найпотужніших мов програмування загального призначення.

Переваги мови C++:

1. Продуктивність. Швидкість роботи програм на C++ практично не поступається програмам на C, хоча програмісти отримали в свої руки нові можливості і нові засоби.

2. Масштабованість. На мові C++ розробляють програми для самих різних платформ і систем.

3. Можливість роботи на низькому рівні з пам'яттю, адресами, портами.

4. Можливість створення узагальнених алгоритмів для різних типів даних, їх спеціалізація, і обчислення на етапі компіляції, з використанням шаблонів.

Реалізації C++ тепер є на всіх машинах, починаючи з найскромніших мікроком'ютерів і до найбільших супер-ЕОМ, і практично для всіх операційних систем.

Існують три причини, без яких неможливо написати гарну програму мовою C++ без використання показників. По-перше, показники дозволяють функціям змінювати свої аргументи. По-друге, за допомогою показників здійснюється динамічний розподіл пам'яті. По-третє, показники підвищують ефективність багатьох процедур [25].

Показники – один із самих потужних й, у той же час, самих небезпечних засобів мови C++. Наприклад, показники, що містять невірні адреси можуть знищити операційну систему комп'ютера. І, що ще гірше, неправильне використання показників породжує помилки, які вкрай важко виявити.

Показник (pointer) – це змінна, у якій зберігається адреса іншого об’єкта (як правило, іншої змінної). Наприклад, якщо один змінна містить

адреса інший змінної, говорять, що перша змінна посилається на другу. Показник ідентифікує змінну, говорячи не про її ім'я, а про те, де вона перебуває в пам'яті. [25].

Microsoft Visual Studio містить безліч інтегрованих засобів візуального програмування. Компілятор VisualC++ містить багато нових інструментальних засобів і поліпшених можливостей, надає величезні можливості в плані оптимізації додатків, внаслідок чого можна отримати виграш як відносно розміру програми, так і відносно швидкості її виконання, незалежно від того, що являє собою ваш додаток.

Система Microsoft Visual Studio дозволяє створювати як маленькі програми і утиліти для персонального використання, так і корпоративні системи, що працюють з базами даних на різних платформах.

В середовищі Visual Studio можна будувати різні типи проектів. Такі проекти після їх створення можна компілювати і запускати на виконання. Фірма Microsoft розробила спеціальний інструментарій, який полегшує і прискорює створення проектів в середовищі Visual Studio. Наприклад, майстер MFC AppWizard (exe) дозволяє створити проект, Windows-додатки які мають однодокументний, багатодокументний або діалоговий інтерфейс і використовують бібліотеку MFC (Microsoft Foundation Classes).

До складу компілятора Microsoft Developer Studio вбудовані засоби, що дозволяють програмісту полегшити розробку додатків. В першу чергу до них відносяться MFC AppWisard, ClassWizard і редактор ресурсів. Завдяки MFC AppWizard середовище розробника дозволяє швидко створювати шаблони нових додатків. При цьому програмісту не доводиться писати жодного рядка коду. Досить відповісти на ряд питань, що стосуються того, яка програма потрібна, і вихідні тексти шаблону разом з файлами ресурсів готові. Ці тексти можна відтранслювати і отримати готовий завантажувальний модуль програми.

Звісно, ніякі засоби автоматизованої розробки не зможуть створити програму повністю без участі програміста. Прикладну частину додатку

доведеться розробляти йому. Для створення ресурсів програми призначений редактор ресурсів. Він дозволяє швидко створювати нові меню, діалогові панелі, додавати кнопки до панелі управління toolbar і т.д. Засіб ClassWizard дозволяє підключити до створених і відредагованих ресурсів керуючий ними код. Велику частину роботи по опису і визначенню функцій, обробних повідомленнях від меню, органів управління діалогових панелей і т.д., також бере на себе засіб ClassWizard [26].

Отже, підсумовуючи усе вищесказане, мова С++ буде найбільш оптимальним вибором для розробки ПЗ для оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java.

3.3.2 Реалізація та тестування основних класів програмного забезпечення

Реалізація програмного забезпечення була виконана на високорівневій мові програмування С++.

Для тестування програмного забезпечення є такі основні групи методів:

- методи тестування по формальним специфікаціям (методи чорного ящика);
- метод структурного тестування (методи білого ящика).

До групи методів чорного ящика відносяться наступні методи: розбиття на класи еквівалентності та граничні умови та інші. До методів білого ящика відносяться наступні методи: покриття операторів, покриття переходів та інші.

Так як методи структурного тестування вимагають більших затрат ресурсів, то в дипломній роботі виконувалось тестування по специфікаціям, а саме розбиття на класи еквівалентності.

Програму розглядали як чорний ящик. Випробування зводилися до послідовного вводу тестових наборів даних й аналізу отриманих результатів. У нашій програмі користувач вводить дані вручну лише на одному етапі

оцінювання ПЗ: на етапі розрахунку нормалізованих параметрів Джонсона. Він повинен ввести чотири параметри для початкового наближення.

В таблиці 3.3 представлені класи еквівалентності та граничні значення вхідних даних:

– гамма;
– ню;
– фі;
– лямбда.

Таблиця 3.3 – Класи еквівалентності вхідних даних для класу нормалізація

Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
Гамма	Число	Усі крім числового типу (символьні, логічні, графічні зображення).
Ню	Число	Усі крім числового типу (символьні, логічні, графічні зображення).
Фі	Число	Усі крім числового типу (символьні, логічні, графічні зображення).
Лямбда	Число	Усі крім числового типу (символьні, логічні, графічні зображення).

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

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

№	Вхідні умови	Правильні класи еквівалентності	Вихідні дані
1	Гамма	В поле «Гамма» веденно число «0,02532».	Результат вірний.
2	Ню	В поле «Ню» веденно число «84156,04».	Результат вірний.
3	Фі	В поле «Фі» веденно число «5,12».	Результат вірний.
4	Лямбда	В поле «Лямбда» веденно число «999».	Результат вірний.

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

№	Вхідні дані	Вихідні дані
1	В поле «Гамма» введемо будь-який символ ASCII.	Програма виводить повідомлення про помилку. Результат підтверджено.
2	В поле «Ню» ведемо графічний елемент – смайлик (скопіюємо з інтернету).	Після ведення смайлика виводиться повідомлення про помилку. Програма далі працює правильно. Результат підтверджено
3	В поле «Фі» не введемо нічого.	Після ведення цифр виводиться повідомлення про помилку. Програма далі працює правильно. Результат підтверджено
4	В поле «Лямбда» введемо будь-який символ ASCII.	Програма виводить повідомлення про помилку. Результат підтверджено

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

3.3.3 Випробування програмного забезпечення

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

Випробування ПЗ проводилося на цільовому обладнанні в тій конфігурації, яка заявлена в Технічному завданні (див. Додаток А).

Всі виявлені недоліки ПЗ були зафіксовані в протоколах і усунуті розробником до моменту впровадження додатку в дію.

Випробування було проведено за стратегією «чорного ящика». За рахунок дуже великої кількості функцій, які потрібно було випробовувати, представлено тільки декілька результатів випробувань функцій програми:

Функція «Завантаження файлу з вихідними даними».

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

Функція «Перевірка вихідних вибірок на нормальність розподілу»

На головному вікні після завантаження файлу та розрахунку початкових параметрів вихідних вибірок натиснули кнопку «Перевірка на нормальність». У результаті отримали параметри хі квадрату (розрахункове та критичне, та повідомлення нижче про ненормальність розподілу вихідних вибірок).

Функція «Розрахунок параметрів одномірного перетворення Джонсона»

У вікні «Нормалізація» ввели параметри для початкового наближення та натиснули кнопку «Розрахувати параметри Джонсона». У результаті з’явилось відповідне вікно, в якому виведено розраховані параметри нормалізації та можливість перевірити нормальність розподілу нормалізованих вибірок.

4 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ ВИКОНАННЯ РОБІТ З РОЗРОБКИ ВЕБ-ЗАСТОСУНКІВ, РЕАЛІЗОВАНИХ МОВОЮ JAVA

Практичним результатом кваліфікаційної роботи є програма для оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java. Було здійснено постановку задачі на розробку ПЗ, розроблено ескізний, технічний та робочий проекти ПЗ.

Згідно з вимогами до ПЗ, наведеними в технічному завданні (Додаток А), програмне забезпечення надає такі функції:

– можливість завантажити файл з вихідними даними у форматах Excel (.xml, .xls, .xlt, .xlsx, .xlsm, .xlsb, .xltm, .xltx, .xlam);

– можливість змінити файл з вихідними даними;

– зчитування даних з файлу;

– розрахунок параметрів для вихідних вибірок;

– розрахунок параметрів одновимірного перетворення Джонсона;

– нормалізація вихідних вибірок за допомогою одновимірного нормалізую чого перетворення Джонсона;

– розрахунок параметрів для нормалізованих вибірок;

– перевірка вихідних та нормалізованих вибірок на нормальність розподілу;

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

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

– оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java;

– експорт результатів у файл.

ПЗ повністю відповідає вимогам до організації вхідних та вихідних даних, вимогам до надійності та іншим вимогам, зазначеним в технічному

завданні (Додаток А).

Також була розроблена наступна програмна документація:

- технічне завдання (Додаток А);
- опис програми (Додаток Б)
- текст програми (Додаток В);
- інструкція користувача (Додаток Г);
- програма та методика випробувань ПЗ (Додаток Д).

ПЗ було розроблено на мові програмування C++ в середовищі Microsoft Visual Studio.

Створене ПЗ було протестоване на відповідність технічному завданню. Під час тестування ПЗ показало повну працездатність.

Розроблене ПЗ може бути використано користувачем, не здатним до програмування. Інтерфейс є інтуїтивно зрозумілим.

5 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ РОЗРОБКИ ТА ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ ВИКОНАННЯ РОБІТ З РОЗРОБКИ ВЕБ-ЗАСТОСУНКІВ, РЕАЛІЗОВАНИХ МОВОЮ JAVA

5.1 Опис програмного забезпечення

Успіх програмних проектів залежить від якості даних, на основі яких приймаються рішення під час стратегічного та тактичного планування. Останнім часом для зменшення ризиків керування проектами по розробці ПЗ використовуються різні методології розробки.

Одним із найважливіших факторів, що впливають на прийняття рішення по пріоритезації задач, включення задач до backlog і т.д. є оцінювання часу виконання робіт.

Основними методами, які використовуються для оцінювання тривалості виконання робіт з розробки ПЗ, є параметричне оцінювання (нелінійні регресійні моделі COCOMO та ISBSG), оцінювання за трьома точками (метод PERT, який базується на бета-розподілі). Однак вказані моделі не завжди адекватно враховують розподіл ЕД про тривалість та трудомісткість робіт з розробки веб-застосунків, реалізованих мовою Java. Тому задача удосконалення методів оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java, є актуальною.

У зв’язку з вузькою спеціалізацією питання не існує готового ПЗ, яке б в повній мірі вирішувало всі необхідні задачі. Впровадження ПЗ для оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java, перш за все націлене на зменшення витрат та підвищення ефективності роботи.

Створення ПЗ вимагає одноразових витрат на розробку, придбання необхідних технічних засобів і поточних витрат на функціонування.

Економія від функціонування ПЗ визначається з урахуванням витрат на його експлуатацію. Відношення цієї економії до витрат на створення ПЗ характеризує економічну ефективність капітальних вкладень. Економічні показники визначаються за діючими на момент розрахунку оптовими цінами, тарифами і ставками заробітної плати.

5.2 Розрахунок витрат на створення та експлуатацію програмного забезпечення

Витрати на розробку системи складаються з витрат на заробітну плату розробника, на амортизацію комп`ютера, на якому виконується розробка, на експлуатацію цього комп`ютера, на засоби розробки та витратні матеріали і комплектуючі.

Розробка ПЗ виконується програмістом, місячний оклад якого складає 8000 грн. Додаткова заробітна плата складає 20% від основної – 1600 грн. Виходячи з цього, основна і додаткова заробітна плата розроблювача системи складає 9600,00 грн/міс, вартість сучасного комп`ютера складає 12000 грн. (приведена вартість машини на базі Intel Core i5). При вартості кіловат-години електроенергії рівної 1,68 грн., розраховується вартість розробки програми. Витрати на допоміжні матеріали приведені в табл. 4.1

Таблиця 4.1 – Витрати на допоміжні матеріали

Пункти витрат	Сума грн.
Папір	90,00
Заправлення картриджа до принтера	130,00
Диск	12,00
Література	300,00
Непередбачені витрати	50,00
Разом матеріали і комплектуючі вироби	582,00

Вартість програми розрахуємо по формулі:

$$Спр = (Взп + Всф + Взг + Ве) * Т + Вм \quad (4.1)$$

де: Взп – основна і додаткова заробітна плата обслуговуючого персоналу, грн.;

Всф – відрахування в соціальні фонди (38% від основної і додаткової заробітної плати), грн.;

Взг – загальногосподарські витрати (10% від основної заробітної плати), грн.;

Ве – витрати на електроенергію;

Т – тривалість розробки, міс.

Вм – витрати на основні і допоміжні матеріали.

Ве при споживанні потужності 500 Вт, тривалості роботи на місяць, рівної 21*8 = 168 годин, вартості кіловат-години електроенергії 1,68 грн.

$$Ве = 168 * 1,68 * 0,5 = 141,12 \text{ грн.}$$

Загальні витрати на розробку системи наведені в табл. 4.2.

Таблиця 4.2 – Витрати на розробку системи

Найменування витрат	Одиниця виміру	Кількість
Тривалість розробки	Міс	3
Основна і додаткова заробітна плата	Грн.	9600,00
Відрахування в соціальні фонди	Грн.	3648,00
Загальногосподарські витрати	Грн.	960,00
Витрати на допоміжні матеріали	Грн.	582,00
Витрати на електроенергію	Грн.	141,12

Відповідно до формули 4.1 вартість програми:

$$Спр = (9600,00 + 3648,00+ 960,00+ 141,12) * 3 + 582,00=43629,36 \text{ грн.}$$

Експлуатаційні витрати для комп`ютера розрахуємо по формулі:

$$Взр = Аоб + Вм + Ве \quad (4.2)$$

де: Аоб – амортизаційні відрахування, грн.

Вм – річні витрати на основні і допоміжні матеріали, грн.

Ве – витрати на електроенергію, грн.

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

$$Аоб = 12000 * 0,6 = 7200,00 \text{ грн.}$$

У масштабах закладу річні витрати на основні і допоміжні матеріали (носії, папір, …) визначаються в розмірі 5% вартості основного устаткування:

$$Вм = 12000 * 0,05 = 600,00 \text{ грн.}$$

Річний обсяг робіт комп`ютера у годинах визначається в такий спосіб:

$$Фм = 253,3 * Тз \quad (4.3)$$

де Тз – це середнє денне завантаження устаткування (близько 7 годин)

253,3 – середня кількість робочих днів у році.

Отже, річний обсяг роботи комп`ютера складе:

$$Фм = 253,3 * 7 = 1773,1 \text{ годин}$$

Витрати на електроенергію Ве при 1773,1 годинах роботи устаткування в рік складуть

$$Ве = 1773,1* 1,68 * 0,5 = 1489,41 \text{ грн.}$$

Експлуатаційні витрати для комп`ютера за рік складуть:

$$Взр = 7200,00 + 600,00 + 1489,41= 9289,41 \text{ грн.}$$

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

$$Все = 43629,36 + 9289,41 = 52918,77 \text{ грн.}$$

5.3 Економічна ефективність розробки та впровадження програмного забезпечення

Основним показником економічної ефективності функціонування ПЗ є зменшення витрат, підвищення ефективності роботи та удосконалена ступень захисту.

До числа найголовніших факторів, що визначають ефективність роботи в зв'язку з впровадженням ПЗ, відносяться:

– підвищення продуктивності праці;

– підвищення швидкості обробки інформації;

– вивільнення робочого часу;

– умовно-річна економія від підвищення якості захисту інформації.

Розрахуємо пряму економічну ефективність від використання ПЗ, ґрунтуючись на тому, що впровадження програмного продукту зменшує кількість працівників (за експертною оцінкою фахівців установи близько двох осіб), яких потрібно було б задіяти для обробки інформації на старому ПЗ.

Зарплата 2 працівників в рік складає:

$$8000 * 12 * 2 = 192000 \text{ грн.}$$

Економічний ефект першого року розраховується за формулою (4.3):

$$A_{экон.эф.} = \Delta C_n - E_n * k, \quad (4.3)$$

де ΔC_n – вивільнені кошти після впровадження програмного продукту (192000 грн.) мінус експлуатаційні витрати (9289,41 грн.) ;

$\Delta C_n = 192000 - 9289{,}41 = 182710{,}59$ грн.

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

k – одноразові витрати на впровадження продукту (43629,36 грн).

$$A_{экон.эф.} = 182710{,}59 - 0{,}6 * 43629{,}36 = 156532{,}97 \text{ грн.}$$

Строк окупності системи розраховується по формулі:

$$T = \frac{k}{П} = \frac{43629,36}{182710,59} \approx 0,24 \text{ (року)}$$

де П – вивільнені кошти після впровадження програмного продукту (182710,59 грн.);

k – одноразові витрати на впровадження системи (43629,36 грн.)

Отже строк окупності системи складає приблизно 2,9 місяці.

5.4 Висновки

Економічний ефект проявляється в отриманні економії витрат шляхом зменшення витрат, підвищення ефективності роботи та підвищенні якості оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java.

З обрахованих вище даних видно, що програмний продукт з точки зору річних експлуатаційних даних може мати практичне застосування.

Реалізація даного програмного продукту є доцільною, як бачимо з отриманих значень періоду окупності та показників ефективності.

6 ОХОРОНА ПРАЦІ

6.1 Визначення, цілі, завдання та актуальність питань, пов'язаних з охороною праці

Охорона праці – це система законодавчих актів соціально-економічних, організаційно-технічних і санітарно-гігієнічних заходів, що забезпечують безпеку, збереження здоров'я і працездатність людини в процесі праці.

Загальне керівництво роботою по охороні праці на підприємстві покладається на директора і головного інженера, а безпосереднє керівництво й організацію цієї роботи здійснює особисто головний інженер, а на великих підприємствах – його заступник по охороні праці. У їхньому підпорядкуванні знаходиться відділ охорони праці, на який покладаються наступні основні функції:

- контроль за дотриманням керівниками цехів, відділів і інших структурних підрозділів діючого законодавства, стандартів, правил і норм по охороні праці; за виконанням розпоряджень контролюючих органів і наказів по підприємству;
- розробка заходів щодо створення здорових і безпечних умов праці в цехах, відділах, на споруджуваних об'єктах;
- проведення вступного інструктажу для людей, які надходять на підприємство і контроль за своєчасним і якісним проведенням інструктажу на робочих місцях;
- участь у роботі комісій з перевірки знань інженерно-технічних працівників в області техніки безпеки і виробничої санітарії, по розслідуванню причин аварій і нещасних випадків, по розгляду проектів будівництва, капітального ремонту цехів і устаткування;
- проведення паспортизації санітарно-технічного стану цехів;

- облік потерпілих від нещасних випадків на виробництві, проведення аналізу і складання звітів про виробничий травматизм, контроль за освоєнням коштів і впровадженням заходів, спрямованих на поліпшення умов праці;
- організація виставок і стендів по охороні праці.

Поліпшення умов праці сприяє:

- підвищенню ефективності виробництва;
- підтримці працездатності людини на високому рівні і його якнайшвидшому відновленні;
- скороченню виробничого травматизму;
- зменшенню професійних захворювань.

6.2 Аналіз небезпечних і шкідливих факторів, що впливають на людину при роботі з персональним комп'ютером

В приміщенні, в якому передбачається робота персоналу за персональним комп'ютером, можливе виникнення наступних небезпечних і шкідливих факторів:

- підвищена небезпека поразки електричним струмом;
- підвищена пожежна небезпека;
- шкідливе електромагнітне випромінювання;
- недостатнє освітлення.

Одним з найбільш розповсюджених небезпечних виробничих факторів є електричний струм. Дія електричного струму на живу тканину має різнобічний і своєрідний характер. Проходячи через організм людини, електрострум здійснює термічний, електролітичний, механічний і біологічний вплив.

Термічна дія струму проявляється опіками окремих ділянок тіла, нагріванням до високої температури органів, розташованих на шляху струму,

викликаючи в них значні функціональні розлади. Електролітична дія виражається в розкладанні органічної рідини, у тому числі крові, у порушенні її фізико-хімічного складу. Механічна дія струму приводить до розшарування, розриву тканин організму в результаті електродинамічного ефекту, а також миттєвого вибухоподібного утворення пари з рідини. Біологічна дія струму виявляється подразненням і порушенням живих тканин організму, а також порушенням внутрішніх біологічних процесів.

Припустимим вважається струм, при якому людина може самостійно звільнитися від електричного ланцюга. Його величина залежить від швидкості проходження струму через тіло людини: при тривалості дії більш 10 с – 2 мА, при 1 менш – 6 мА. Струм, при якому потерпілий не може самостійно відірватися від струмоведучих частин називається струмом, що не відпускає, і складає 10...20 мА. Змінний струм небезпечніше постійного, при високій напрузі не безпечнішим є постійний струм.

Пожежі на обжитих людиною територіях, на підприємствах виникають у більшості випадків у зв'язку з порушенням технологічного режиму. Це, на жаль, часте явище і державою передбачені спеціальні документи, що описують основи протипожежного захисту. Це стандарти: ДСТ 12.1.004-76 "Пожежна безпека" і ДСТ 12.1.010-76 "Вибухобезпека".

Заходи щодо пожежної профілактики розділяються на організаційні, технічні, режимні й експлуатаційні.

Організаційні заходи передбачають правильну експлуатацію машин і внутрішньозаводського транспорту, правильний зміст будинків, території, протипожежний інструктаж робітників та службовців, організацію добровільні пожежних дружин, пожежно-технічних комісій, видання наказів по питаннях посилення пожежної безпеки та ін.

До технічних заходів відносяться дотримання протипожежних правил, норм при проектуванні будинків, встановленні електропроводів і устаткування, опалення, вентиляції, освітлення, правил розміщення устаткування.

Заходи режимного характеру – це заборона паління в невстановлених місцях, проведення зварювальних і інших вогневих робіт у пожежно небезпечних приміщеннях та ін.

Експлуатаційними заходами є своєчасні профілактичні огляди, ремонти й випробування технологічного устаткування.

Шкідливий вплив електромагнітного випромінювання, здійснюваного обладнанням комп‘ютерних мереж і систем, обумовлений взаємодією людини як біологічного об‘єкта з оточуючим середовищем. Для нейтралізації цього впливу застосовують спеціальні екрани чи покриття моніторів, розташовують робочі місця таким чином, щоб працівник не знаходився ззаду дисплея тощо. Крім того, при розробці інтер‘єру приміщення повинні враховуватися ергономічні вимоги, зокрема, забезпечуватись правильне розташування клавіатури відносно рук оператора і монітора відносно очей людини (не менше 80 см).

Важливим фактором, що впливає на продуктивність праці, є освітленість робочого місця. Організація раціонального освітлення робочого місця – одне з основних питань охорони праці. При незадовільному освітленні різко знижується продуктивність праці, можливі нещасливі випадки, поява короткозорості, швидка стомленість.

Основною задачею виробничого освітлення є підтримка на робочому місці освітленості, що відповідає характеру зорової роботи. Збільшення освітленості робочої поверхні поліпшує видимість об'єктів за рахунок підвищення їхньої яскравості, збільшує швидкість розрізнення деталей, що позначається на зростанні продуктивності праці.

При організації виробничого освітлення необхідно забезпечити рівномірний розподіл яскравості на робочій поверхні і навколишніх предметах. Переведення погляду з яскраво освітленої на слабко освітлену поверхню змушує око переадаптуватися, що веде до стомлення зору і, відповідно, до зниження продуктивності праці. Для підвищення рівномірності природного освітлення великих приміщень здійснюється

комбіноване освітлення. Світле фарбування стелі, стін і устаткування сприяє рівномірному розподілу яскравостей у полі зору працюючого.

Виробниче освітлення повинне забезпечувати відсутність у полі зору працюючого різких тіней. Наявність різких тіней спотворює розміри і форми об'єктів розрізнення і тим самим підвищує стомлюваність, знижує продуктивність праці. Особливо шкідливі тіні, що рухаються, які можуть привести до травм.

Для поліпшення умов видимості об'єктів у полі зору працюючого повинна бути відсутня пряма і відбита блисткість. Блисткість – це підвищена яскравість світлих поверхонь, що викликає порушення зорових функцій (засліпленість), тобто погіршення видимості об'єктів. Блисткість обмежують зменшенням яскравості джерела світла, правильним вибором типу світильника.

Коливання освітленості на робочому місці, викликані, наприклад, різкою зміною напруги в мережі, обумовлюють переадаптацію ока, приводячи до значного стомлення.

Виробниче освітлення повинне забезпечувати необхідний спектральний склад світлового потоку. Це вимога особлива істотно для забезпечення правильної передачі кольору, а в окремих випадках – для посилення колірних контрастів. Оптимальний спектральний склад забезпечує природне освітлення.

Освітлювальні установки повинні бути зручні і прості в експлуатації, довговічні, відповідати вимогам естетики, електробезпечності, а також не повинні бути причиною вибуху чи пожежі.

Штучне освітлення застосовується як у темний час доби, так і у світлий час, коли для нормальних умов по освітленості природне освітлення неприпустиме чи недостатньо неможливо.

Система штучного освітлення – це група світильників з розміщеними в них електричними джерелами світла, спроектована по визначеному принципу в залежності від виконуваних задач. При розміщенні джерел світла

над освітлюваною площею часто виникає необхідність одночасного рішення питання вибору світильників по таких характеристиках, як дальність дії, припустима висота підвісу, одинична потужність джерел світла і фону.

Системи штучного освітлення класифікуються по двох основних ознаках:

- конструктивне виконання;
- функціональне призначення.

По конструктивному виконанню розрізняють дві системи штучного освітлення:

- загальне (рівномірне або локалізоване);
- комбіноване, коли до загального освітлення додається місцеве, що концентрує світловий потік безпосередньо на робочому місці.

Загальне рівномірне освітлення забезпечує рівномірний розподіл світлового потоку без врахування розташування устаткування, а загальне локалізоване (нерівномірне) – з урахуванням розташування робочих місць (наприклад, у складальних цехах при технічній неможливості устаткування місцевого освітлення).

За функціональним призначенням штучне освітлення ділиться на робоче, аварійне, охоронне і чергове.

Робоче освітлення призначене для забезпечення нормальних зорових умов при виконанні робіт, проходу людей і руху транспорту. Воно забезпечує нормовані характеристики (освітленість, яскравість освітлення) у приміщеннях і місцях провадження робіт поза будинками. Для приміщень, що мають зони з різними умовами природного освітлення і зони різних режимів роботи влаштовується роздільне керування освітленням таких зон.

Аварійне освітлення поділяється на освітлення безпеки, застосовуване для продовження роботи, і евакуаційне – для евакуації з приміщення при аварійному відключенні робочого освітлення. Світильники систем аварійного освітлення підключаються до мережі, незалежно від робочого освітлення, починаючи від щита підстанції.

Чергове освітлення призначене для приміщень, а охоронне — для освітлення площадок підприємства, що охороняються в неробочий час, який збігається з темним часом доби. Для чергового освітлення виділяють частину світильників робочого чи аварійного освітлення, які забезпечують мінімальну освітленість для несення чергувань і охорони.

6.3 Розрахунок системи штучного освітлення приміщення для роботи за комп’ютером

Приміщення включає в себе лише одну кімнату і має наступний вигляд (див. рис. 5.1).

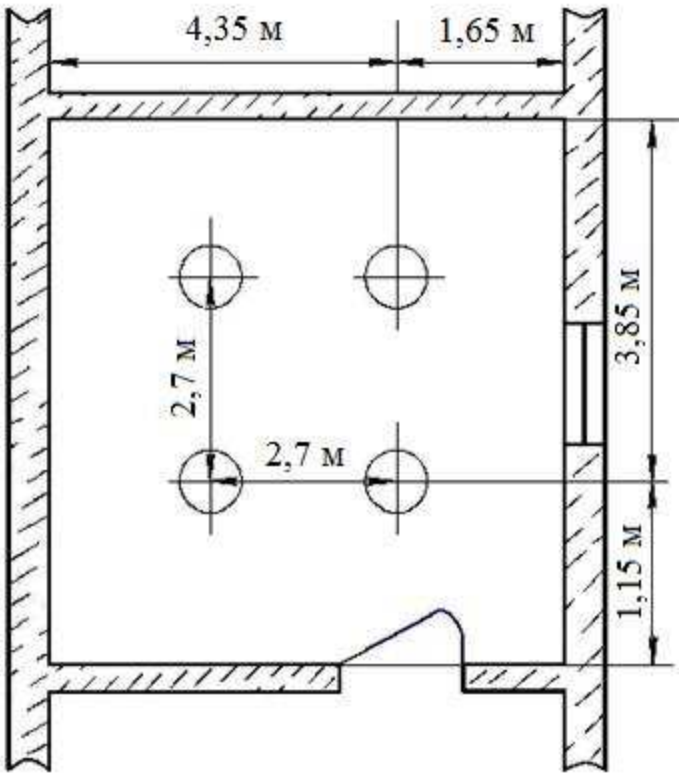

Рисунок 5.1 – Розташування світильників

Тож розрахунок буде проводитись лише для неї. Оскільки конфігурація приміщення відділу має просту форму, то розрахунок спрощується тільки до простого розрахунку.

Для виконання розрахунку використовуємо навчальний посібник [27].

Розрахунок освітлення для приміщення. Дано:

- довжина A = 6 м;
- ширина B = 5 м;
- висота H = 3 м;
- висота робочої поверхні h_p = 0.75 м;
- для освітлення приймаємо світильники типу ПВЛМ-Р;
- мінімальна освітленість люмінесцентної лампи за нормами E_{min} = 300 лк;
- коефіцієнт відображення стелі $\rho_{стелі}$ = 70%, стін $\rho_{стін}$ = 50%, робочої поверхні $\rho_{пов}$ = 30%;
- напруга мережі 220В.

Рішення:

Відстань від стелі до робочої поверхні:

$$H_0 = H - h_p = 3 - 0{,}75 = 2{,}25 \text{ м.}$$

Відстань від стелі до світильника:

$$h_c = 0{,}2 \cdot H_0 = 0{,}2 \cdot 2{,}25 = 0{,}45 \text{ м.}$$

Висота підвішування світильника над освітлюваною поверхнею:

$$h = H_0 - h_c = 2{,}25 - 0{,}45 = 1{,}8 \text{ м.}$$

Висота підвішування світильника над підлогою:

$$H_п = h + h_p = 1{,}8 + 0{,}75 = 2{,}55 \text{ м.}$$

Для досягнення найбільшої рівномірності освітлення приймаємо відношення L / h = 1,5.

Тоді відстань між центром світильників:

$$L = 1{,}5 \cdot h = 1{,}5 \cdot 1{,}8 = 2{,}7 \text{ м.}$$

Необхідна кількість світильників:

$$N = S / L^2 = A \cdot B / L^2 = 6 \cdot 5 / 2{,}7^2 = 4{,}1 \approx 4 \text{ світ,}$$

де S = A · B – площа приміщення.

Приймаємо 4 світильника для кімнати, в 2 ряди по 2 шт.

$$L_1 = (A – L) / 2 = (6 – 2,7) / 2 = 1,65 \text{ м.}$$

$$L_2 = (B – L) / 2 = (5 – 2,7) / 2 = 1,15 \text{ м.}$$

Індекс приміщення:

$$i = (A \cdot B) / (h \cdot (A + B)) = (6 \cdot 5) / (1,8 \cdot (6 + 5)) = 30 / 19,8 = 1,52.$$

За таблицею «Коефіцієнт використання світлового потоку» визначаємо, що для світильників типу ПВЛМ-Р та коефіцієнтів відображення стелі $\rho_{\text{стелі}}$ = 70%, стін $\rho_{\text{стін}}$ = 50%, робочої поверхні $\rho_{\text{пов}}$ = 30% та при i = 1,52 коефіцієнт використання світового потоку η = 0,6.

Світловий потік однієї лампи:

$$\Phi_p = (E_{min} \cdot S \cdot Kз \cdot Z) / (N \cdot \eta \cdot n) = (300 \cdot 6 \cdot 5 \cdot 1,5 \cdot 1,1) / (4 \cdot 0,6 \cdot 2)$$

$$\Phi_p = 3093,75 \text{ лм.}$$

де $K_З$ = 1,5 – коефіцієнт запасу, що враховує експлуатаційне зниження освітленості, в порівнянні з запроектованою, внаслідок забруднення підволоки, переборок, світильників і ламп, а також зменшення світлового потоку ламп у процесі їхньої експлуатації;

Z – коефіцієнт мінімальної освітленості, що враховує нерівномірність освітленості. Для люмінесцентних ламп значення цього коефіцієнту приймають рівним 1,1;

n – кількість ламп у світильніках; n = 2.

За знайденим світловим потоком з таблиці «Параметри лампи розжарювання загального призначення» обираємо лампу типу ЛБ потужністю 40 Вт, що має світловий потік Ф = 3040 лм, найбільш близький до розрахункового.

При цьому фактична освітленість:

$$E_ф = E_{min} \cdot (\Phi_л / \Phi_p) = 300 \cdot (3040 / 3093,75) = 294,79 \text{ лк.}$$

Загальна потужність освітлювальної установки:

$$P_{заг} = P_л \cdot N \cdot n = 40 \cdot 4 \cdot 2 = 320 \text{ Вт} = 0,32 \text{ кВт.}$$

6.4 Заходи щодо запобігання шкідливих факторів при роботі з персональним комп’ютером

Усе комп’ютерне устаткування повинне відповідати міжнародним стандартам якості і безпеки.

Згідно з результатами досліджень англійської фірми RELs TEMPEST, екрани дисплеїв інтенсивно випромінюють, починаючи з частоти 10 кГц, найбільш потужне випромінювання відповідає частоті термінової розгортки – 15 кГц. Для того, щоб уникнути цього, необхідно придбати дисплеї із металевим корпусом, що зменшує потужність випромінювань на 12%.

Дисплеї на рідких кристалах (LSD – Liquid Crstal Display), що випускаються нідерландською фірмою Philips, абсолютно позбавлені будь-якого випромінювання, що дозволяє зняти питання про шкідливість низькочастотного випромінювання екрана.

З метою зниження ризику для здоров’я різними організаціями були розроблені рекомендації з параметрів моніторів, випливаючи яким виробники моніторів борють за наше здоров’я. Усі стандарти безпеки для моніторів регламентують максимально припустимі значення електричних і магнітних полів, створюваних монітором при роботі. Практично в кожній розвитій крайні є власні стандарти, але особливу популярність в усьому світі (так склалося історично) завоювали стандарти, розроблені у Швеції і відомим під ім’ям TCO. TCO (The Swedish Confederation of Professional Employees, Шведська Конфедерація Професійних Колективів Робітників), членами якої є 1,3 мільйона шведських професіоналів, організаційно складається з 19 об’єднань, що працюють разом з метою поліпшення розумів роботи своїх членів. Ці 1,3 млн. членів представляють широкий спектр робітників та службовців з державного і приватного сектора економіки.

TCO ніяк не зв’язана з чи то політикою, чи то релігією, що є однією з визначальних причин, що дозволяє поєднуватися різними колективним членам під дахом однієї організації.

Справа у тім, що більш 80% службовців і робітників у Швеції мають справу з комп'ютерами, тому головна задача TCO – це розробити стандарти безпеки при роботі з комп'ютерами, тобто забезпечити своїм членам і всім іншим безпечне і комфортне робоче місце. Крім розробки стандартів безпеки, TCO бере доля в створенні спеціальних інструментів для тестування моніторів і комп'ютерів.

Стандарти TCO розроблені з метою гарантувати користувачам комп'ютерів безпечну роботу. Цим стандартом повинний відповідати кожен монітор, проданий у Швеції й у Європі. Рекомендації TCO використовуються виробниками моніторів для створення більш якісних продуктів, що менш небезпечні для здоров'я користувачів. Суть рекомендацій TCO складається не тільки у визначенні припустимих значень різного типові випромінювань, але й визначенні мінімально прийнятих параметрів моніторів, наприклад, підтримуваних дозволів, інтенсивності світіння люмінофора, запас яскравості, енергоспоживання, гучність і т.д. Більш того, крім вимог, у документах TCO приводяться докладні методики тестування моніторів.

Рекомендації TCO застосовуються як у Швеції, так і у всіх Європейських країнах для визначення стандартних параметрів, яким повинні відповідати всі монітори. До складових розроблених TCO рекомендацій сьогодні входять три стандарти: TCO'92, TCO'95 і TCO'99.

TCO'99 пред'являє більш тверді вимоги, чим TCO'95, в наступних областях: ергономіка (фізична, візуальна і зручність використання), енергія, випромінювання (електричних і магнітних полів), навколишнє середовище й екологія, а також пожежна й електрична безпека. Стандарт TCO'99 поширюється на традиційні CRT-монітори, плоскопанельні монітори (Flat Panel Displays), портативні комп'ютери (Laptop і Notebook), системні блоки і клавіатури.

Екологічні вимоги містять у собі обмеження на присутність важких металів, бромінатів і хлоринатів, фреонів (CFC) і хлорованих речовин усередині матеріалів, більш зручні умови для роботи.

Уже зроблено клавіатури, які можуть згинатися, складатися, розділятися, скручуватися, що забезпечує більш природне положення рук та зап’ясть при друкуванні і значно знижує ймовірність професіональних захворювань, пов’язаних з монотонністю рухів.

Однак спеціалісти різних країн вважають, що остаточно вирішити цю проблему можна буде тільки при повній відмові від клавіатури і створенні комп’ютерів, які друкують і виконують команди, що подаються голосом.

Загальні рекомендації-характеристики при роботі на ПК:

1) Дотримання обмежень за медичними вказівками.

2) Уважне ставлення до характеристик дисплея.

3) Правильна організація робочого місця.

4) Раціональна організація робочого часу.

Часткові рекомендації при роботі на ПК:

- Необхідно дотримуватися часових обмежень щодо роботи з ПК для людей, які страждають захворюваннями опорно-рухового апарату, очей, шкіри, а також вагітних жінок.
- Надавати перевагу використанню дисплеїв з високою роздільною можливістю і раціональним розміром екрана (15 дюймів і більше). Не використовуйте CGA, EGA, HGA, MGA монітори.
- Краще вибирати відеоадаптери з високою роздільною можливістю і частотою оновлення екранного зображення (регенерація) не менше 72 Гц, а краще – 85 Гц і вище.
- Обов’язково ставити на незахищений дисплей екранні поляризаційні фільтри (сіточні, плівочні, пластмасові, скляні), які зменшують шкідливе електромагнітне та ультрафіолетове випромінювання, роблять мерехтіння менш помітним, захищають дисплей від осідання зарядженого пилу, знижують електростатичний заряд, затримують рентгенівське випромінювання.
- Сидіти не ближче 70 см від дисплея (приблизно на відстані витягнутої руки).

- Екран дисплея не повинен бути орієнтованим у бік джерела світла (вікон, настільних ламп та ін.).
- При розміщенні робочого місця поряд із вікном кут між екраном дисплея і площиною вікна повинен становити не менше 90 градусів (для виключення блимань), ближню частину вікна бажано зашторити.
- Не слід розміщувати дисплей безпосередньо під джерелом освітлення чи поряд із ним, уникнути виникнення блимань на екрані.
- Освітленість робочого місця не повинна перевищувати 2/3 нормальної освітленості приміщення.
- Стіна позаду дисплея повинна освітлюватися приблизно так, як і його екран.
- При розміщенні в кімнаті декількох комп’ютерів, відстань між ними повинна складати не менше 1,2 м (особливо між задніми і боковими стінками сусідніх ПК, бо через них відбувається найбільш сильне випромінювання від блоків розгортки зображення).
- Робоче місце повинно бути обладнано так, щоб виключити незручні пози і довгочасні статичні напруження тіла.
- Загальний час роботи з ПК не повинен перевищувати 50% усього робочого часу, тобто не більше 4 годин на день для дорослих (за кордоном для опереторів ПК передбачено скорочений робочий день – 6 годин) і 2 години для дітей.
- Не слід перевищувати темпи роботи порядку 10 тис. нажимів клавіш за годину (приблизно 1500 слів).
- При роботі з ПК необхідно робити 15-хвилинні перерви через кожних 2 години, а при інтенсивній роботі – через 1 годину і навіть півгодини.
- Під час роботи на ПК необхідно слідкувати, щоб лінія руки в області зап’ястя залишалась прямою, зап’ястя були розслаблені, а пальці трохи зігнуті; не напружуйте руки, зігніть їх у ліктях приблизно під прямим кутом; слідкуйте, щоб удари по клавішах були не занадто сильними.

- Підлокітники робочого місця повинні бути опорою для рук як при роботі з клавіатурою, так і користуванні мишею (при цьому клавіатура і миша повинні розміщуватися так, щоб до них не потрібно було тягнутися)
- При виникненні напруження або спазмів у м’язах слід припинити роботу і зробити декілька вправ для розслаблення.
- Відрегулюйте висоту стільця так, щоб стегна розміщувалися паралельно підлозі, а ноги – твердо стояли на ній.
- Під час роботи сидіть прямо чи нахиліть корпус вперед, намагаючись зберегти природній згин тіла в поясниці; не сутультесь і не зводьте плечі; щоб не втомлювалась шия, не нахиляйте голову; верхній край монітора при цьому повинен знаходитися на рівні очей.
- Регулярно робіть вправи для очей; „переключайте” зір на різні відстані; працюючи з текстом, встановіть крупний шрифт із розміром символів не менше 12 чи 14 пунктів.
- Поверхня стола, клавіатури і корпуси ПК повинні бути одного кольору.
- Якщо можливо, з дисплеєм необхідно працювати в режимі „темні символи на світлому фоні” (найшкідливіший режим – „оранжеві символи на темному фоні”).
- У приміщенні, де встановлено ПК, необхідно підтримувати відносну вологість повітря не нижче 40% з метою зменшення шкідливого впливу електростатичного поля.
- Рекомендується відсунути все обладнання ПК (і взагалі все, що включено в електричну розетку) на 60 – 90 см від робочого місця.
- Не рекомендується стояти перед пристроєм, що працює (наприклад, принтером при виводі документів на друк, особливо перед лазерним).

Електробезпека забезпечується відповідною конструкцією електроустаткування, застосуванням технічних засобів захисту, організаційними і технічними заходами.

Конструкція електроустаткування повинна відповідати умовам його експлуатації, забезпечувати захист персоналу від зіткнення зі струмоведучими чистинами й устаткуванням – від влучення усередину сторонніх предметів і води.

Найбільш розповсюдженими технічними засобами захисту є захисне заземлення і занулення.

Організаційні і технічні заходи щодо забезпечення електробезпечності полягають в основному у відповідному навчанні, інструктажі і допуску до роботи людей, що пройшли медичний огляд, виконанні ряду технічних заходів при проведенні робіт з електроустаткуванням, дотриманні особливих вимог при роботі з частинами, що находяться під напругою.

Для постійного магнітного поля допустимим рівнем на робочому місці є напружність, що не повинна перевищувати 8 кА/м.

Одним з найбільш ефективних і часто застосовуваних методів захисту від низькочастотних і радіо випромінювань є екранування. Для екранів використовуються, головним чином, матеріали з великою електричною провідністю.

Як засоби індивідуального захисту застосовуються спецодяг, виготовлений з металізованої тканини у виді комбінезонів, халатів, фартухів, курток з каптурами і вмонтованими в них захисними окулярами.

7 ОХОРОНА НАВКОЛИЩНЬОГО СЕРЕДОВИЩА

7.1 Вплив людини на навколишнє середовище

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

Відносини у галузі охорони навколишнього природного середовища в Україні регулюються Законом України № 1268-XII від 26.06.91р. «Про охорону навколишнього природного середовища» [28], а також розроблюваними відповідно до нього земельним, водним, лісовим законодавством, законодавством про надра, про охорону атмосферного повітря, про охорону і використання рослинного і тваринного світу та іншим спеціальним законодавством.

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

Накопичення промислових відходів сприяє підвищенню захворюваності людей і тварин, прискоренню корозії машин і металевого устаткування, зниженню врожайності сільськогосподарських культур і продуктивності тваринництва, прискореному і нераціональному

використовуванню природних ресурсів і енергії, погіршенню багатьох властивостей екологічних систем, загибелі деяких унікальних природних територіальних комплексів, зникненню окремих видів тварин і рослин.

Основну загрозу для навколишнього середовища в межах галузі розробки ПЗ становлять комплекси персональних комп’ютерів, а саме електромагнітні випромінювання від них.

Дослідження останніх років показали, що електромагнітні випромінювання вищезгаданих електронних пристроїв містять торсіонову компоненту, котра несе інформацію про процеси, що відбуваються в тому чи іншому електронному пристрої.

Торсіонові поля мають високу проникаючу здатність і їх неможливо заекранувати.

Останніми роками при проектуванні (конструюванні), виготовленні (будівництві) і експлуатації технічних систем управління в різних сферах діяльності надзвичайно широко застосовуються персональні комп'ютери і всілякі комп’ютерні програми.

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

7.2 Утилізація відходів комп’ютерної техніки

Використання комп'ютерів вимагає вирішення таких важливих питань, як утилізація відходів (мікросхеми з вмістом кольорових металів, плати, диски).

При утилізації старих комп'ютерів відбувається їх розділ на сім фракцій: метали, пластмаси, штекери, дроти, батареї, стекло. Жодна деталь не йде для повторного використання, оскільки не можна гарантувати їх надійність, але у формі вторинної сировини вони йдуть на виготовлення нових комп'ютерів або інших пристроїв.

Детально розглянемо декілька прикладів переробки відходів обчислювальної техніки.

Гадолінієво-галлеві грати (ГГГ) використовуються у виробництві компонентів пристроїв, що запам'ятовують. В ході обробки біля 80% вихідного матеріалу перетворюється на відходи або відбраковується. ГГГ мають високу вартість і їх виділення з відходів представляє інтерес з економічної точки зору.

При здобутті досить чистих продуктів можливі повторне їх використання як вихідний матеріал. При цьому значно підвищується економічність виробництва заготівок з ГГГ. Під терміном відходи маються на увазі кристалічні залишки (залишки середовища для зростання кристалів, частини кристалів, виробництва, що утворюються на різних стадіях), а також дрібний порошок, що виходить при різанні, шліфовці, поліровці кристалів граната або подібних матеріалів.

Переробка цих відходів протягом ряду останніх років викликає труднощі і не вирішена до цих пір. Всі попередні спроби були безуспішними із-за низької розчинності цих складних оксидів.

Вдосконалений процес, розроблений Е. Гуссетом (патент США 4-198231 від 15 квітня 1980 року, фірма «Свісс Алюмініум Лтд.», Швейцарія), призначений для виділення галію і гадолінія з відходів, що містять обидва ці

елементи у вигляді оксидів або з'єднань, що перекладаються в оксиди. Відходи дрібно подрібнюються і потім розчиняються в сильних мінеральних кислотах. Гадоліній осідає з очищених розчинів у вигляді оксалата, галій виділяється в металевому вигляді електролітично. Електролітичне виділення галію може проводитися до виділення гадолінія у вигляді оксалата з розчину.

Розглянемо приклад проведення такого процесу. Відходи піддаються переробці, є залишки завантаження в пристрої для зростання кристалів, розколені частини кристалів зі всіх стадій переробки, дрібнозернисті порошки і пудри після операції різання, шліфовки і поліровки гранатів GdGaO. Змельчений порошок після обробки кристалів ГГГ висушується при 1200^0C і потім нагрівається при 6000^0C протягом декількох годин для розкладання летких забруднень.

Дрібнозернисті відходи в кількості 1000 г розміром менше 40 мкм., що містять 34% галію і 46% гадолінія кип'ятяться із зворотним холодильником протягом двох годин в 2100 мл 35%-ной соляної кислоти. Це відповідає 99% завантаження. Після кип'ячення частина відходів, що не розчинилася, фільтрується і промивається. Після висушування залишок важить 20 г. Залишки такого типа об'єднуються і піддаються повторній обробці кип'ятінням. Фільтрат, що включає промивні води, об'ємом 2300 мл, обробляється 4000 г металевого галію при 500^0C протягом 45 хвилин. Метал має максимально диспергувати.

В ході цієї операції благородніші елементи, присутні в розчині, виділяються і частково розчиняються в галії до його насичення і далі охолоджуються у вигляді інтерметалевих включень або в елементарній формі. Висадження можна проводити з меншою кількістю галію. Метал може періодично замінюватися на нову порцію до досягнення необхідної міри очищення. В результаті процесу очищення виходить розчин з вмістом галію 140 г/л і гадолінія 190 г/л.

Встановлюється величина *ph*=1 шляхом додавання 900 мл 4% розчину перекису водню для окислення домішок заліза. Осадження гадолінія

проводиться при 500^0C шляхом додавання 1500 г кристалічної технічної щавлевої кислоти COH*2HO; суспензія акуратно перемішується 12 годин для повторного осадження у вигляді оксалата гадолінія.

Оксалат гадолінія Gd(CO)*10HO відділяється центрифугуванням, промивається 20 мл розбавленої щавлевої кислоти (6 г/л) і висушується при 1300^0C; перетворення на оксид гадолінія досягається прожаренням при 8000^0C. Подальше очищення може проводитися розчиненням в кислоті і осадженням у вигляді окасалата гадолінія. До рідини після центріфугованія і промивань осаду (3300 мл) додають 2300 г KOH при інтенсивному перемішуванні до pH=12,6000 мл розчину з вмістом галію 55 г/л і гадолінія 1 г/л піддається електролізу при 600 Із з використанням катода з нержавіючої сталі і графітового анода при щільності струму близько 0,1 А/кв.см. Після 48 годин осідає 325 г галію і залишається розчин з вмістом 0,4 г/л, що не піддається подальшій переробці. Обложений метал має чистоту 99,99% і може бути безпосередньо перетворений в оксид.

При експлуатації персонального комп'ютера витрачаються наступні ресурси:

- електроенергія;
- папір для принтера;
- картріджи з фарбувальним порошком.

Для раціонального використання електроенергії не слід залишати включеним комп'ютер і принтер, якщо вони не потрібні в даний час. Друкувати можна з двох сторін. Витрати на папір навряд чи вдасться скоротити удвічі, проте економія буде вельми істотною. Проблему з утилізацією паперових відходів може вирішити вторинна переробка.

Сучасна технологія виготовлення елементів засобів обчислювальної техніки дозволяє досягти дуже низького рівня відмов елементів під час експлуатації. У зв'язку з цим відпадає необхідність проведення ремонтних робіт на місці експлуатації сучасних засобів обчислювальної техніки і як наслідок не утворюються відходи (несправні мікросхеми), що містять

дорогоцінні і рідкоземельні метали. Природно, в сервісних центрах, що спеціалізуються на ремонті і технічному обслуговуванні комп’ютерів, має бути організований збір і облік матеріалів, що містять коштовні метали, з подальшою обробкою цих матеріалів на спеціалізованих заводах з метою їх вилучення. У зв'язку з тим, що вітчизняне виробництво сучасних компонентів інформаційних технологій знаходиться в сьогоднішні дні лише в зачатковому стані, комп’ютери складаються з парку імпортних машин і устаткування. Із-за відсутності інформації про вміст дорогоцінних металів в елементах устаткування, строгий облік не представляється можливим і має бути покладений на фахівців експортних фірм. В умовах ринкової економіки підприємства мають бути самі зацікавлені у вторинній переробці, що містять дорогоцінні метали вузлів і елементів за умови неможливості їх використання.

Технологічний процес витягання дорогоцінних металів з плат здійснюється за наступною схемою. Плати сортуються по переважанню в них кількості дорогоцінних металів, дробляться і подрібнюються, обпалюються і плавляться. В процесі випалення піролітичному розкладанню піддається пластмасова основа, а основа дорогоцінних металів у вигляді металевих залишків відновлюється до оксидів. Металевий залишок подрібнюється, гранулюється, проходить магнітну сепарацію і відбувається відділення магнітних від немагнітних часток. Отриманий таким чином порошок, розділений по видах дорогоцінних металів, у вигляді гранул розплавляється в індукційних плавильних печах з подальшим розділенням кожного металу окремо.

ВИСНОВКИ

Мета, яка була поставлена в кваліфікаційній роботі, а саме: підвищення достовірності оцінювання тривалості виконання задач з розробки веб-застосунків, реалізованих мовою Java, досягнута в повному обсязі. Усі завдання, поставлені при підготовці кваліфікаційної роботи, були успішно виконані.

Були проаналізовані існуючі алгоритми та методи оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java. Сформульована постановка задачі.

Удосконалено однофакторну регресійну модель для оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java, за рахунок використання нормалізуючого перетворення сім'ї S_B Джонсона, що дозволило підвищити достовірність оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java, в порівнянні з існуючими моделями COCOMO та ISBSG.

Розроблена програма для оцінювання тривалості виконання задач з розробки веб-застосунків, реалізованих мовою Java, яка показала повну працездатність та ефективність, дозволить автоматизувати та скоротити час відповідних розрахунків.

Виконані спецрозділи з охорони труда, охорони навколишнього середовища та економічний розділи.

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. Software Test Estimation – 9 General Tips on How to Estimate Testing Time Accurately [Електронний ресурс] / Режим доступу: \www/ URL: http://www.softwaretestinghelp.com/software-test-estimation-how-to-estimate-testing-time-accurately/ – 25.09.2020 р. – Загол. з екрану.

3. Стадии цикла разработки ПО [Електронний ресурс] / Режим доступу: \www/ URL: http://qalight.com.ua/baza-znaniy/stadii-tsikla-razrabotki-po/ – 25.09.2020 р. – Загол. з екрану.

4. Сокращение затрат на обеспечение качества программных продуктов. Решение IBM Rational Quality Management [Електронний ресурс] / Режим доступу: \www/ URL: https://www-01.ibm.com/software/ru/smb/rational/quality/pdf/reducequalitycost_rrc_ru2.pdf – 25.09.2020 р. – Загол. з екрану.

5. App Download and Usage Statistics (2018) [Електронний ресурс] / Режим доступу: \www/ URL: http://www.businessofapps.com/data/app-statistics/ – 25.09.2020 р. – Загол. з екрану.

6. Макарова, Л.М. Удосконалення математичної моделі для оцінювання тривалості виконання робіт з розробки програмного забезпечення [Текст] / Л.М. Макарова, М.Ю. Пічугін // VI Міжнародна науково-технічна конференція «Комп'ютерне моделювання та оптимізація складних систем», Дніпро, ДВНЗ УДХТУ, 4-6 листопада 2020, с.51-52.

7. Веб-приложения – дань моде или будущее ПО? [Електронний ресурс] / Режим доступу: \www/ URL: http://article.techlabs.com.ua/50_17239.html – 05.10.2020 р. – Загол. з екрану.

8. Беллиньясо, М. Разработка Web-приложений в среде ASP.NET [Текст] / М. Беллиньясо. – М.: «Диалектика», 2007. – 640 с.

9. Макарова, Л.М. Моделі та інформаційна технологія переробки інформації для прогнозування відмов в обслуговуванні пристроїв термінальної мережі: дис. … канд. техн. наук: 05.13.06 / Нац. ун-т кораблебудування ім. адмірала Макарова. Миколаїв, 2015. 178 с.

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

11. 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.

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

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

14. Приходько, С.Б. Інтервальне оцінювання статистичних моментів часу затримок виконання програмних проектів на основі перетворення Джонсона [Текст] / С.Б. Приходько, А.В. Пухалевич // Збірник наукових праць НУК. – Миколаїв: НУК, 2010. – № 2 (431). – С.118-124. – ISSN 2313-0415.

15. Приходько, С.Б. Розробка нелінійної регресійної моделі трудомісткості програмних проектів на основі нормалізуючого перетворення Джонсона [Текст] / С.Б. Приходько, А.В. Пухалевич // Радіоелектронні і комп’ютерні системи. – Харків: ХАІ, 2012. – № 4 (56) – С.90-93.

16. Приходько, С.Б. Розробка нелінійних регресійних моделей трудомісткості програмних проектів на основі перетворення Джонсона

[Текст] / С.Б. Приходько, А.В. Пухалевич // Збірник наукових праць НУК. – Миколаїв: НУК, 2014. – № 2 (2014). – С.76-80.

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

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

19. Уніфікована мова моделювання [Електронний ресурс] / Режим доступу: \www/ URL: https://bibliofond.ru/view.aspx?id=446563#1 – 18.10.2020 р – Загол. з екрану.

20. Діаграма варіантів використання [Електронний ресурс] / Режим доступу: \www/ URL: http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?id=28086 – 18.10.2020 р – Загол. з екрану.

21. Діаграма станів [Електронний ресурс] / Режим доступу: \www/ URL: http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?id=2808710 – 18.10.2020 р – Загол. з екрану.

22. Інтерфейс [Електронний ресурс] / Режим доступу: \www/ URL: http://sven.ua/ua/service/glossary/ukr/11/ – 18.10.2020 р – Загол. з екрану.

23. Діаграма класів [Електронний ресурс] / Режим доступу: \www/ URL: http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?id=28088 – 18.10.2020 р – Загол. з екрану.

24. Діаграма послідовності [Електронний ресурс] / Режим доступу: \www/ URL: http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?r=16538 – 18.10.2020 р – Загол. з екрану.

25. Страуструп, Б. Программирование: принципы и практика с использованием C++, 2-е изд.: Пер. с англ. [Текст] / Б. Страуструп. – М.: ООО «И.Д. Вильямс», 2016. – 1328 с.

26. Прата, С. Язык программирования С++. Лекции и упражнения, 6-е изд., обновл. и расшир. [Текст] / С. Прата. – М.: Диалектика – Вильямс, 2016. – 1248 с.

27. Тубальцев, А.М. Виробниче освітлення та його розрахунок: Навчальний посібник [Текст] / А.М. Тубальцев – Миколаїв: УДМТУ, 2001. – 84 с. – ISBN 5–87848–113–8

28. Закон України № 1268-XII від 26.06.91р. «Про охорону навколишнього природного середовища» [Електронний ресурс] / Режим доступу: \www/ URL: https://zakon.rada.gov.ua/laws/show/1264-12#Text – 10.11.2020 р – Загол. з екрану.

Додаток А Технічне завдання

Вступ

Назва розроблюваного проекту: «Програма для оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java». Область застосування – підприємство, яке займається розробкою ПЗ.

1. Підстави для розробки

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

2. Призначення розробки

2.1 Експлуатаційне призначення

Експлуатаційним призначенням ПЗ є спрощення бізнес-процесів IT-компаній, покращення та легкість обробки масивів статистичних даних, полегшення оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java, та підвищення достовірності оцінювання.

2.2 Функціональне призначення програми

Функціональним призначенням ПЗ є автоматизація процесів:

– вводу статистичних даних;

– нормалізації вихідних вибірок за допомогою одновимірного перетворення Джонсона;

– розрахунку параметрів для вихідних та нормалізованих вибірок;

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

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

– оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java.

3. Вимоги до програмного забезпечення

3.1 Вимоги до функціональних характеристик

3.3.1 Вимоги до складу виконуваних функцій

Програмне забезпечення повинно виконувати наступні функції:

1) Загальні функції:

– можливість завантажити файл з вихідними даними у форматах Excel (.xml, .xls, .xlt, , .xlsx, .xlsm, .xlsb, .xltm, .xltx, .xlam);

– можливість змінити файл з вихідними даними;

– зчитування даних з файлу;

– розрахунок параметрів для вихідних вибірок;

– розрахунок параметрів одновимірного перетворення Джонсона;

– нормалізація вихідних вибірок за допомогою одновимірного нормалізую чого перетворення Джонсона;

– розрахунок параметрів для нормалізованих вибірок;

– перевірка вихідних та нормалізованих вибірок на нормальність розподілу;

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

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

– оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java;

– експорт результатів у файл.

3.1.2 Вимоги до організації вхідних та вихідних даних

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

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

Вхідні дані:

- файл у форматах Excel (.xml, .xls, .xlt, , .xlsx, .xlsm, .xlsb, .xltm, .xltx, .xlam);
- параметри нормалізації Джонсона (γ, η, ϕ, λ) для першого наближення.

Вихідні дані:

1) Розраховані параметри для вихідної вибірки:
   - кількість значень вибірки;
   - вибіркове середнє;
   - дисперсія;
   - середньоквадратичне відхилення;
   - асиметрія;
   - квадрат асиметрії;
   - ексцес.
2) Розраховані параметри одновимірного перетворення Джонсона:
   - γ;
   - η;
   - ϕ;
   - λ.
3) Розраховані параметри для нормалізованої вибірки:
   - кількість значень вибірки;
   - вибіркове середнє;
   - дисперсія;
   - середньоквадратичне відхилення;
   - асиметрія;
   - квадрат асиметрії;
   - ексцес.
4) Параметри для перевірки вихідних та нормалізованих вибірок на нормальність розподілу:

– значення χ^2 розрахункове;

– значення χ^2 критичне.

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

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

7) Значення оцінки тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java.

3.2 Вимоги до надійності

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

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

3.3 Вимоги до умов експлуатації

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

- температура повітря від +10^{o}C до +30^{o}C;
- відносна вологість повітря не більше 80
- запиленість повітря не більше 0,75 мг/м^2.

3.4 Вимоги до складу та параметрів технічних засобів

Система користувача повинна задовольняти вказаним мінімальним вимогам для коректного запуску програми:

– CPU: Intel Pentium 4405Y (1,2 ГГц);

– RAM: 1 GB;

– 500 Mb вільного місця на диску.

3.5 Вимоги до інформаційної та програмної сумісності

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

3.6 Вимоги до маркування та пакування

Додаток для оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java, буде упаковано в електронний архів і мати найменування «free-soft- duration.rar».

3.7 Вимоги до транспортування та зберігання

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

4. Вимоги до програмної документації

Програмна документація повинна містити:

– технічне завдання.

– опис програми

– текст програми

– інструкція користувача

– програма і методика випробувань ПЗ

5. Техніко-економічні показники

– Не розраховувалися.

6. Стадії та етапи розробки

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

Таблиця А.1 – Стадії та етапи розробки проекту

Стадії розробки	Етапи робіт	Термін виконання робіт	
		Початок етапу	Кінець етапу
1	2	3	4
1. Технічне завдання	1.1Обгрунтування необхідності розробки програми	06.09.20	11.09.20
	1.2 Розробка технічного завдання	11.09.20	16.09.20
	1.3 Затвердження технічного завдання	16.09.20	28.09.20
2. Ескізний проект	2.1 Розробка ескізного проекту	28.09.20	08.10.20
	2.2 Затвердження ескізного проекту	08.10.20	15.10.20
3. Технічний проект	3.1 Розробка технічного проекту	15.10.20	21.10.20
	3.2 Затвердження технічного проекту	21.10.20	01.11.20
4. Робочий проект	4.1 Розробка робочого проекту	01.11.20	07.11.20
	4.2 Затвердження робочого проекту	07.11.20	15.11.20
	4.3 Розробка документації	15.11.20	20.11.20

7. Порядок контролю та прийому

Контроль за аналізом та проектуванням кожної окремої частини програмного забезпечення для автоматизації обробки інформації для оцінювання розміру веб-додатків, реалізованих мовою Java, відбувається на кожному етапі з урахуванням вимог, визначених у технічному завдані. Кожна стадія розробки повинна бути представлена в зазначені строки та узгоджена із замовником.

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

Додаток Б Текст програми

файл MainWindow.cpp

```
#include"StartForm.h"
using System;
using System::Collections::Generic;
using System::Linq;
using System::Threading::Tasks;
using System::Windows::Forms;
using System::Windows::Documents;


          static class DurationEstimation
          {
              // The main entry point for the application.
              [STAThread]
              static void Main()
              {
            // Enabling Windows XP visual effects before any controls are created
                  Application.EnableVisualStyles();
                  Application.SetCompatibleTextRenderingDefault(false);
                  Application.Run(new StartForm());
              }
          }

#pragma once

namespace JavaWebAppDuration {

       using namespace System;
       using namespace System::ComponentModel;
       using namespace System::Collections;
       using namespace System::Windows::Forms;
       using namespace System::Data;
       using namespace System::Drawing;

       /// <summary>
       /// Summary for StartForm
       /// </summary>

       public ref class StartForm : public System::Windows::Forms::Form
       {
       public:
              StartForm(void)
              {
                     InitializeComponent();
                     Text = "JavaWebAppDuration";

              }

       protected:
              ~StartForm()
              {
                     if (components)
                     {
                            delete components;
                     }
```

```
      }
private:  System::Windows::Forms::Button^   button1;
private:  System::Windows::Forms::Label^   label1;
private:  System::Windows::Forms::OpenFileDialog^   openFileDialog1;

private:  System::Windows::Forms::Label^   label2;

private:  System::Windows::Forms::Label^   label3;
private:  System::Windows::Forms::Button^   button3;
private:  System::Windows::Forms::Button^   button4;

private:  System::Windows::Forms::Label^   label5;
private:  System::Windows::Forms::Label^   label6;
private:  System::Windows::Forms::Label^   label7;
private:  System::Windows::Forms::Label^   label8;

private:  System::Windows::Forms::TextBox^   textBox2;
private:  System::Windows::Forms::TextBox^   textBox3;
private:  System::Windows::Forms::TextBox^   textBox4;
private:  System::Windows::Forms::Label^   label9;
private:  System::Windows::Forms::Label^   label10;
private:  System::Windows::Forms::TextBox^   textBox5;
private:  System::Windows::Forms::TextBox^   textBox6;
private:  System::Windows::Forms::TextBox^   textBox7;
private:  System::Windows::Forms::Label^   label11;
private:  System::Windows::Forms::Label^   label12;


private:  System::Windows::Forms::Label^   label13;
private:  System::Windows::Forms::Label^   label14;
private:  System::Windows::Forms::TextBox^   textBox9;
private:  System::Windows::Forms::TextBox^   textBox10;
private:  System::Windows::Forms::TextBox^   textBox1;
private:  System::Windows::Forms::TextBox^   textBox8;
private:  System::Windows::Forms::TextBox^   textBox11;
private:  System::Windows::Forms::TextBox^   textBox12;
private:  System::Windows::Forms::Button^   button2;
private:  System::Windows::Forms::TextBox^   textBox13;
private:  System::Windows::Forms::TextBox^   textBox14;

private:  System::Windows::Forms::Label^   label15;

private:  System::Windows::Forms::Label^   label16;
private:  System::Windows::Forms::Label^   label18;
private:  System::Windows::Forms::TextBox^   textBox15;
private:  System::Windows::Forms::TextBox^   textBox16;
private:  System::Windows::Forms::Label^   label4;
private:  System::Windows::Forms::Button^   button5;



private:  System::ComponentModel::IContainer^   components;




protected:

private:
      /// <summary>
      /// Required designer variable.
```

```
            /// </summary>


#pragma region Windows Form Designer generated code
            /// <summary>
            /// Required method for Designer support - do not modify
            /// the contents of this method with the code editor.
            /// </summary>
            void InitializeComponent(void)
            {
                  this->button1 = (gcnew System::Windows::Forms::Button());
                  this->label1 = (gcnew System::Windows::Forms::Label());
                  this->openFileDialog1 = (gcnew
System::Windows::Forms::OpenFileDialog());
                  this->label2 = (gcnew System::Windows::Forms::Label());
                  this->label3 = (gcnew System::Windows::Forms::Label());
                  this->button3 = (gcnew System::Windows::Forms::Button());
                  this->button4 = (gcnew System::Windows::Forms::Button());
                  this->label5 = (gcnew System::Windows::Forms::Label());
                  this->label6 = (gcnew System::Windows::Forms::Label());
                  this->label7 = (gcnew System::Windows::Forms::Label());
                  this->label8 = (gcnew System::Windows::Forms::Label());
                  this->textBox2 = (gcnew System::Windows::Forms::TextBox());
                  this->textBox3 = (gcnew System::Windows::Forms::TextBox());
                  this->textBox4 = (gcnew System::Windows::Forms::TextBox());
                  this->label9 = (gcnew System::Windows::Forms::Label());
                  this->label10 = (gcnew System::Windows::Forms::Label());
                  this->textBox5 = (gcnew System::Windows::Forms::TextBox());
                  this->textBox6 = (gcnew System::Windows::Forms::TextBox());
                  this->textBox7 = (gcnew System::Windows::Forms::TextBox());
                  this->label11 = (gcnew System::Windows::Forms::Label());
                  this->label12 = (gcnew System::Windows::Forms::Label());
                  this->label13 = (gcnew System::Windows::Forms::Label());
                  this->label14 = (gcnew System::Windows::Forms::Label());
                  this->textBox9 = (gcnew System::Windows::Forms::TextBox());
                  this->textBox10 = (gcnew System::Windows::Forms::TextBox());
                  this->textBox1 = (gcnew System::Windows::Forms::TextBox());
                  this->textBox8 = (gcnew System::Windows::Forms::TextBox());
                  this->textBox11 = (gcnew System::Windows::Forms::TextBox());
                  this->textBox12 = (gcnew System::Windows::Forms::TextBox());
                  this->button2 = (gcnew System::Windows::Forms::Button());
                  this->textBox13 = (gcnew System::Windows::Forms::TextBox());
                  this->textBox14 = (gcnew System::Windows::Forms::TextBox());
                  this->label15 = (gcnew System::Windows::Forms::Label());
                  this->label16 = (gcnew System::Windows::Forms::Label());
                  this->label18 = (gcnew System::Windows::Forms::Label());
                  this->textBox15 = (gcnew System::Windows::Forms::TextBox());
                  this->textBox16 = (gcnew System::Windows::Forms::TextBox());
                  this->label4 = (gcnew System::Windows::Forms::Label());
                  this->button5 = (gcnew System::Windows::Forms::Button());
                  this->SuspendLayout();
                  //
                  // button1
                  //
                  this->button1->Cursor = System::Windows::Forms::Cursors::Hand;
                  this->button1->FlatAppearance->BorderColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>
(128)),
                        static_cast<System::Int32>(static_cast<System::Byte>(128)),
static_cast<System::Int32>(static_cast<System::Byte>(255)));
                  this->button1->FlatAppearance->BorderSize = 2;
```

```
                this->button1->Font   =   (gcnew   System::Drawing::Font(L"Microsoft
Sans            Serif",            9.75F,            System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                        static_cast<System::Byte>(204)));
                this->button1->Location = System::Drawing::Point(277, 82);
                this->button1->Name = L"button1";
                this->button1->Size = System::Drawing::Size(201, 37);
                this->button1->TabIndex = 0;
                this->button1->Text = L"Вибрати файл";
                this->button1->UseVisualStyleBackColor = true;
                this->button1->Click     +=     gcnew     System::EventHandler(this,
&StartForm::button1_Click);
                //
                // label1
                //
                this->label1->Font   =   (gcnew   System::Drawing::Font(L"Microsoft
Sans            Serif",            14.25F,            System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                        static_cast<System::Byte>(204)));
                this->label1->ForeColor                                          =
System::Drawing::SystemColors::ControlText;
                this->label1->ImageAlign                                         =
System::Drawing::ContentAlignment::MiddleLeft;
                this->label1->Location = System::Drawing::Point(12, 17);
                this->label1->Name = L"label1";
                this->label1->Padding = System::Windows::Forms::Padding(5);
                this->label1->Size = System::Drawing::Size(466, 38);
                this->label1->TabIndex = 1;
                this->label1->Text = L"Оцінювання розміру";
                this->label1->TextAlign                                          =
System::Drawing::ContentAlignment::MiddleCenter;
                this->label1->Click     +=     gcnew     System::EventHandler(this,
&StartForm::label1_Click);
                //
                // openFileDialog1
                //
                this->openFileDialog1->FileName = L"openFileDialog1";
                // // label2
                //
                this->label2->AutoSize = true;
                this->label2->Font   =   (gcnew   System::Drawing::Font(L"Microsoft
Sans           Serif",           11.25F,           System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
                        static_cast<System::Byte>(204)));
                this->label2->Location = System::Drawing::Point(14, 82);
                this->label2->Name = L"label2";
                this->label2->Size = System::Drawing::Size(257, 18);
                this->label2->TabIndex = 3;
                this->label2->Text = L"Виберіть файл з вихідними даними:";
                this->label2->Click     +=     gcnew     System::EventHandler(this,
&StartForm::label2_Click);
                //
                // label3
                //
                this->label3->AutoSize = true;
                this->label3->Font   =   (gcnew   System::Drawing::Font(L"Microsoft
Sans           Serif",           8.25F,           System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
                        static_cast<System::Byte>(204)));
                this->label3->Location = System::Drawing::Point(14, 104);
                this->label3->Name = L"label3";
                this->label3->Size = System::Drawing::Size(234, 13);
```

```
                    this->label3->TabIndex = 6;
                    this->label3->Text = L"(.xml, .xls, .xlt, .xlsx, .xlsm, .xlsb,
.xltm, .xltx, .xlam)";
                    this->label3->Click += gcnew System::EventHandler(this,
&StartForm::label3_Click);
                    //
                    // button3
                    //
                    this->button3->Cursor = System::Windows::Forms::Cursors::Hand;
                    this->button3->FlatAppearance->BorderColor =
System::Drawing::Color::Gray;
                    this->button3->FlatAppearance->BorderSize = 5;
                    this->button3->FlatStyle =
System::Windows::Forms::FlatStyle::System;
                    this->button3->Font = (gcnew System::Drawing::Font(L"Microsoft
Sans Serif", 9.75F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                        static_cast<System::Byte>(204)));
                    this->button3->Location = System::Drawing::Point(277, 171);
                    this->button3->Name = L"button3";
                    this->button3->Size = System::Drawing::Size(201, 37);
                    this->button3->TabIndex = 7;
                    this->button3->Text = L"Розрахувати";
                    this->button3->UseVisualStyleBackColor = true;
                    this->button3->Click += gcnew System::EventHandler(this,
&StartForm::button3_Click);
                    //
                    // button4
                    //
                    this->button4->Cursor = System::Windows::Forms::Cursors::Hand;
                    this->button4->FlatAppearance->BorderColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>
(128)),
                        static_cast<System::Int32>(static_cast<System::Byte>(128)),
static_cast<System::Int32>(static_cast<System::Byte>(255)));
                    this->button4->FlatAppearance->BorderSize = 2;
                    this->button4->Font = (gcnew System::Drawing::Font(L"Microsoft
Sans Serif", 9.75F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                        static_cast<System::Byte>(204)));
                    this->button4->Location = System::Drawing::Point(17, 171);
                    this->button4->Name = L"button4";
                    this->button4->Size = System::Drawing::Size(221, 37);
                    this->button4->TabIndex = 8;
                    this->button4->Text = L"Очистити";
                    this->button4->UseVisualStyleBackColor = true;
                    this->button4->Click += gcnew System::EventHandler(this,
&StartForm::button4_Click_1);
        }
}



#include "Normalization.h"
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

        namespace JavaWebAppDuration
        {
```

```
    {  // Enabling Windows XP visual effects before any controls are created
          [STAThread]
       Application.EnableVisualStyles();
       Application.SetCompatibleTextRenderingDefault(false);

       // Create the main window and run it
       Application.Run(new MainWindow());
       return 0;
}

#pragma once
       #include "StatisticalDistribution.h"
       #include <boost/math/distributions/normal.hpp>

       namespace StatisticalDistributions {
         class Normalization : public Normalization <long double> {
         public:
           inline JohnsonSb(long double nu, long double gamma,
                        long double phi, long double lambda) {
             init(nu, gamma, phi, lambda);
           }
           void init(long double, long double, long double, long double);
           virtual long double pdf(long double value) const;
           virtual long double cdf(long double value) const;
           virtual long double Inverse(long double value) const;

           ACCESSOR(cdist, pcdist);
           ACCESSORC(nu, nu);
           ACCESSORC(phi, phi);
           ACCESSORC(gamma, gamma);
           ACCESSORC(lambda, lambda);
         private:
           long double nu, gamma, phi, lambda;
           boost::math::normal_distribution<long double> pcdist;
           mutable std::normal_distribution<long double> dist;
         };

       }
```

файл Normalization.h

```
#include <iostream>
#include <vector>
#include <map>
#include <algorithm>
#include <math.h>
#include <float.h>

class Normalization
{
static void  JohnsonTranslation(
       double *v,
       double matrix[3][3]
)
{
       double d;
       double dp;
       double c;
       double s;
       double x;
       double r;
       double w;
       int i;
```

```
        int j;
        bool skip;

        skip=false;
        w=1.0;
        for (i=0;i<2;i++) {
       if (!skip) {
                    if (0.0 equals (x=v[i])) continue;
                    d=matrix[i][i];
                    dp=d+w*x*x;
                    matrix[i][i]=dp;
                    c=d/dp;
                    s=w*x/dp;
                    if (d equals 0.0) skip=true;
                    else w*=c;
                    for (j=i+1;j<3;j++) {
                           matrix[i][j]=s*v[j]+c*(r=matrix[i][j]);
                       v[j]-=x*r;
                    }
       }
        }
}


bool JohnsonMomentSb(
        JohnsonParms& parms,
        double mean,
        double sd,
        double sqrtB1,
        double B2
)
{
        const double TOLSB=0.01;
        const double tTol=1e-2;
        const int iterLimit=50;
        const int N=6;

        bool howExit=true;
        double absSqrtB1=fabs(sqrtB1);
        double B1=sqrtB1*sqrtB1;
        bool negativeB1=(sqrtB1<0.0);

        double delta;
        double x=1.0+0.5*B1;
        double y=absSqrtB1*sqrt(1.0+B1/4.0);
        double w=pow(x+y,1.0/3.0)+pow(x-y,1.0/3.0)-1.0;
        double f=w*w*(3.0+w*(2.0+w))-3.0;
        double e=1.0+B1;
        e=(B2-e)/(f-e);
        if (absSqrtB1<=TOLSB) {
               f=2.0;
        }
        else {
               delta=1.0/sqrt(log(w));
               if (delta<0.64) {
                      f=1.25*delta;
               }
               else {
                      f=2.0-8.5245/(delta*(delta*(delta-2.163)+11.346));
               }
        }
        f=1.0+e*f;
```

```
	if (f<1.8) {
		delta=0.8*(f-1.0);
	}
	else {
		delta=(0.626*f-0.408)*pow((3.0-f),-0.479);
	}

	double gamma=0.0;
	if (B1>=tTol) {
		if (delta<=1.0) {
			gamma=(0.7466*pow(delta,1.7973)+0.5955)*pow(B1,0.485);
		}
		else {
			if (delta<=2.5) {
				gamma=pow(B1,0.0623*delta+0.4043);
			}
			else {
				gamma=pow(B1,0.0124*delta+0.5291);
			}
			gamma*=(0.9281+delta*(1.0614*delta-0.7077));
		}
	}

	int count=0;
	bool more=false;
	bool errorSet=false;
	double moments[N];
	double h2=0;
	double oldDeltaGamma=100.0;
	double oldDeltaDelta=100.0;
	repeat
		if (JohnsonMOM(gamma,delta,moments)) {
			h2=moments[1]-moments[0]*moments[0];
			if (h2>0.0) {

				double h2a=sqrt(h2)*h2;
				double h2b=h2*h2;
				double		h3=moments[2]-moments[0]*(3.0*moments[1]-
2.0*moments[0]*moments[0]);
				double rbet=h3/h2a;
				double h4=moments[3]-moments[0]*(4.0*moments[2]-moments[0]*
					(6.0*moments[1]-3.0*moments[0]*moments[0]));
				double bet2=h4/h2b;
				double w=gamma*delta;
				double u=delta*delta;

					// Get derivatives
				double dd[N];
				double deriv[N];

				for (int j=0;j<2;j++) {
					for (int k=0;k<4;k++) {
						double t=(double)k;
						double s;
						if (! j) {
							s=moments[k+1]-moments[k];
						}
						else {
							s=((w-t)*(moments[k]-
moments[k+1])+(1.0+t)*
								(moments[k+1]-moments[k+2]))/u;
						}
```

```
                                    dd[k]=t*s/delta;
                              }
                              double  t=2.0*moments[0]*dd[0];
                              double  s=moments[0]*dd[1];
                              double  y=dd[1]-t;
                              deriv[j]=(dd[2]-3.0*(s+moments[1]*dd[0]-
t*moments[0])-
                                                  1.5*h3*y/h2)/h2a;
                              deriv[j+2]=(dd[3]-
4.0*(dd[2]*moments[0]+dd[0]*moments[2])+
                                                  6.0*(moments[1]*t+moments[0]*(s-
t*moments[0]))-
                                                  2.0*h4*y/h2)/h2b;
                        }
                        double  t=1.0/(deriv[0]*deriv[3]-deriv[1]*deriv[2]);
                        double                deltaGamma=(deriv[3]*(rbet-absSqrtB1)-
deriv[1]*(bet2-B2))*t;
                        double        deltaDelta=(deriv[0]*(bet2-B2)-deriv[2]*(rbet-
absSqrtB1))*t;

                              // New estimates of gamma and delta
                        gamma-=deltaGamma;
                        if (B1 equals 0.0 || gamma<0.0) {
                              gamma=0.0;
                        }
                        delta-=deltaDelta;
                        deltaGamma=fabs(deltaGamma);
                        deltaDelta=fabs(deltaDelta);
                        more=(deltaGamma>tTol  ||  deltaDelta>tTol);
                              // error if iterates increase
                        errorSet=(deltaGamma>oldDeltaGamma                              ||
deltaDelta>oldDeltaDelta);
                              oldDeltaGamma=deltaGamma;
                              oldDeltaDelta=deltaDelta;
                        }
                  }
            until(! more  ||  errorSet  ||  count++>iterLimit);
            if (! errorSet && ! more) {
                  parms.delta=delta;
                  parms.lambda=sd/sqrt(h2);
                  if (negativeB1)     {
                        gamma=-gamma;
                        moments[0]=1.0-moments[0];
                  }
                  parms.gamma=gamma;
                  parms.xi=mean-parms.lambda*moments[0];
                  parms.type=SB;

                  goto  theExit;
            }

            if  (JohnsonMomentSb(parms,mean,sd,sqrtB1,B2))  {
                        return  parms;
                  }
                  else {
                        error("\nCouldn't  do  an  Sb  fit");
                        return  parms;
                  }
            }
      }
```

```
class LinearRegression: public Regression
	{
	public:
	    LinearRegression(const DataSet&);

	    virtual vec h_Theta(vec) const;
	    virtual double cost(mat&, const mat&) const;
	    virtual mat derivative(const mat&, const mat&) const;

	    vec predict(mat) const;
	    void create_model(const unsigned int) const;
	};


//chi square

static void  ChiSquare(
		double* a,
		double* b
	)
{
	    hi_square(a);
	    hi_square(b);
	    return 0;
}


void hi_square(std::vector<double> &a) {
	    std::vector<int> counter;
	    for (int i = 0; i < 10; ++i) {
	        counter.push_back(std::count_if(a.begin(), a.end(), [&i](double
val) {
	            return (val < i * 0.1 + 0.1 && val > i * 0.1);
	        }));
	    }
	       double hi = 0;

	    for (auto item : counter) {
	        hi += std::pow(item - arraySize * 0.1, 2) / (arraySize * 0.1);
	    }
	      return hi;
	}




#include<iostream>
#include<fstream>
#include<math.h>
class Regression
{
public:
    enum RegressionType{Regres, Classif};

    Regression(const DataSet&, const char*);
    ~Regression();

    double gradientdescent(mat, const mat, const double, const unsigned int);

    mat theta(void) const;
    void init_theta(void);
```

```
    void printTheta(void) const;

    string regressionType(void) const;
    void set_regressionType(const string&);

    double alpha(void) const;
    void set_alpha(const double);

    double lamda(void) const;
    void set_lamda(const double);

    virtual vec h_Theta(vec) const = 0;
    virtual double cost(mat&, const mat&) const = 0;
    virtual mat derivative(const mat&, const mat&) const = 0;

protected:
    mat d_Theta;

    const DataSet& d_dset;
    RegressionType d_reg_type;

    double d_alpha;
    double d_lamda;

    fstream d_lamdaCostGraph;
};
```

Додаток В Опис програми

ПЗ для оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java, яке розроблялось у рамках кваліфікаційної роботи, спрямовано на використання на підприємствах, які займаються розробкою ПЗ.

Дане ПЗ розроблене з метою спрощення бізнес-процесів ІТ-компаній, покращення процесу обробки масивів статистичних даних, полегшення оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java, та підвищення достовірності оцінювання.

Функціональним призначенням ПЗ є автоматизація процесів:

– вводу статистичних даних;

– нормалізації вихідних вибірок за допомогою одновимірного перетворення Джонсона;

– розрахунку параметрів для вихідних та нормалізованих вибірок;

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

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

– оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java.

Реалізація ПЗ була виконана високорівневою мовою програмування C++ у середовищі розробки Microsoft Visual Studio.

При розробці ПЗ значну увагу було приділено користувацькому інтерфейсу. Важливим завданням було створити його якомога простим у використанні і в той самий час багатофункціональним і зручним, адже ним користуватимуться дуже часто. Всі графічні елементи і кольорові гами було використано відповідно до загальноприйнятих, і тепер навіть тому, хто вперше користується ПЗ, буде інтуїтивно зрозуміло, як виконати бажану дію.

Було проведено повне функціональне тестування, а також навантажувальне тестування. Усі виявлені недоліки ПЗ були зафіксовані в протоколах і усунуті розробником до моменту впровадження ПЗ в дію. Наступні випробування були успішними та показали повну готовність ПЗ до використання.

Додаток Г Інструкція користувача

ПЗ автоматизованої обробки інформації для оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java, яке розроблялось у рамках кваліфікаційної роботи, запаковано в архів з назвою JavaWebAppDuration.zip.

Після розпакування архіву заходимо в папку та вікриваємо програму JavaWebAppDuration. Перед нами з’являється початкове вікно програми, зображене на рис. Г.1.

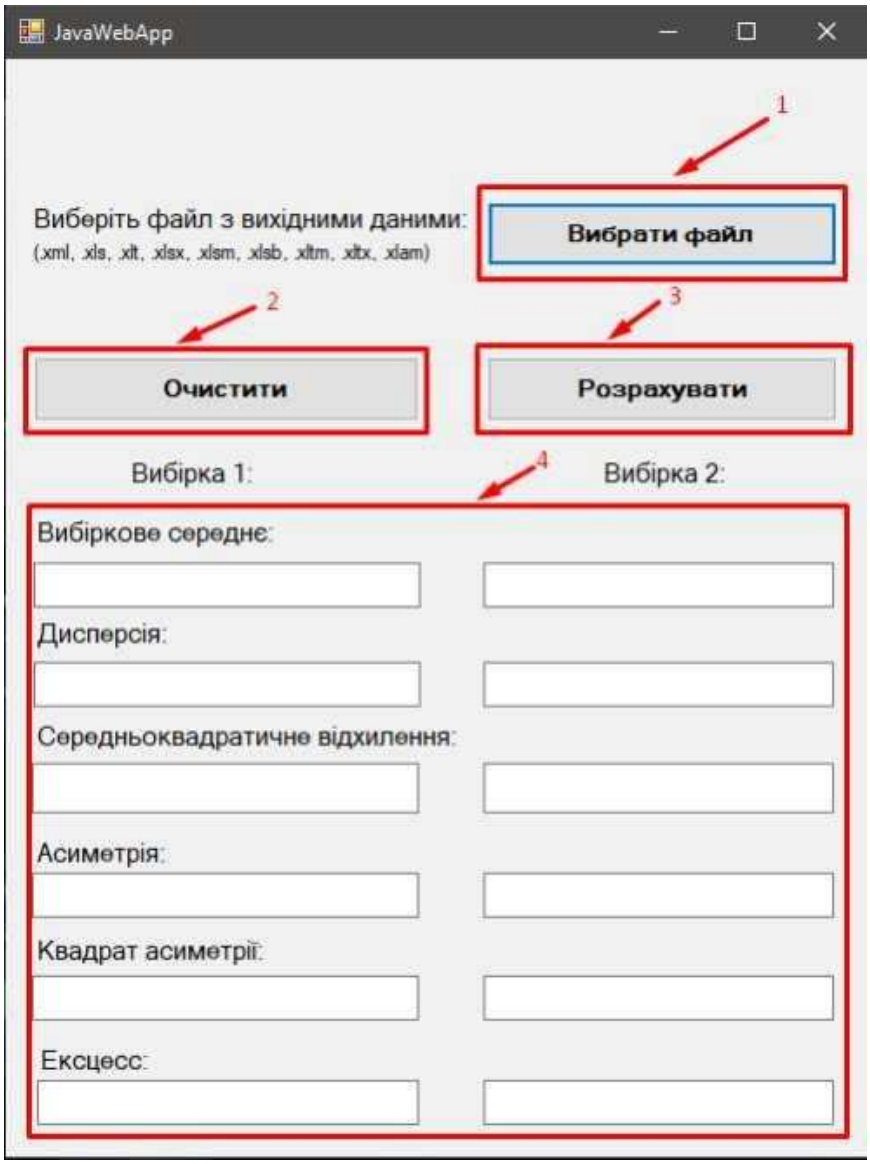

Рисунок Г.1 – Початкове вікно програми

Опис функцій та інформаційних блоків, наявних на початковому вікні програми:

1. Кнопка для завантаження файлу.

2. Кнопка для очищення полів та файлу.

3. Кпопка для розрахунку параметрів вихідних вибірок.

4. Блок параметрів вихідних вибірок.

Порядок дій для оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java:

1. Натискаємо кнопку «Вибрати файл» (1), відкривається вікно вибору файлу, зображене на рис. Г.2. Вибираємо файл в одному з вказаних форматів.

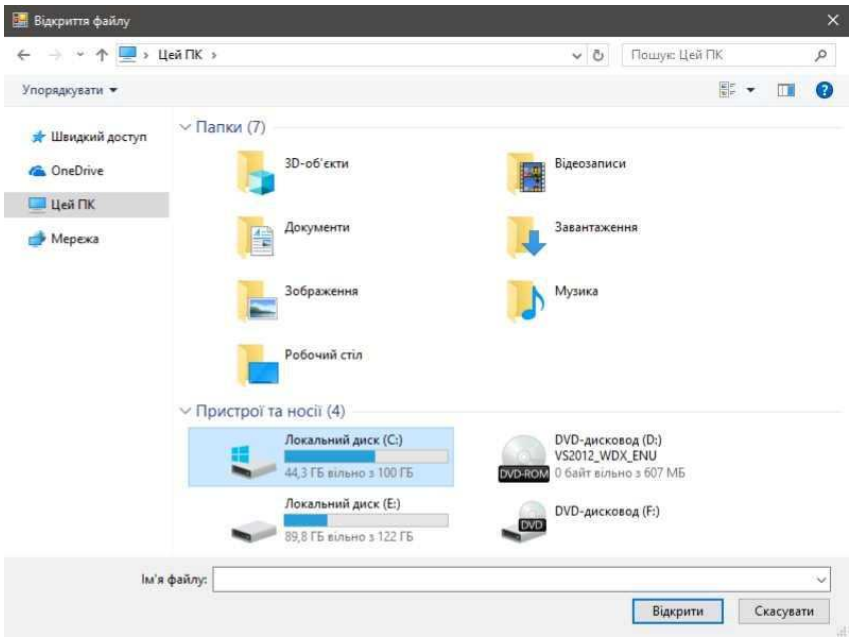

Рисунок Г.2 – Вибір файлу з вихідними параметрами

2. Натискаємо кнопку «Розрахувати» (3) та отримуємо розраховані параметри у блоці 4 (див. рис. Г.1).

3. Після розрахунку параметрів з’являється можливість перевірити нормальність вихідних вибірок (5) та перейти до нормалізації (6) (див. рис. Г.3).

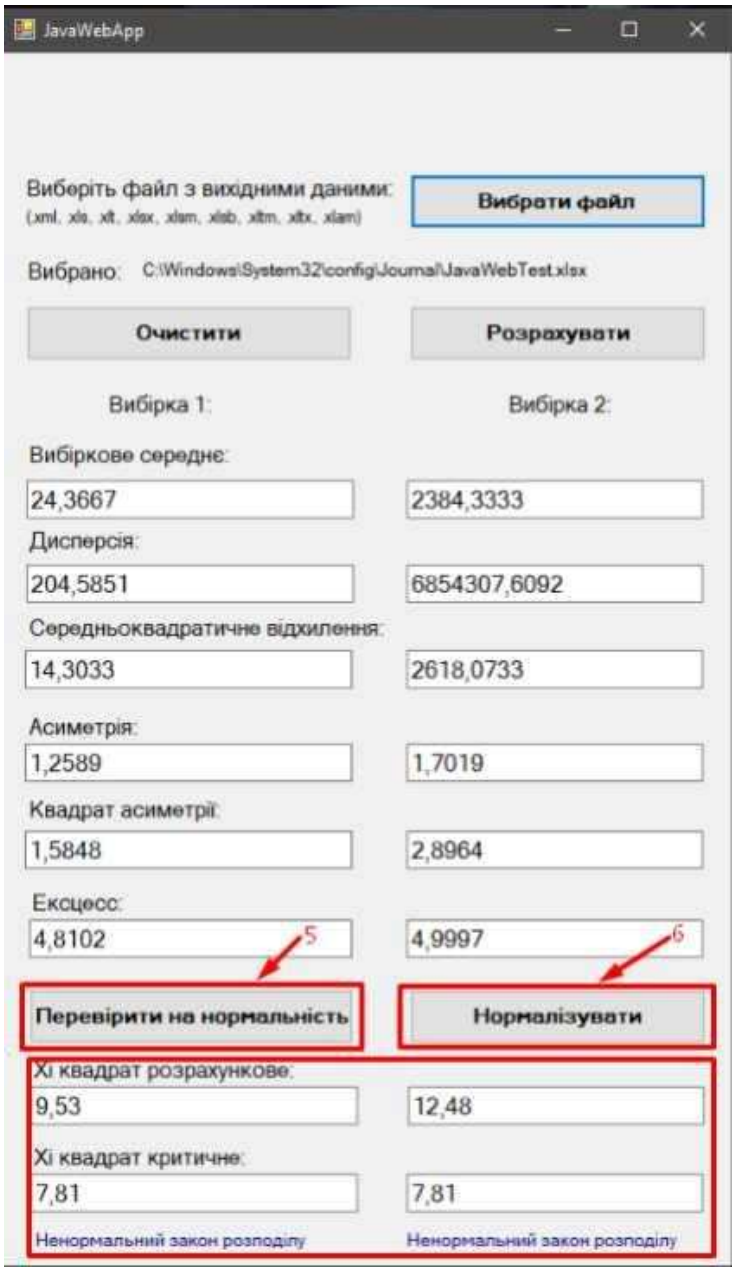

Рисунок Г.3 – Вікно перевірки нормальності розподілу вихідних вибірок

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

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

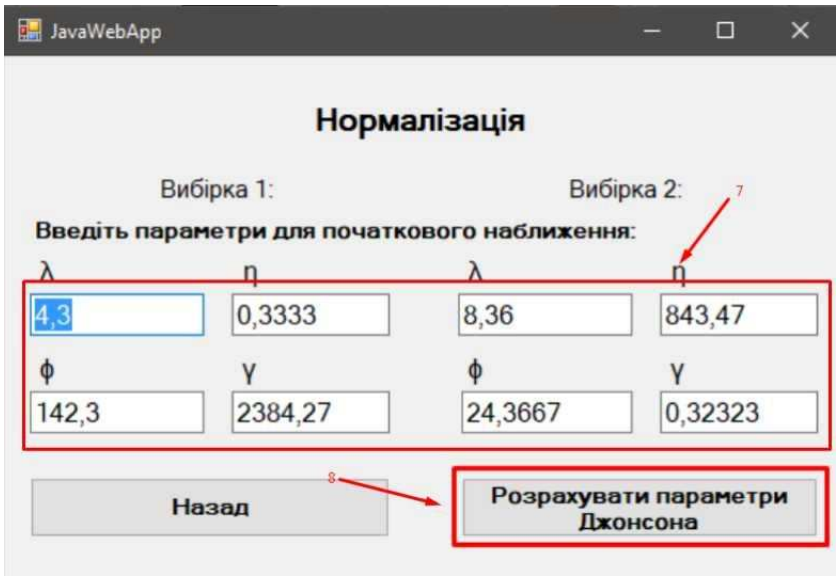

Рисунок Г.4 – Вікно вводу початкових параметрів Джонсона для нормалізації

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

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

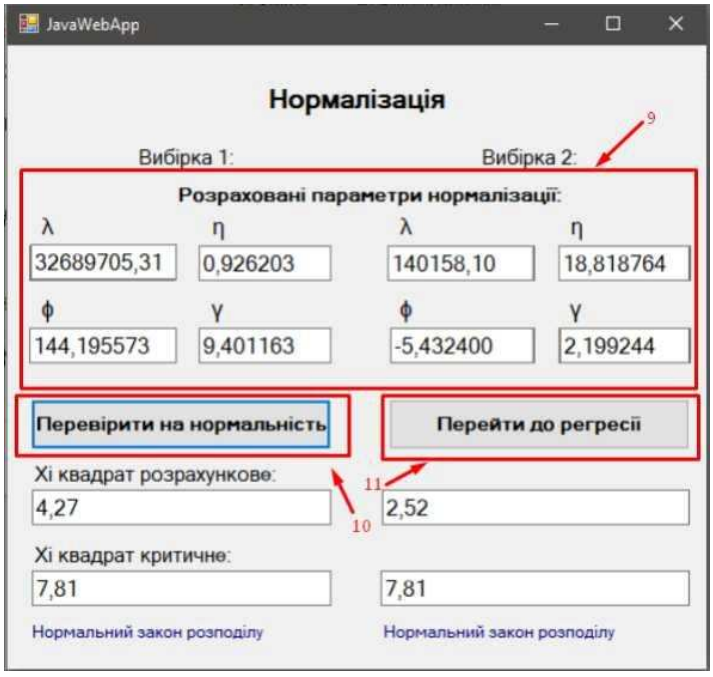

Рисунок Г.5 – Вікно розрахованих оптимальних параметрів Джонсона для нормалізації

8. Натискаємо кнопку «Перейти до регресії» (11). Виводиться нове вікно «Рівняння регресії» з двома вкладками для лінійного та нелінійного рівняння, яке зображене на рис. Г.6.

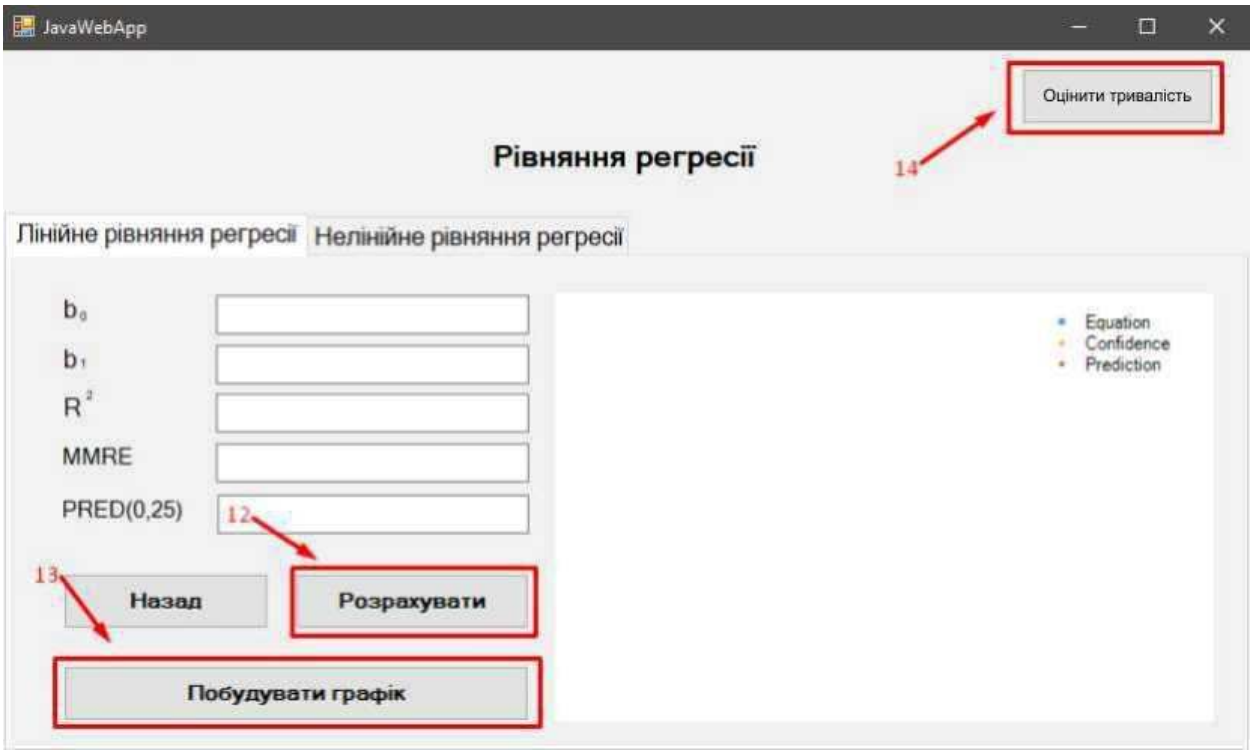

Рисунок Г.6 – Вікно лінійного рівняння регресії

9. Натискаємо кнопку «Розрахувати» (12) та отримуємо розраховані параметри для відповідного типу рівняння.

10. Натискаємо кнопку «Побудувати графік» (13) та отримуємо побудований графік для відповідного рівняння регресії, його довірчого інтервалу та інтервалу прогнозування в правій стороні вікна (див. рису. Г.6).

11. Натискаємо кнопку «Оцінити тривалість» (14). Виводиться нове вікно «Оцінювання тривалості», яке зображене на рис. Г.7.

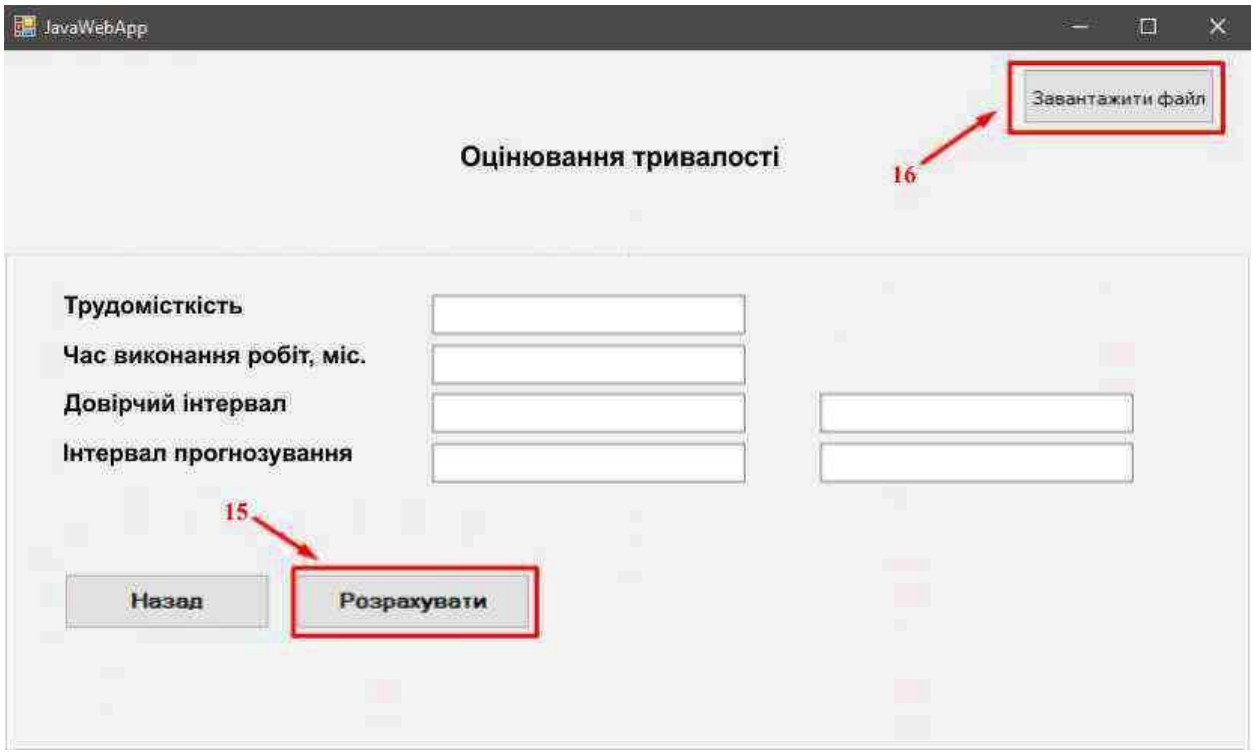

Рисунок Г.7 – Вікно оцінювання тривалості

12. Вводимо вручну значення трудомісткості розробки та натискаємо кнопку «Розрахувати» (15). Отримуємо значення часу виконання робіт та границі довірчого інтервалу та інтервалу прогнозування у відповідних полях.

13. Натиснувши кнопку «Завантажити файл» (16), отримуємо файл у форматі CSV з розрахованими параметрами. Цей універсальний формат можна відкрити як у текстовому редакторі, так і в таблицях Excel. Також цей файл може слугувати вхідним файлом у бідь-яку базу даних.

Додаток Д Програма і методика випробувань ПЗ

ПЗ для автоматизованої обробки інформації для оцінювання тривалості виконання робіт з розробки веб-застосунків, реалізованих мовою Java, яке розроблялось у рамках кваліфікаційної роботи, пройшло ряд випробувань. Усі виявлені недоліки ПЗ були зафіксовані й усунуті до моменту впровадження ПЗ в дію. Випробування ПЗ проводилося на цільовому обладнанні в тій конфігурації, яка заявлена в технічному завданні (див. Додаток А).

Нижче наведено список випробувань, які проводились для виявлення недоліків програми та описано послідовність дій для кожного з них, показано результати реагування ПЗ.

1. Випробування на завантаження некоректного формату файлу.

У головному вікні програми натиснули кнопку «Вибрати файл». Зліва було наведено допустимі параметри. Було вибрано картинку з недопустимим розширенням .jpg. У результаті отримали повідомлення системи про недопустимий формат файлу та можливість вибрати інший файл. Випробування пройдено успішно.

2. Випробування на завантаження коректного формату файлу.

У головному вікні програми натиснули кнопку «Вибрати файл». Зліва було наведено допустимі параметри. Було вибрано таблицю Excel з розширенням .xlsx. У результаті файл успішно завантажився та нижче було виведено його назву. Випробування пройдено успішно.

3. Випробування на розрахунок параметрів без завантаження файлу.

У головному вікні програми не вибрали жодного файлу. Натиснули кнопку «Розрахувати». У результаті отримали повідомлення системи про необхідність вибрати файл. Випробування пройдено успішно.

4. Випробування на розрахунок параметрів після коректного завантаження файлу.

Після коректного завантаження файлу натиснули «Розрахувати».У відповідних текстових блоках з`явились розраховані параметри, нижче – можливість перевірити нормальність розподілу та перехід до нормалізації. Випробування пройдено успішно.

5. Випробування на розрахунок параметрів Джонсона при некоректному вводі.

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

6. Випробування на розрахунок параметрів Джонсона при коректному вводі.

У вікні «Нормалізація» ввели коректні параметри для початкового наближення та натиснули кнопку «Розрахувати параметри Джонсона». У результаті з’явилось відповідне вікно, в якому виведено розраховані параметри нормалізації та можливість перевірити нормальність розподілу нормалізованих вибірок. Випробування пройдено успішно.

7. Випробування на побудову графіку без попереднього розрахунку параметрів лінійного рівняння.

У вікні «Рівняння регресії» натискаємо кнопку побудувати графік без попереднього розрахунку параметрів. У результаті нічого не відбувається, а курсор при наведенні на кнопку стає забороненим, а кнопка неклікабельною. Випробування пройдено успішно.

8. Випробування на побудову графіку після розрахунку параметрів рівняння.

У вікні «Рівняння регресії» натискаємо кнопку побудувати графік після розрахунку параметрів. У результаті графік успішно побудований. Випробування пройдено успішно.

Отже, усі наведені перевірки тестування показали очікуваний результат реагування системи на дії користувача. Можна зробити висновок, що ПЗ

працює правильно, не допускаючи некоректних дій з боку користувача та у разі виникнення проблем інформує користувача про можливі шляхи їх вирішення.