МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»
Завідувач кафедри
_____
«___» _____ 2021 р.

КВАЛІФІКАЦІЙНАРОБОТА

на здобуття ступеня вищої освіти «магістр»

на тему: «Нелінійна регресійна модель для оцінювання часу розробки Java-застосунків для платформи midrange та створення програми для її реалізації»

Виконав: студент 6151м групи
_____Фаріонова Т.А.
(підпис, ПІБ)

Керівник роботи:
зав. кафедрою ПЗАС, д.т.н., проф.
(посада, науковий ступень вчене звання)
_____Приходько С.Б.
(підпис, ПІБ)

Миколаїв – 2021 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____проф. С.Б.Приходько
(підпис)
«___» _____ 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «магістр»

Студенту Фаріоновій Тетяні Анатоліївні
(Прізвище, ім'я, по батькові)

1. Тема роботи: «Нелінійна регресійна модель для оцінювання часу розробки Java-застосунків для платформи midrange та створення програми для її реалізації»

Керівник роботи Приходько Сергій Борисович, д.т.н., проф. _____
Затверджені наказом ректора № 1239уч від «13» _____10_____ 2021 р.
2. Термін подання роботи: 06.12.2021 р.
3. Вихідні дані по роботі: _____
_____

4. Перелік питань, що належать до розробки (найменування розділів) _____
Вступ (Актуальність теми. Зв'язок роботи з науковими програмами, планами, темами. Мета і завдання дослідження. Об'єкт дослідження. Предмет дослідження. Методи дослідження. Наукова новизна одержаних результатів. Практичне значення одержаних результатів. Особистий внесок здобувача. Апробація результатів досліджень**.** Публікації.); Огляд літератури та обґрунтування необхідності проведення досліджень за обраною темою; Викладення результатів власних досліджень з висвітленням того нового, що пропонується; Проект 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 презентаційних матеріалів_____
_____
_____
_____
_____
_____
_____
_____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
5	Трушлякова А.Б.		
6	Гурець Н.В.		
7	Гурець Н.В.		

7. Дата видачі завдання 13.10.2021 р.

К А Л Е Н Д А Р Н И Й П Л А Н

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу Вступ	18.10.2021	НС*
2.	Підготовка розділу з огляду літератури та обґрунтування необхідності проведення досліджень за обраною темою	19.10.2021	НС*
3.	Підготовка розділу (ів) з результатів власних досліджень	22.10.2021	НС*
4.	Підготовка розділу з проекту програмного забезпечення	16.11.2021	
5.	Підготовка організаційно-економічного розділу	19.11.2021	
6.	Підготовка розділу з охорони праці	22.11.2021	
7.	Підготовка розділу з охорони навколишнього середовища	24.11.2021	
8.	Підготовка розділу Висновки	25.11.2021	
9.	Оформлення списку використаних джерел та додатків	29.11.2021	
10.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версії роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	06.12.2021	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку
11.	Підготовка презентації та доповіді	09.12.2021	
12.	Попередній захист роботи на засіданні кафедри ПЗАС	10.12.2021	
13.	Подання на кафедру ПЗАС електронних версії наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді	20.12.2021	

* - за результатами наукового стажування (НС), яке було з 01.09.2021 до 10.10.2021 р.

Студент _____ Фаріонова Т.А.
(підпис) (ПІБ)

Керівник роботи _____ Приходько С.Б.
(підпис) (ПІБ)

РЕФЕРАТ

Фаріонова Тетяна Анатоліївна

«Нелінійна регресійна модель для оцінювання часу розробки Java-застосунків для платформи midrange та створення програми для її реалізації»

Кваліфікаційна робота на здобуття ступеня вищої освіти магістр зі спеціальності 121 – «Інженерія програмного забезпечення» (ОП «Інженерія програмного забезпечення»). Національний університет кораблебудування імені адмірала Макарова. Миколаїв, 2021 р.

Обсяг роботи: 133 стор., 14 табл., 28 рис., 46 використаних джерел, 5 додатків.

Актуальність теми роботи: За статистикою біля 30% програмних проєктів мають труднощі з реалізацією (не вкладаються в бюджет, виходять з порушенням часових обмежень, або неналежної якості). Програмні проєкти для платформи midrange характеризуються своїм значним розміром, великою кількістю компонентів та спеціальними вимогами до надійності. За статистичними даними компанії ISBSG, частка ПЗ для платформи midrange, основною мовою розробки якого є Java, складає приблизно 25%. Тому достовірне оцінювання тривалості розробки Java-застосунків для цієї платформи є актуальною задачею та представляє науково-практичний інтерес.

Мета та завдання дослідження. Метою є підвищення достовірності оцінювання тривалості розробки Java-застосунків для платформи midrange за рахунок побудови нелінійної регресійної моделі. **Завдання дослідження**: 1. Провести аналіз існуючих методів та моделей оцінювання тривалості програмних проектів. 2. Побудувати нелінійну регресійну модель тривалості розробки Java-застосунків для платформи midrange в залежності від трудомісткості, шляхом побудови рівняння нелінійної регресії, рівняння границь довірчого інтервалу та рівняння границь інтервалу прогнозування нелінійної регресії для тривалості розробки Java-застосунків для платформи. 3. На основі удосконалених моделей розробити програмне забезпечення для оцінювання тривалості розробки Java-застосунків для платформи.

Об'єктом дослідження є процес оцінювання тривалості розробки Java-застосунків для платформи midrange.

Предметом дослідження є математичні моделі для оцінювання тривалості розробки Java-застосунків для платформи midrange.

Методи дослідження. Для вирішення поставлених завдань в роботі були застосовані методи теорії ймовірностей, математичної статистики, регресійного аналізу.

Наукова новизна одержаних результатів. Отримала подальший розвиток модель ISBSG для оцінювання часу розробки застосунків для платформи midrange в залежності від трудомісткості їх розробки за рахунок створення нелінійної регресійної моделі для Java-застосунків, яка побудована з урахуванням викидів. Це дозволяє підвищити достовірність оцінювання тривалості розробки Java-застосунків для платформи midrange.

Практичне значення одержаних результатів. Розроблене програму для оцінювання тривалості розробки Java-застосунків для платформи midrange, на мові Java, що дозволяє скоротити час відповідного оцінювання, забезпечує зберігання результатів оцінювання, а також надає користувачу швидкий доступ до результатів попередніх оцінювань.

Ключові слова: ТРУДОМІСТКІСТЬ, ТРИВАЛІСТЬ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, НЕЛІНІЙНА РЕГРЕСІЯ, РЕГРЕСІЙНА МОДЕЛЬ, JAVA-ЗАСТОСУНОК, ПЛАТФОРМА MIDRANGE

ABSTRACT

Farionova Tetyana Anatoliyivna

"A nonlinear regression model for estimating the development duration of Java applications for the midrange platform and creating the software for its implementation "

The qualification work for the degree of higher education Master's degree in specialty 121 - "Software Engineering" (EP "Software Engineering"). Admiral Makarov National University of Shipbuilding. Mykolaiv, 2021

The qualification work is presented on the 133 pages of typewritten text, contains 14 tables, 28. figures, 5 appendices and 46 references.

Relevance of the topic of the work. According to statistics, about 30% of software projects have difficulties with implementation (not invested in the budget, come out in violation of time constraints, or inadequate quality). Software projects for the midrange platform are characterized by their large size, large number of components and special requirements for reliability. According to ISBSG statistics, the share of software for the midrange platform, the main language of which is Java, is about 25%. Therefore, a reliable estimate of the duration of development of Java-applications for this platform is an urgent task and is of scientific and practical interest.

The purpose and objectives of the study. The aim is to increase the reliability of estimating the development duration of Java applications for the midrange platform by building a nonlinear regression model. Objectives of the study: 1. To analyze existing methods and models for estimating the duration of software projects. 2. Build a nonlinear regression model of Java application development duration for the midrange platform depending on effort, by constructing a nonlinear regression equation, a confidence interval boundary equation, and a nonlinear regression prediction interval equation for the Java application development duration for the platform. 3. Based on advanced models, develop software to estimate the duration of development of Java applications for the platform.

The object of the study is the process of estimating the duration of development of Java-applications for the midrange platform.

The subject of the study is mathematical models for estimating the duration of development of Java applications for the midrange platform.

Research methods. Methods of probability theory, mathematical statistics, and regression analysis were used to solve the tasks.

Scientific novelty of the obtained results. The ISBSG model for estimating application development duration for the midrange platform, depending on the effort of their development, was further developed by creating a nonlinear regression model for Java applications, which is built with emissions in mind. This increases the reliability of estimating the development duration of Java applications for the midrange platform.

The practical significance of the results obtained. A program for estimating the duration of development of Java applications for the midrange platform has been developed, which reduces the duration of the corresponding evaluation, provides storage of evaluation results, and provides the user with quick access to the results of previous evaluations.

Keywords: EFFORT, DURATION OF SOFTWARE DEVELOPMENT, NONLINEAR REGRESSION, REGRESSION MODEL, JAVA APPLICATION, MIDRANGE PLATFORM.

ЗМІСТ

ВСТУП

Актуальність теми.

Згідно з PMBOK та стандарту ISO/IEC 12207 – Software life cycle processes – процес оцінювання тривалості робіт з розробки програмного забезпечення (ПЗ) є невід'ємною складовою інженерії вимірювання ПЗ [1]. За статистикою біля 30% програмних проєктів мають труднощі з реалізацією (не вкладаються в бюджет, виходять з порушенням часових обмежень, або неналежної якості). Програмні проєкти для платформи midrange характеризуються своїм значним розміром, великою кількістю компонентів та спеціальними вимогами до надійності. За статистичними даними компанії ISBSG (International Software Benchmarking Standards Group) (https://www.isbsg.org/about-isbsg), частка ПЗ для платформи midrange, основною мовою розробки якого є Java, складає приблизно 25%. Тому достовірне оцінювання тривалості розробки Java-застосунків для цієї платформи є актуальним завданням.

На теперішній час існує велика кількість досліджень та наукових праць, які присвячені оцінюванню тривалості розробки програмних проєктів [2-28]. Слід зазначити, що більшість існуючих моделей для оцінювання тривалості розробки програмних проєктів є нелінійними регресійними моделями, що базуються на використанні нормалізуючих перетворень. До таких можна віднести відому модель COCOMO [6], моделі на основі нормалізуючого перетворення Джонсона [7, 8], моделі ISBSG [9, 10].

У зв'язку з тим, що емпіричні дані тривалості програмних проєктів та трудомісткості програмних проєктів мають закон розподілу, який відрізняється від гаусівського [6-9, 25-28], для розробки таких моделей виконувалась нормалізація емпіричних даних з тривалості та трудомісткості розробки програмних проектів. Так при побудові моделей COCOMO та ISBSG було використано десятковий логарифм для нормалізації емпіричних даних

тривалості програмних проектів та емпіричних даних трудомісткості часу розробки програм. Слід зазначити, що моделі COCOMO та ISBSG не враховують особливості програмного продукту, такі як мова та засоби розробки, сфера використання ПЗ тощо. Крім того, для цих моделей немає побудованих рівнянь довірчих інтервалів та рівнянь інтервалів прогнозування тривалості.

Програмні проєкти для платформи midrange характеризуються своїм значним розміром, великою кількістю компонентів та спеціальними вимогами до надійності. Тому підвищення достовірності оцінювання тривалості розробки Java-застосунків за рахунок побудови нелінійної регресійної моделі, а також розробка програмного забезпечення для її реалізації є актуальним та має практичну цінність.

Мета роботи. Метою роботи є підвищення достовірності оцінювання тривалості розробки Java-застосунків для платформи midrange за рахунок побудови нелінійної регресійної моделі.

Завдання дослідження. Для досягнення поставленої мети, в магістерський кваліфікаційній роботі необхідно вирішити наступні завдання:

1. Провести аналіз існуючих методів та моделей оцінювання тривалості програмних проектів.

2. Побудувати нелінійну регресійну модель тривалості розробки Java-застосунків для платформи midrange в залежності від трудомісткості, шляхом побудови рівняння нелінійної регресії, рівняння границь довірчого інтервалу та рівняння границь інтервалу прогнозування нелінійної регресії для тривалості розробки Java-застосунків для платформи.

3. На основі удосконалених моделей розробити програмне забезпечення для оцінювання тривалості розробки Java-застосунків для платформи midrange.

Об’єкт дослідження: процес оцінювання тривалості розробки Java-застосунків для платформи midrange.

Предмет дослідження: математичні моделі оцінювання тривалості розробки Java-застосунків для платформи midrange.

Методи дослідження. Для вирішення поставлених завдань в магістерський кваліфікаційній роботі були застосовані методи теорії ймовірностей, математичної статистики, регресійного аналізу.

Методи теорії ймовірностей, математичної статистики та регресійного аналізу використовувалися для побудови нелінійної регресійної моделі оцінювання тривалості розробки Java-застосунків для платформи midrange.

Наукова новизна одержаних результатів. Отримала подальший розвиток модель ISBSG для оцінювання часу розробки застосунків для платформи midrange в залежності від трудомісткості їх розробки за рахунок створення нелінійної регресійної моделі для Java-застосунків, яка побудована з урахуванням викидів. Це дозволяє підвищити достовірність оцінювання тривалості розробки Java-застосунків для платформи midrange.

Практичне значення одержаних результатів. Розроблене програму для оцінювання тривалості розробки Java-застосунків для платформи midrange, на мові Java, що дозволяє скоротити час відповідного оцінювання, забезпечує зберігання результатів оцінювання, а також надає користувачу швидкий доступ до результатів попередніх оцінювань.

Особистий внесок здобувача. Усі наукові результаті отримано здобувачем самостійно. У праці, що опублікована в співавторстві [29], здобувачу належать нелінійні регресійні рівняння для оцінювання тривалості розробки Java-застосунків платформи midrange в залежності від трудомісткості з урахуванням типів ПЗ.

Апробація результатів досліджень. Результати досліджень, викладених у кваліфікаційній роботі, пройшли апробацію на 1 конференції, II Всеукраїнської науково-практичної Інтернет конференції «Інформаційні технології: моделі, алгоритми, системи» (м. Миколаїв, 26 – 28 жовтня 2021 р.).

Публікації. Результати роботи опубліковано в 1 матеріалах конференцій.

1 АНАЛІЗ ІСНУЮЧИХ МАТЕМАТИЧНИХ МОДЕЛЕЙ ДЛЯ ОЦІНЮВАННЯ ЧАСУ РОЗРОБКИ ПРОГРАМНИХ ПРОЄКТІВ

1.1 Аналіз існуючих методів і моделей для оцінювання часу розробки програмних проєктів

На сучасний момент моделям для оцінювання часу або тривалості розробки програмних проєктів присвячено велика кількість досліджень і наукових праць. Перелік моделей для оцінювання тривалості програмних проєктів постійно збільшується, створюються все нові моделі та методи. Значний внесок у розвиток цього напряму зробив американський професор Баррі Боем (Barry Boehm) [6, 30]. Вчений класифікував основні алгоритми визначення тривалості розробки ПЗ, представив їх такими методами:

- експертного оцінювання;
- аналогій;
- дослідні та емпіричні;
- алгоритмічного моделювання;
- динамічні;
- функціональних точок;
- імітаційного моделювання;
- нейронні мережі;
- байєсовські мережі;
- COCOMO (COnstructive COst MOdel);
- оцінка для отримання договору.

Таким чином, з точки зору таксономії, методи та моделі оцінки тривалості розробки ПЗ можна розділити на дві групи: неалгоритмічні методи та алгоритмічні моделі . До неалгоритмічних методів відносяться Price-to-win, оцінка за Паркінсоном, експертна оцінка, оцінка за аналогією. До

алгоритмічних моделей відносяться COCOMO, ISBSG, нейроні мережі [31], метод критичного шляху, метод критичного ланцюга тощо.

Неалгоритмічні методи. Сутність неалгоритмічних методів у тому, що для оцінки тривалості розробки ПЗ використовуються певні схеми та принципи, а не математичні формули. До цих методів належать такі:

Price-to-win [6]. Метод ґрунтується на принципі «клієнт завжди правий». Суть методу полягає в тому, що незалежно від передбачуваних реальних витрат часу на розробку проєкту оцінка вартості ПЗ коригується відповідно до побажань замовника. Price-to-win фактично є політикою проведення переговорів із клієнтом, тому часто застосовується компаніями, які не мають коштів для якісної оцінки проектів. Застосування методу може мати для розробника такі негативні наслідки: брак ресурсів на виконання проекту, невиконання термінів здачі проекту як наслідок – втрата контракту чи банкрутство.

Оцінка за Паркінсоном [6]. Метод ґрунтується на принципі: «Обсяг роботи зростає тією мірою, якою це необхідно, щоб зайняти час, виділений на її виконання» [6, 32]. У застосуванні до розробки програмних проєктів закон Паркінсона використовується у вигляді наступної схеми: щоб підвищити продуктивність праці розробника, необхідно зменшити час, відведений на розробку [6].

Експертне оцінювання [32] широко використовується для оцінювання тривалості проектів з розробки програмного забезпечення [11, 13]. На відміну від параметричних методів оцінювання, експертні методи спираються на інтуїцію та досвід, накопичений людиною-експертом. Метод застосовується в проєктах, що використовують нові технології, нові процеси або вирішують інноваційні завдання. Припущення, на яких ґрунтується оцінка окремих експертів, заносяться до протоколу та відкрито обговорюються. Під час опитування експертів використовується Дельфійська чи розширена Дельфійська методика, орієнтована на приведення експертів до консенсусу [6]. В результаті досягається баланс оцінки при інтеграції окремих

компонентів у загальну систему. Далі слідує чергова стадія покомпонентної оцінки, зі збільшенням кількості ітерацій точність оцінки підвищується. До недоліків цього методу слід віднести наступні: підстави для отримання оцінок не є явними; складно знаходити висококваліфікованих експертів для кожного нового проекту; адекватна експертна оцінка може бути отримана лише в тому випадку, коли поточний проект і попередні є приблизно однакового трудомісткості.

Оцінка за аналогією [32] є різновидом експертної оцінки, часто виділяється як окремий метод. Оцінка за аналогією, як й алгоритмічні моделі, використовує емпіричні дані про характеристики завершених проєктів. Схема оцінки складається з декількох етапів. На першому етапі здійснюється збір даних по проекту, що розробляється. В рамках життєвого циклу ПЗ оптимальними формами для цього є аналіз вимог та проектування. На основі експертної оцінки проводиться відбір характеристик, за якими порівнюватимуться проекти. Вибір параметрів залежить від типу програми, середовища розробки та набору відомих параметрів програми. Наступний етап включає пошук і аналіз проєктів «аналогічних» за обраними характеристиками розроблюваному. Результатом цього етапу є, як правило, кілька проєктів, що мають найменші відмінності в чисельних значеннях характеристик оцінки. Для відбору проектів, найбільш близьких до розроблюваного, може використовуватися метод вимірювання Евклідової відстані в n-вимірному просторі. Кожній характеристиці надається значення ваги (множник), що визначає важливість характеристики для проекту. У спрощеному варіанті вага дорівнює одиниці, тобто всі характеристики проекту вважаються рівнозначними за важливістю. Далі проєкти та їх відповідні характеристики відображаються в n – мірному просторі як точки (n дорівнює кількості змінних, для кожної змінної використовується свій вимір), після чого обчислюється Евклідова відстань між відповідними точками [33]:

$$d(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 \ldots + (a_n - b_n)^2} = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2},$$

де a і b – точки у просторі, $a_1 \ldots a_i$ та $b_1 \ldots b_i$ – координати точок. Проєкти, що мають найбільшу подібність будуть розташовані найближче, тобто Евклідова відстань у них буде найменшою. Останнім етапом є експертна оцінка проекту, що розробляється, в якій значення, взяті з аналогічного проекту, використовуються як базис оцінки.

Алгоритмічні моделі. Моделі оцінки ПЗ, які описують залежність між характеристиками проєкту та часом на його реалізацію. Алгоритмічні методи використовують ітераційний підхід, який базується на математичних моделях та формулах. При цьому вхідними даними є трудомісткість розробки програмного забезпечення проекту (розрахована у людино-годинах). Моделі поділяють за типом на лінійні, мультиплікативні, ступеневі та за використанням історичних даних на емпіричні та аналітичні. Найбільш відомими та добре документованими моделями є модель COCOMO (ступенева, емпірична) модель ISBSG (ступенева, емпірична).

Модель COCOMO. Сімейство моделей COCOMO було створено у 1981 році на основі бази даних про проекти консалтингової фірми TRW [34]. COCOMO представлено трьома моделями, які орієнтовані використання у трьох фазах життєвого циклу ПО: базова (Basic) застосовується на етапі розробки специфікацій; розширена (Intermediate) – після визначення вимог до ПЗ; Advanced – поглиблена використовується після проектування ПЗ. Загалом, рівняння моделей має вигляд:

$$E = aS^b \times EAF,$$

де E - трудомісткість (у людино-місяцях); S – розмір коду (KLOC); EAF – фактор уточнення витрат (effort adjustment factor). Параметри a і b залежать від виду програми, що розробляється.

Регресійні моделі COCOMO були побудовані для різних типів проектів:

- органічні (organic) – 2-50 тисяч рядків програмного коду (програми, призначені для проміжних розрахунків, для потреб програміста, для наукових обчислень);

- напів-розділені (semi-detached) – 50-300 тисяч рядків програмного коду

(прикладні системи, компілятори);

- вбудовані (embedded) – більше 300 тисяч рядків коду (системи контролю повітряного руху, мережі АТМ, військові системи).

Для оцінювання тривалості програмних проектів в залежності від трудомісткості регресійні моделі COCOMO можуть бути представлені наступним чином:

$D = 2{,}4 \cdot E^{1{,}05}$ (для органічного типу програмних проєктів), (1.1)

$D = 3{,}0 \cdot E^{1{,}12}$ (для напів-розділеного типу програмних проєктів), (1.2)

$D = 3{,}6 \cdot E^{1{,}20}$ (для вбудованого типу програмних проєктів), (1.3)

де D - тривалість проєкту (в місяцях), Е – трудомісткість проєкту (в людино-годинах).

Модель ISBSG [9, 10] – нелінійна регресійна модель для оцінювання тривалості програмних проектів, яка має загальний вигляд:

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

Тут D –тривалість програмних проєктів з розробки ПЗ (місяці), E – трудомісткість розробки програмних проєктів для платформи midrange (людино-годинах).

Слід підкреслити, що модель ISBSG була побудована для різних типів платформ, таких як PC, midrange, mainframe [9, 10]:

$D = 1{,}936\ E^{0{,}221}$, для платформи PC, (1.5)

$D = 0{,}548\ E^{0{,}360}$, для платформи midrange, (1.6)

$D = 0{,}458\ E^{0{,}366}$, для платформи mainframe. (1.7)

Змішані методи [36]. Змішані методи були створені для того, щоб подолати зростаючий розмір і складність програмних проектів. Вони поєднують в собі використання статистичних, алгоритмічних, математичних методів, а також методів експертного оцінювання [10].

1.2 Аналіз існуючих математичних моделей для оцінювання програмних застосунків для платформи midrange

Перед проведенням аналізу існуючих математичних моделей для оцінювання програмних застосунків для платформи midrange, необхідно представити особливості використання, як самих програмних застосунків, так й в цілому платформи midrange. Midrange – клас комп'ютерних систем, який знаходиться між мейнфреймами і мікрокомп'ютерами [35]. Це системи середнього рівня, в першу чергу високопродуктивні мережеві сервери та інші типи серверів, які можуть обробляти великомасштабну обробку багатьох бізнес-додатків. Вони не такі потужні, як мейнфрейми, однак їх покупка, експлуатація та обслуговування дешевше, ніж у мейнфреймів, і, таким чином, вони задовольняють обчислювальні потреби багатьох організацій. Midrange стали популярними як потужні мережеві сервери, які допомагають керувати великими веб-сайтами в Інтернеті, корпоративними інтрамережами, тощо. Сьогодні системи середнього рівня включають в себе сервери, що використовуються на промислових підприємствах і виробничих підприємствах, грають важливу роль в автоматизованому виробництві (CAM). Вони також можуть приймати форму потужних технічних робочих станцій для автоматизованого проектування (САПР) та інших обчислювальних і графічних додатків. Системи середнього рівня також використовуються в якості зовнішніх серверів для допомоги мейнфрейм-комп'ютерам в телекомунікаційній обробці та управлінні мережею.

Згідно з PMBOK та стандарту ISO/IEC 12207 – Software life cycle processes – процес оцінювання тривалості робіт з розробки програмного забезпечення (ПЗ) є невід’ємною складовою інженерії вимірювання ПЗ [1]. За статистикою біля 30% програмних проєктів мають труднощі з реалізацією (не вкладаються в бюджет, виходять з порушенням часових обмежень, або неналежної якості). Програмні проєкти для платформи midrange характеризуються своїм значним розміром, великою кількістю компонентів та

спеціальними вимогами до надійності. Тому достовірне оцінювання тривалості розробки програмних проектів для цієї платформи є актуальним завданням.

За статистичними даними компанії ISBSG (International Software Benchmarking Standards Group) (https://www.isbsg.org/about-isbsg), частка ПЗ для платформи midrange, основною мовою розробки якого є Java, складає приблизно 25%

Найбільш поширеними моделями, для оцінювання тривалості програмних проектів є COCOMO [6] та ISBSG [9, 10]: нелінійні регресійні рівняння для оцінювання тривалості програмних проектів в залежності від трудомісткості. У зв'язку з тим, що емпіричні дані тривалості програмних проектів та трудомісткості програмних проектів мають закон розподілу, який відрізняється від гаусівського [6, 7, 10], для побудови таких моделей виконувалась нормалізація емпіричних даних з тривалості та трудомісткості розробки програмних проектів за допомогою десяткового логарифму.

Слід зазначити, що тільки модель ISBSG враховує особливості платформи, для якої виконується створюється програмного забезпечення (ПЗ), але не враховує мову розробки ПЗ та особливості його призначення.

Для побудови моделі ISBSG для платформи midrange, було використано десятковий логарифм для нормалізації емпіричних даних тривалості програмних проєктів та емпіричних даних трудомісткості програмних проєктів [10]:

$$D = 0{,}548\ E^{0{,}360}.$$

Тут D –тривалість програмних проєктів з розробки ПЗ (місяці), E – трудомісткість розробки програмних проєктів для платформи midrange (люд.-год.). Однак, слід зазначити, що модель (1.7) була розроблена за даними проєктів 1989-1996 рр. Для оцінки моделі (1.7) в кваліфікаційній роботі було використано дані з 529 програмних проєктів (рис. 1.1) з репозитарію ISBSG (International Software Benchmarking Standards Group, 2021 р.).

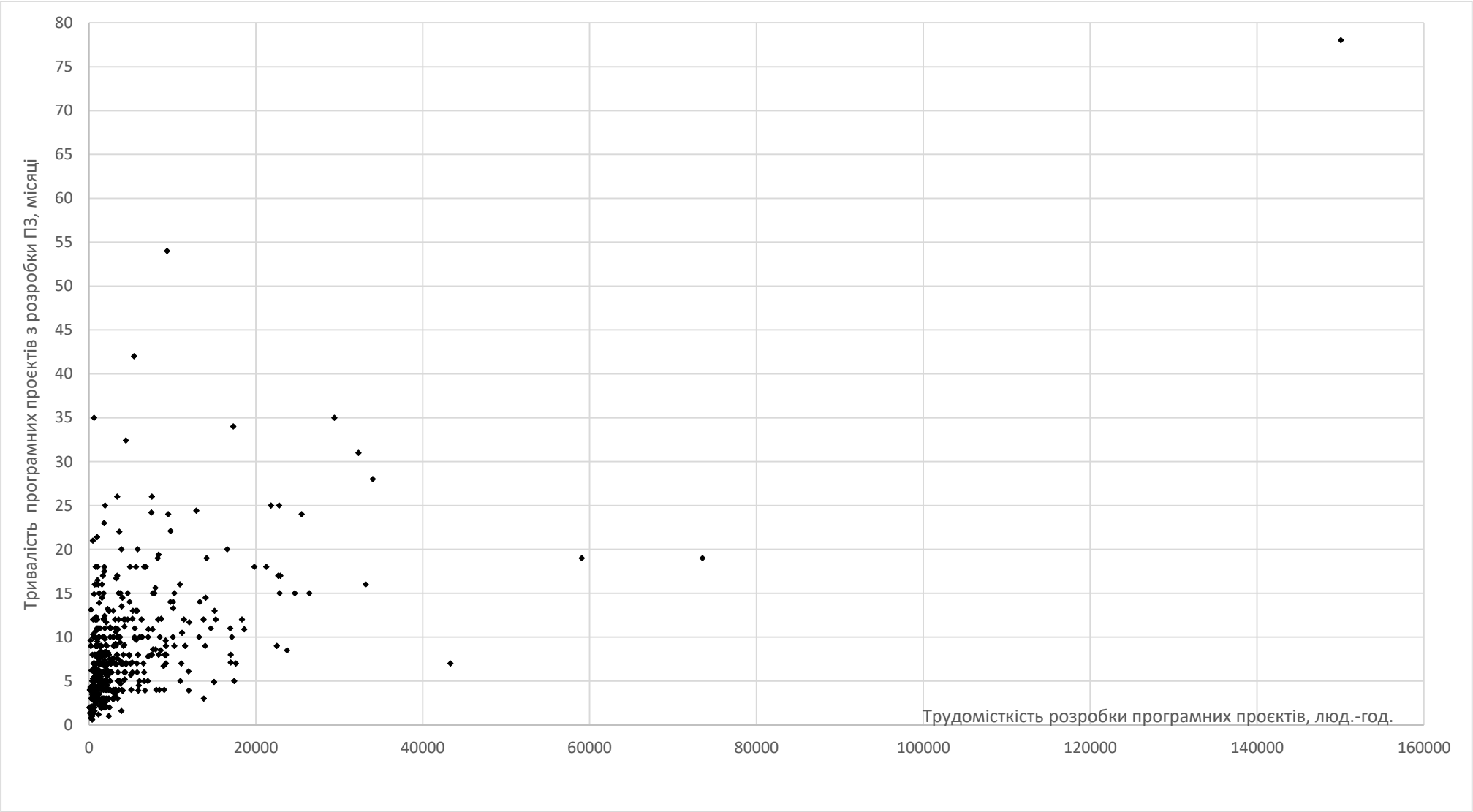

Рисунок 1.1 - Тривалость проєктів з розробки програмного забезпечення в залежності від трудомісткості цих проектів для платформи midrange (за даними репозитарію ISBSG (2021 р.))

За розрахунками отримані такі результати якості моделі ISBSG: R^2=0,253; MMRE=0,708; PRED(0,25)=0,301. Аналізуючи результати, можна зробити висновок, що модель (1.7) має невисоку якість. Крім того, модель ISBSG для платформи midrange (1.7) не враховує ймовірнісній характер трудомісткості розробки програмних застосунків та випадкову похибку.

Побудові нелінійних регресійних моделей тривалості проектів з розробки програмного забезпечення в залежності від трудомісткості цих проектів для платформи midrange присвячена робота [36], в якій запропоновано використовувати для нормалізації емпіричних даних перетворення Джонсона та десятковий логарифм.

1) Нелінійна регресійна модель тривалості проектів з розробки програмного забезпечення в залежності від трудомісткості цих проектів із використанням перетворення у вигляді десяткового логарифм має вигляд:

$$\widehat{D}(E) = 0{,}453E^{0{,}362}. \tag{1.8}$$

Показники якості моделі: R^2=0,2025; SSE=22766; PRED(0,25)=0,3529.

2) Нелінійна регресійна модель на основі нормалізуючого перетворення Джонсона для платформи midrange:

- рівняння нелінійної регресії

$$\widehat{D}(E) = \varphi_D + \frac{p\lambda_D}{p+1}, \qquad \varphi_E < E < \varphi_E + \lambda_E, \tag{1.9}$$

де

$$p = exp\left\{\frac{(\hat{z}_D(z_E) - \gamma_D)}{\eta_D}\right\};\ \ \hat{z}_D(z_E) = b_0 + b_1 z_E;$$

$$z_E = \gamma_E + \eta_E ln\left(\frac{E-\varphi_E}{\lambda_E+\varphi_E-E}\right),$$

$$\gamma_E = 4{,}0686, \eta_E = 0{,}77997, \varphi_E = 82{,}51282, \lambda_E = 565242.$$

$$[\overline{D}(E)] = \varphi_D + \frac{[\bar{q}]\lambda_D}{[\bar{\bar{q}}] + 1},\ \ \varphi_E < E < \varphi_E + \lambda_E,$$

де $[\bar{q}]$ – довірчий інтервал лінійної регресії.

- рівняння нижньої та верхньої границь інтервалу прогнозування нелінійної регресії

$$[D(E)] = \varphi_D + \frac{[q]\lambda_D}{[q]+1}, \quad \varphi_E < E < \varphi_E + \lambda_E,$$

де $[q]$ –інтервал прогнозування лінійної регресії.

Характеристики отриманих рівнянь нелінійної регресії тривалості в залежності від трудомісткості для платформи midrange були наступні [36]:

Показники якості моделі: R^2=0, 0,2033; SSE = 22744; PRED(0,25)=0,3588.

1.3 Обґрунтування необхідності проведення досліджень за обраною тематикою

Аналізуючи результати, щодо якості існуючих математичних моделей для оцінювання програмних застосунків для платформи midrange (1.6), (1.8) та (1.9) можна зробити висновок, що вони мають невисоку якість. Крім того, для розробки цих моделей були використані застарілі данні, не було враховано тип розробляємого ПЗ та мова розробки.

Варто підкреслити, що підвищення достовірності оцінювання тривалості таких програмних проєктів для платформи midrange, може значно зменшити можливі втрати від недостовірного оцінювання. Тому підвищення достовірності оцінювання часу розробки застосунків для платформи midrange є актуальним завданням.

Для підвищення достовірність оцінювання часу розробки проєктів зі створення програмного забезпечення, необхідно побудувати моделі тривалості, які будуть краще враховувати розподіл емпіричних даних, а також дозволять виконувати оцінювання довірчих інтервалів тривалості проектів з розробки програмного забезпечення, тому що інтервальна оцінка дозволить враховувати ймовірнісній характер точкових оцінок [30, 36].

Для вдосконалення існуючих математичних моделей для оцінювання часу розробки програмних застосунків для платформи midrange (1.6), (1.8), (1.9) для оцінювання тривалості розробки програмних проектів для midrange

як додаткову ознаку впливу визначимо мову ПЗ. Розробимо нелінійну регресійну модель для оцінювання часу розробки Java-застосунків для платформи midrange, видаливши викиди з емпіричних даних та побудувавши рівняння границь довірчого інтервалу, рівняння границь інтервалу прогнозування нелінійної регресії тривалості програмних проєктів в залежності від трудомісткості розробки Java-застосунків. На основі нової моделі необхідно розробити програмне забезпечення для оцінювання часу розробки Java-застосунків для платформи midrange.

2 ПОБУДОВА НЕЛІНІЙНОЇ РЕГРЕСІЙНОЇ МОДЕЛІ ОЦІНЮВАННЯ ЧАСУ РОЗРОБКИ JAVA-ЗАСТОСУНКІВ ДЛЯ ПЛАТФОРМИ MIDRANGE

2.1 Загальний алгоритм побудови нелінійної регресійної моделі з використанням нормалізуючих перетворень

Для побудови нелінійної регресійної моделі на основі нормалізуючих перетворень необхідно виконати наступні кроки:

- провести нормалізацію емпіричних даних з використанням нормалізуючого перетворення у вигляді десяткового логаріфму;
- за нормалізованими даними побудувати лінійну регресію;
-з а рівнянням лінійної регресії з використанням зворотнього нормалізуючого перетворення, побудувати нелінійну регресійну модель, рівняння нижньої та верхньої границь довірчого інтервалу нелінійної регресії, рівняння нижньої та верхньої границь інтервалу передбачення.

Згідно з рекомендаціями [37] загальний алгоритм процесу побудови нелінійної регресійної моделі (після нормалізації емпіричних даних) має ітераційний характер та проходить в декілька етапів що представлено на рис. 2.1

Рисунок 2.1 – Загальний алгоритм побудови нелінійної регресійної моделі

2.2 Побудова нелінійних регресійних моделей з використанням нормалізуючих перетворень

Лінійна регресійна модель в загальному вигляді може бути представлена рівнянням [37]:

$$y = \hat{y} + \varepsilon,$$

де y – залежна змінна (випадкова величина); $\hat{y}$ – функція, яка визначає вид регресійної моделі; x – незалежна змінна (випадкова величина), або фактор; ε – випадкова похибка.

Найчастіше випадкові величини, що входять до регресійної моделі, мають негаусівський закон розподілу, що призводить до необхідності побудови нелінійної регресії. Для побудови нелінійних регресійних рівнянь та моделей використовуються методи на основі простого перебору, лінеаризуючих та нормалізуючих перетворень. Для уникнення простого перебору при побудові нелінійних регресійних рівнянь застосовують методи, що базуються на нормалізуючих взаємо-зворотних перетвореннях [37].

Метод нормалізуючих перетворень дозволяє 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 зворотніх перетворень:

$$z_{\text{y}} = b_1 z_x + b_0 + \varepsilon, \tag{2.1}$$

де z_x, z_{y} – нормалізовані випадкові величини x, y. В роботі будемо використовувати нормалізацію даних за допомогою десяткового логарифму, оскільки саме таке перетворення використовувалося при розробці моделей ISBSG:

$$z_x = \lg x, z_y = \lg y, \tag{2.2}$$

ε – випадкова величина, розподілена за нормальним законом, що визначається як ε~N(0,1); b_0, b_1 – коефіцієнти лінійної регресії, які знаходяться методом найменших квадратів:

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

$$b_0 = \frac{\sum_{i=1}^{n} z_{yi} - b_1 \sum_{i=1}^{n} z_{xi}}{n}. \tag{2.3}$$

Надалі для оцінювання точності побудови рівняння регресії намагаються необхідно знайти довірчий інтервал. У випадку нормального закону розподілу випадкових величин довірчий інтервал лінійного рівняння регресії можливо побудувати із використанням *t*-розподілу Стьюдента, що визначається за таблицею верхніх 100α% точок *t* -розподілу Стьюдента за рівнем значимості α/2 та кількістю ступенів вільності *n*-2 [37]:

$$y = \hat{y} \pm t_{\left(\frac{\alpha}{2}, n-2\right)} \cdot S \cdot \sqrt{\frac{1}{n} + \frac{(x - \bar{x})^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}}, \tag{2.4}$$

де $\hat{y}$ – значення *y*, розраховане за рівнянням регресії; $t_{(\alpha/2,n-2)}$ – квантіль *t*-розподілу Стьюдента; α – рівень значущості; *n* – кількість значень випадкових величин у вибірці;

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

У разі однофакторної лінійної регресії інтервал передбаченнярозраховується за формулою:

$$\hat{y}_i \pm t_{\left(\frac{\alpha}{2}, n-2\right)} \cdot S \cdot \sqrt{1 + \frac{1}{n} + \frac{(x_i - \bar{x})^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}}. \tag{2.5}$$

Використання методу нормалізуючих перетворень для оцінювання тривалості розробки програмних проєктів дає гарні результати [26, 27], застосуємо цій підхід для оцінювання часу розробки Java-застосунків для платформи midrange.

Нехай D – випадкова величина, емпіричні значення тривалості програмних проектів (в місяцях), яка залежить від випадкової величини E, емпіричні значення трудомісткості програмних проектів (в людино-годинах).

Використовуючи нормалізуюче перетворення (2.2), отримаємо нормалізовані випадкові величини та z_D та z_E.

$$z_E = \lg E, z_D = \lg D. \quad (2.6)$$

Рівняння лінійної регресії для нормалізованих значень має вигляд:

$$\hat{z}_D(z_E) = b_1 z_E + b_0 + \varepsilon,$$

де ε – випадкова величиною, що має нормальний закон розподілу, $\varepsilon \sim (0, \sigma_\varepsilon^2)$,.з дисперсією σ_ε^2 .

2.3 Оцінка адекватності нелінійної регресійної моделі

Для оцінювання якості прогнозування за допомогою регресійної моделі будемо коефіцієнт детермінації R^2 (the coefficient of determination), середню величину відносної помилки MMRE (Mean Magnitude of Relative Error) та відсоток прогнозованих результатів PRED(0,25) (Percentage of Prediction), для яких величини відносної помилки MRE (Magnitude of Relative Error) менші за 0,25, PRED(0,25).

Вибірковий коефіцієнт детермінації R^2 визначається як

$$R^2 = 1 - \frac{SS_E}{SS_T}, \quad (2.7)$$

де SSE – сума квадратів залишків, $SS_E = \sum_{i=1}^{n} \left(D_i - \widehat{D_i}\right)^2$; SST – загальна сума квадратів, $SS_T = \sum_{i=1}^{n} (D_i - \overline{D})^2$; $\overline{D}$ - середнє значення випадкової велічини D

$$\overline{D} = \frac{1}{n} \sum_{i=1}^{n} D_i. \quad (2.8)$$

При значенні $R^2 \geq 0{,}5$ вважають, що дана модель є прийнятною. Достатньо ефективною та результативною вважається модель з показником

детермінації R^2 ≥0,8. Якщо ж R^2 =1, то лінія регресії точно відповідає всім спостереженням та вимогам, а модель вважається адекватною та достовірною.

MMRE – розраховується за формулою:

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

де

$$MRE = \left|\frac{D_i - \widehat{D}_i}{D_i}\right|.$$

Рівень прогнозування PRED(0,25) розраховується за формулою:

$$\text{PRED}(0{,}25) = \frac{1}{n}\sum_{i=1}^{n}\begin{Bmatrix}1, if\ MRE \le 0.25\\ 0 \quad else\end{Bmatrix}. \tag{2.10}$$

Зазвичай вважається прийнятною точністю прогнозування за допомогою регресійних моделей , якщо $\text{PRED}(0{,}25)$≥0,75.

2.4 Побудова нелінійної регресійної моделі для оцінювання часу розробки Java-застосунків для платформи midrange.

Для створення нелінійної регресійної моделі для оцінювання часу розробки Java-застосунків для платформи midrange, скористаємось статистичними даними компанії ISBSG (International Software Benchmarking Standards Group) (https://www.isbsg.org/about-isbsg) з 129 проєктів. На рисунку 2.2 представлені емпіричні данні часу розробки Java-застосунків для платформи midrange (*D*) в залежності від трудомісткості (*E*) цих проектів для платформи midrange (за даними репозитарію ISBSG (2021 р.))

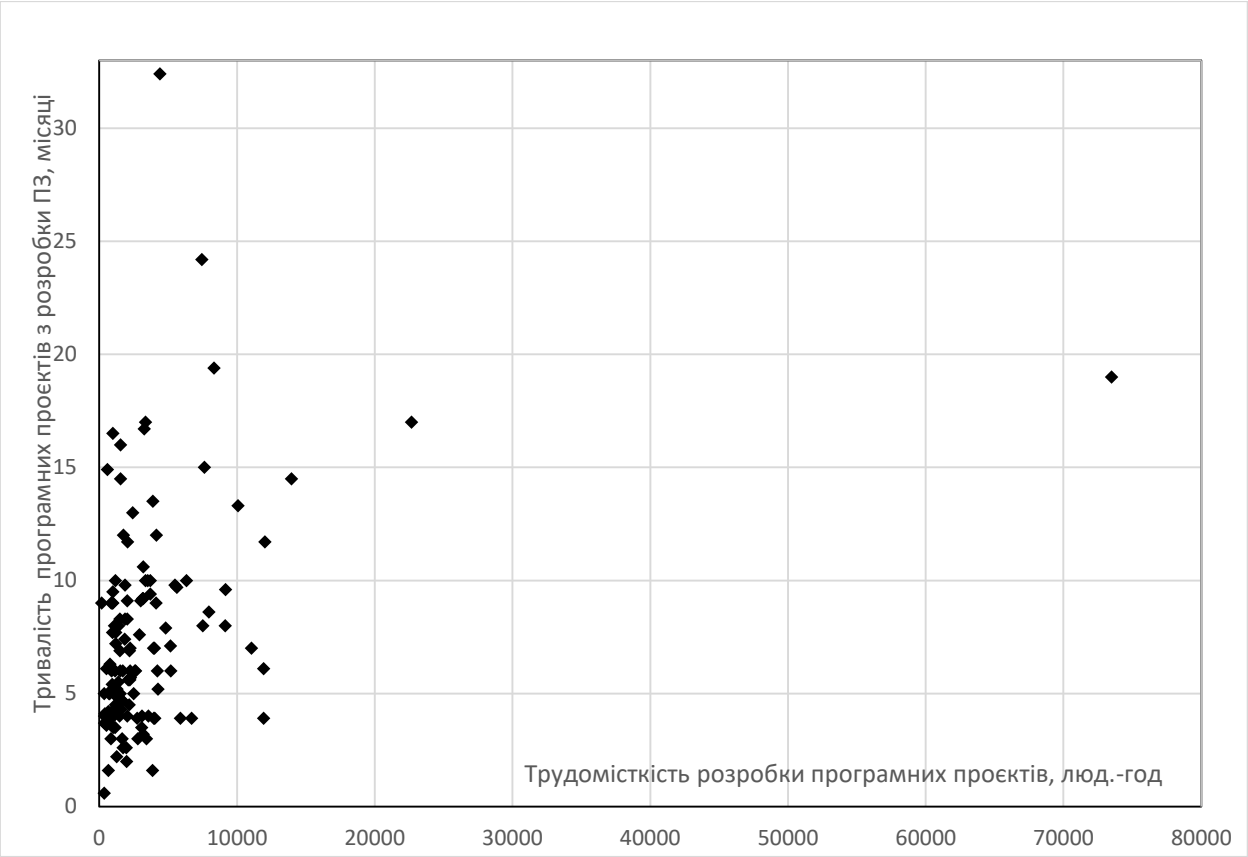

Рисунок 2.2 – Час розробки Java-застосунків для платформи midrange в залежності від трудомісткості цих проектів за даними репозитарію ISBSG (2021 р.)

Через те, що емпіричні дані часу розробки Java-застосунків для платформи midrange та емпіричні дані трудомісткості програмних проектів мають закон розподілу, який відрізняється від гаусівського закону розподілу, з'являється проблема побудови рівнянь довірчого інтервалу тривалості програмних проектів та рівнянь інтервалу прогнозування тривалості програмних проектів. Для вдосконалення нелінійної регресійної моделі ISBSG для midrange можна застосувати метод побудови рівнянь нелінійної регресії та метод побудови довірчого інтервалу регресії на основі нормалізуючих перетворень [37].

Щоб вдосконалити модель ISBSG для оцінювання тривалості розробки програмних проектів для midrange як додаткову ознаку впливу визначимо

мову розробки ПЗ. А саме побудуємо нелінійне регресійне рівняння для оцінювання часу розробки Java-застосунків платформи midrange. В якості даних було використано 129 Java проєктів з репозитарію ISBSG (див рис. 2.1).

Після нормалізації ЕД за допомогою десяткового логарифму, згідно з рекомендаціями п.2.1, перевіряємо данні на викиди, так як їх наявності в може бути причиною того, що дані не розподіляються за законом Гаусу. Таку перевірку проводимо ітераційно (етапи 1 – 3, див. рис. 2.1).

Для кожного набору значень обчислюємо квадрат відстані Махаланобіса. Якщо для будь-яких даних квадрат відстані Махаланобіса перевищує критичне значення, то з ЕД видаляється такі значення. Наступним кроком (етап 2, рис. 2.1) будуємо лінійну регресійну модель та перевіряємо закон розподілу залишків. Якщо розподіл не є нормальним, то з ЕД видаляється пару значень $(z_{\mathrm{i}_E}, z_{\mathrm{i}_D})$ з залишком найбільшим за модулем та переходимо до етапу 1 (рис. 2.1). У тому випадку, коли нормалізовані значення ЕД виходять за межи інтервалу передбачення лінійної регресії, то вони також видаляються.

За результатами розрахунків з 129 проєктів було видалено 29. Остаточно набір склав 100 пар даних залежності тривалості проєктів розробки ПЗ від їх трудомісткості.

Перевірка розподілу нормалізованих ЕД закону Гауса виконувалась за критерієм Пірсона χ^2: $\chi^2 = 8{,}491$; ${\chi_{\text{кр}}}^2 = 9{,}49$; $\chi^2 < {\chi_{\text{кр}}}^2$ (для рівня значущості 0,05 та чотирьох ступенів свободи).

Надалі для нормалізованих даних, згідно з (2.1), будуємо лінійне рівняння регресії та за допомогою методу найменших квадратів знаходимо його коефіцієнти (2.3): b_1=0,27, b_0=-0,07.

Лінійна регресійна модель має вигляд

$$z_D = 0{,}27 z_E - 0{,}07 + \varepsilon\ , \tag{2.11}$$

.

де ε – випадкова величина, що має нормальний закон розподілу, з дисперсією $\sigma_\varepsilon^2 = 0{,}0183$

За формулою (2.4) знаходимо довірчі інтервали лінійного рівняння регресії та інтервали передбачення за формулою (2.5).

Значення *t*–розподілу Стьюдента, що використовувалось при побудові рівнянь довірчих інтервалів та інтервалів передбачення значення:

$$t_{(0,05/2),(100-2)} = 1{,}99.$$

На рисунку 2.3. представлено нормалізовані дані, лінійне рівняння регресії та відповідні інтервали регресійної моделі.

Всі нормалізовані дані знаходяться в середині інтервалу прогнозування (див. рис. 2.3). Це свідчить про те, що викиди на етапі первинної обробки даних були видалені.

Для побудови нелінійного рівняння регресії необхідно використати побудоване лінійне рівняння регресії (2.11) та зворотне нормалізуюче перетворення

$$\mathrm{E} = 10^{z_E}, D = 10^{z_D} \qquad (2.12)$$

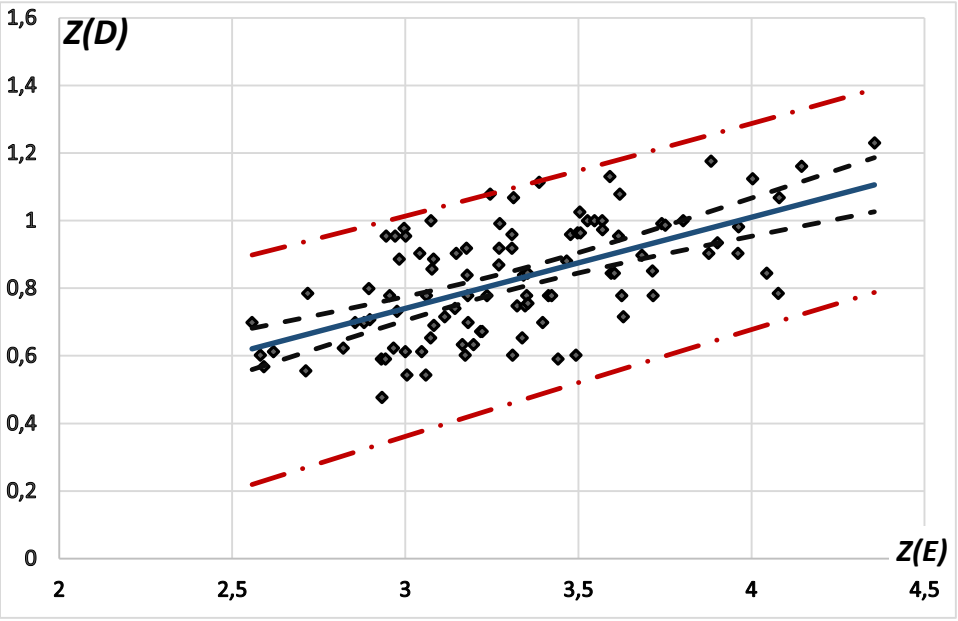

Рисунок 2.3 – Нормалізовані ЕД, лінійне рівняння регресії, довірчий інтервал та інтервал прогнозування лінійного рівняння регресії часу розробки Java-застосунків платформи midrange (n=100)

На рисунку 2.4 представлено нелінійне рівняння регресії та відповідні інтервали регресії часу розробки Java-застосунків платформи midrange.

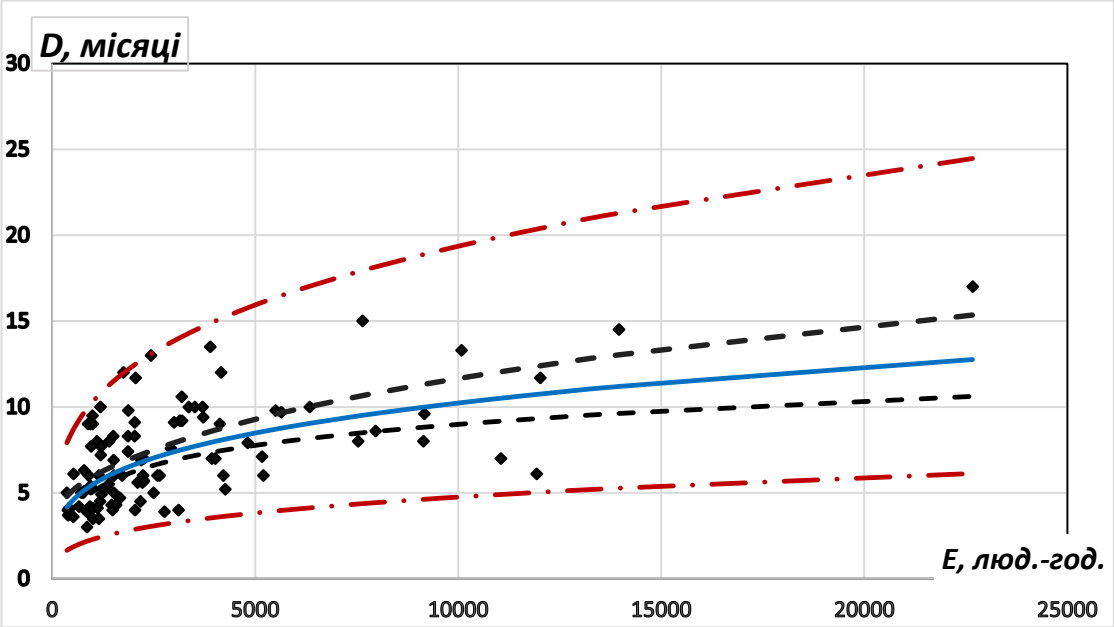

Рисунок 2.4 – Вихідні емпіричні дані, нелінійне рівняння регресії, довірчий інтервал та інтервал прогнозування нелінійного рівняння регресії часу розробки Java-застосунків платформи midrange (n=100)

Побудована нелінійна регресійна модель для оцінювання часу розробки Java-застосунків для midrange має наступні характеристики: R^2=0,3849; MMRE=0,2676; PRED(0,25)=0,57. Таким чином, ми отримали кращі оцінки за параметрами R^2, MMRE та PRED(0,25), в порівнянні з моделлю ISBSG (див. п.1.2 (1.6)): R^2=0,253; MMRE=0,708; PRED(0,25)=0,301. Слід зауважити, що отримана модель, має показники PRED(0,25)<0,75, та PRED(0,25)>0,25, що свідчить про необхідність подальшого дослідження щодо покращення нелінійної регресійної моделі для оцінювання часу розробки Java-застосунків платформи midrange.

В наборі даних Java-застосунків платформи midrange, що використовувалися в розрахунках, були присутні як нові проєкти – New Development (ND), так й такі, що вдосконалювалися – Enhancement or Re-

development (E). Тому автором було запропоновано покращити оцінку за рахунок побудови окремо нелінійного рівняння регресії як для (ND), так й для (E).

В якості даних було використано для ND Java застосунків 32 проєкти з репозитарію ISBSG.

Після нормалізації ЕД за допомогою десяткового логарифму, згідно з рекомендаціями п.2.1, перевіряємо данні на викиди, так як їх наявності в може бути причиною того, що дані не розподіляються за законом Гаусу. Таку перевірку проводимо ітераційно (етапи 1 – 3, див. рис. 2.2).

Для кожного набору значень обчислюємо квадрат відстані Махаланобіса. Якщо для будь-яких даних є перевищення критичного значення, то з ЕД видаляється пара значень з найбільшим квадратом відстанні Махаланобіса. Наступним кроком (етап 2, рис. 2.2) будуємо лінійну регресійну модель та перевіряємо закон розподілу залишків. Якщо розподіл не є нормальним, то з ЕД видаляється пару значень $(z_{\text{i}_E}, z_{\text{i}_D})$ із залишком найбільшим за модулем та переходимо до етапу 1 (рис. 2.2). У тому випадку, коли нормалізовані значення ЕД виходять за межи інтервалу передбачення лінійної регресії, то вони також видаляються.

За результатами розрахунків з 32 проєктів було видалено 2. Остаточно набір склав 30 пар даних залежності тривалості розробки нових проєктів Java застосунків від їх трудомісткості.

Перевірка розподілу нормалізованих ЕД закону Гауса виконувалась за критерієм Пірсона χ^2: $\chi^2 = 1{,}010$; $\chi_{\text{кр}}{}^2 = 5{,}99$; $\chi^2 < \chi_{\text{кр}}{}^2$ (для рівня значущості 0,05 та двох ступенів свободи). Надалі для нормалізованих даних, згідно з (2.1), будуємо лінійне рівняння регресії та за допомогою методу найменших квадратів знаходимо його коефіцієнти (2.3): b_1=0,418, b_0=-0,74.

Лінійна регресійна модель має вигляд

$$z_D = 0{,}418 z_E - 0{,}74 + \varepsilon \qquad (2.11)$$

За формулою (2.4) знаходимо довірчі інтервали лінійного рівняння регресії та інтервали передбачення за формулою (2.5).

Значення *t*–розподілу Стьюдента, що використовувалось при побудові рівнянь довірчих інтервалів та інтервалів передбачення значення:

$$t_{(0,05/2),(100-2)} = 2,048.$$

На рисунку 2.5. представлено нормалізовані дані, лінійне рівняння регресії та відповідні інтервали регресійної моделі.

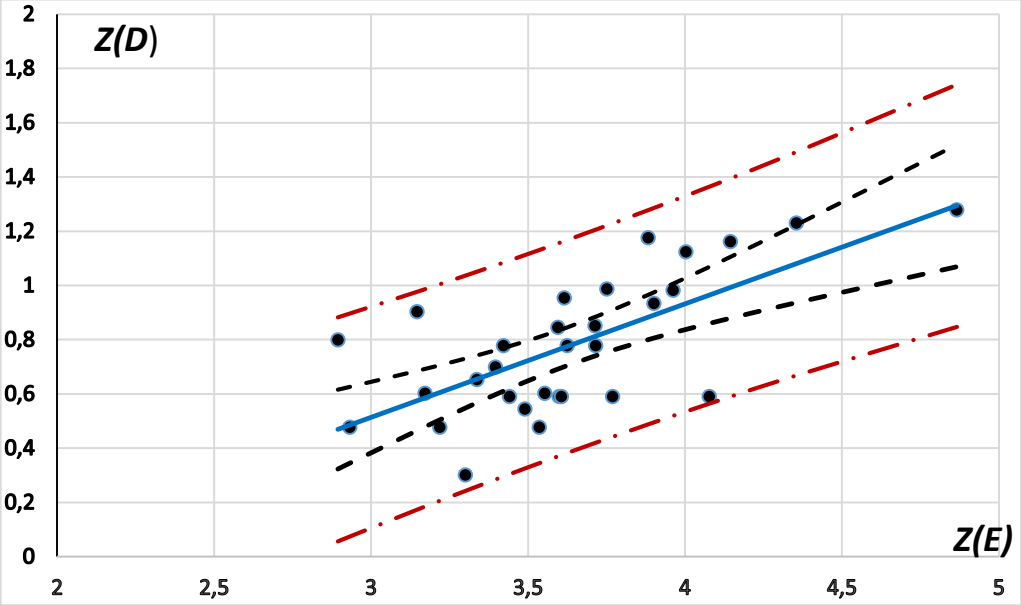

Рисунок 2.5 – Нормалізовані дані, лінійне рівняння регресії, довірчий інтервал та інтервал прогнозування лінійного рівняння регресії (New Development)

Всі нормалізовані дані знаходяться в середині інтервалу прогнозування (див. рис. 2.5). Це свідчить про те, що викиди на етапі первинної обробки даних були видалені.

Для побудови нелінійного рівняння регресії необхідно використати побудоване лінійне рівняння регресії (2.11) та зворотне нормалізуюче перетворення (2.12).

На рисунку 2.6 представлено нелінійне рівняння регресії та відповідні інтервали регресії часу розробки нових проєктів Java-застосунків платформи midrange.

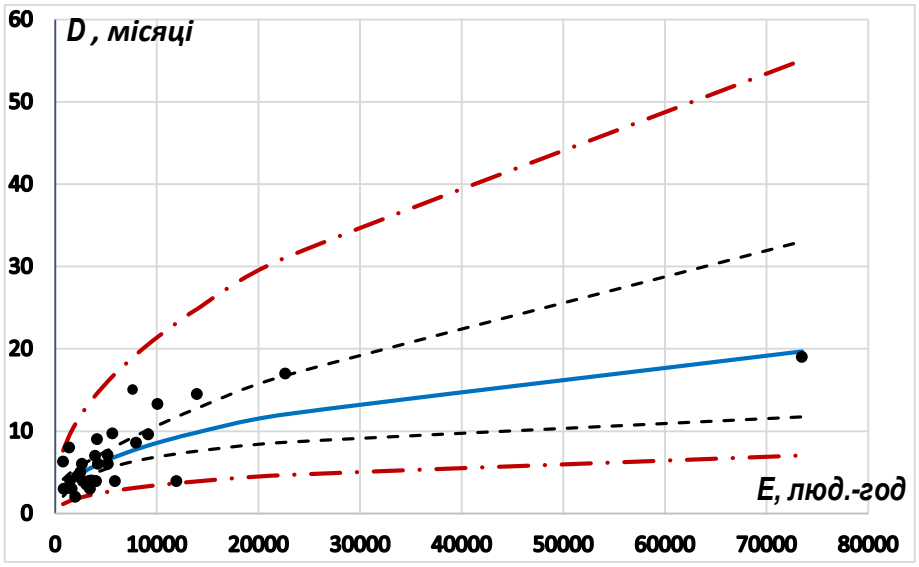

Рисунок 2.6 – Вихідні емпіричні дані, нелінійне рівняння регресії, довірчий інтервал та інтервал прогнозування нелінійного рівняння регресії (нові проєкти)

Аналогічно розробимо нелінійну регресійну модель для оцінювання часу розробки проєктів, що вдосконалюються. Первинний набір даних склав 97 проєків. За результатами розрахунків з них було видалено 20. Остаточно набір склав 77 пар даних залежності тривалості розробки проєктів Java застосунків, що вдосконалюються, від їх трудомісткості.

Перевірка розподілу нормалізованих ЕД закону Гауса виконувалась за критерієм Пірсона χ^2: $\chi^2 = 7{,}225$; ${\chi_{\text{кр}}}^2 = 9{,}488$; $\chi^2 < {\chi_{\text{кр}}}^2$ (для рівня значущості 0,05 та двох ступенів свободи).

Лінійна регресійна модель для ЕД з розробки проєктів Java застосунків, що вдосконалюються, має вигляд

$$z_D = 0{,}252 z_E - 0{,}005 + \varepsilon$$

Значення *t*–розподілу Стьюдента, що використовувалось при побудові рівнянь довірчих інтервалів та інтервалів передбачення значення:

$$t_{(0,05/2),(100-2)} = 2,0.$$

За формулою (2.4) знаходимо довірчі інтервали лінійного рівняння регресії знаходимо згідно та інтервали передбачення за формулою (2.5). На рисунку 2.7. представлено нормалізовані дані, лінійне рівняння регресії та відповідні інтервали

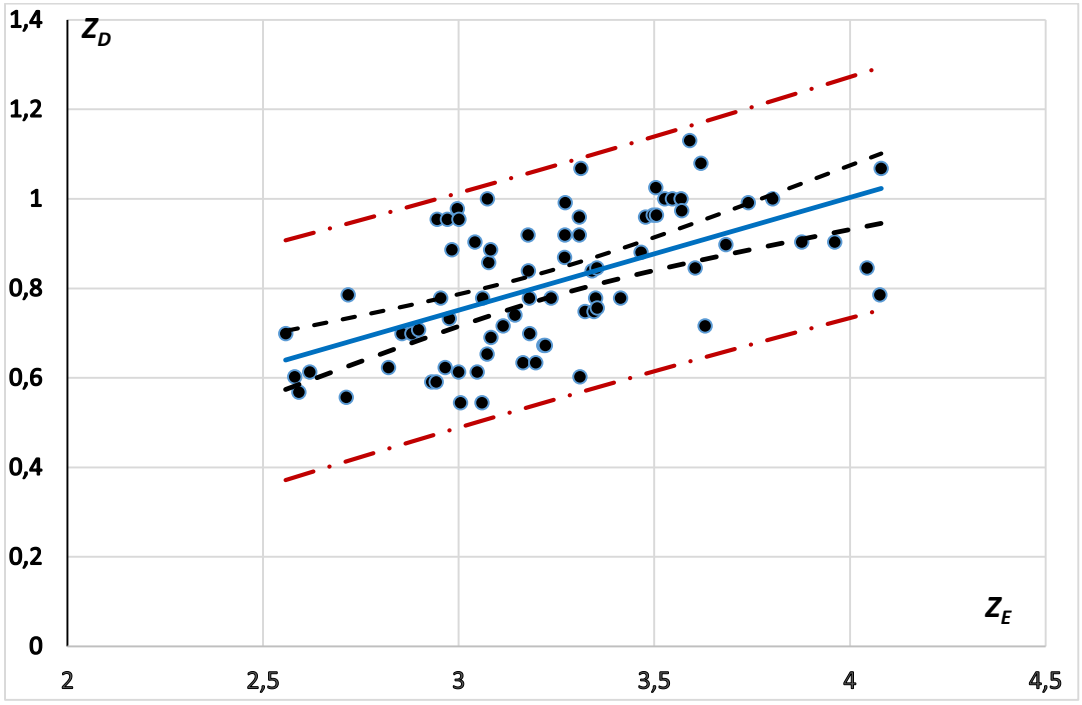

Рисунок 2.7 – Нормалізовані дані, лінійне рівняння регресії, довірчий інтервал та інтервал прогнозування лінійного рівняння регресії (проєкти, що вдосконалюються)

Всі нормалізовані дані знаходяться в середині інтервалу прогнозування (див. рис. 2.7). Це свідчить про те, що викиди на етапі первинної обробки даних були видалені.

Для побудови нелінійного рівняння регресії необхідно використати побудоване лінійне рівняння регресії (2.11) та зворотне нормалізуюче перетворення (2.12).

На рисунку 2.8 представлено нелінійне рівняння регресії та відповідні інтервали регресії часу розробки нових проєктів Java-застосунків платформи midrange.

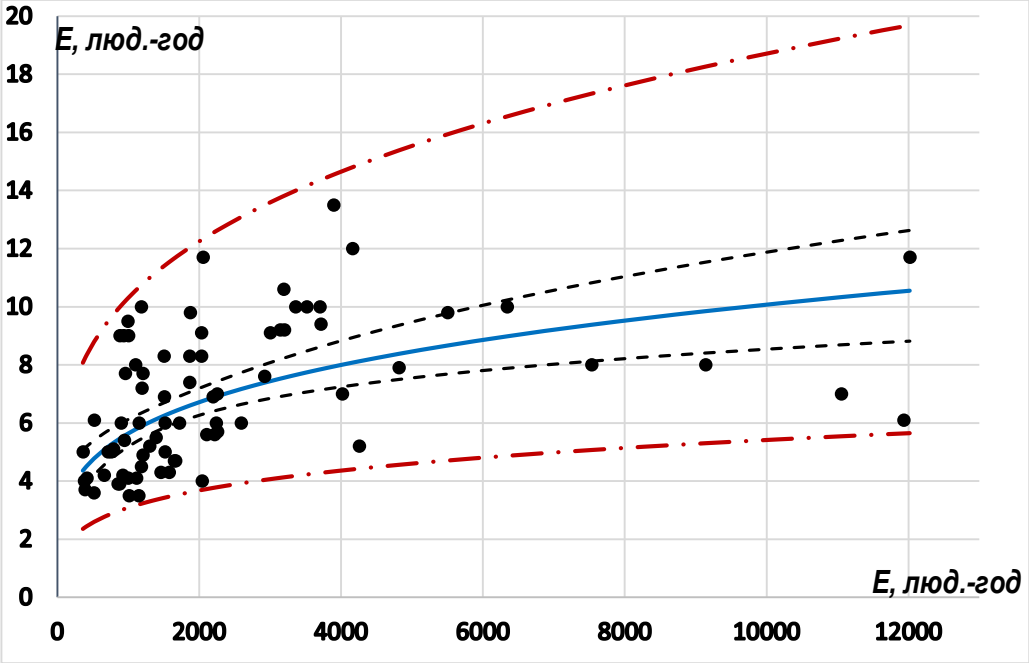

Рисунок 2.8 – Вихідні емпіричні дані, нелінійне рівняння регресії, довірчий інтервал та інтервал прогнозування нелінійного рівняння регресії (проєкти, що вдосконалюються)

В таблиці 2.1 наведені порівняльні характеристики нелінійних регресійних моделей для оцінювання часу розробки Java-застосунків для midrange в цілому, розробки нових проєктів та проєктів, що вдосконалювалися

Таблиця 2.1 – Порівняльні характеристики нелінійних моделей для оцінювання часу розробки Java-застосунків платформи midrange

Показники моделі	New Development (ND), n=30	Enhancement (E), n=70	All project, n=100
b_0	-0,74	-0,005	-0,07
b_1	0,418	0,252	0,27
R^2	0,5916	0,3251	0,3849
MMRE	0,3564	0,2979	0,2676
PRED(0,25)	0,4	0,5974	0,57

Як видно з табл. 2.1, значення наведених параметрів кращі для нелінійного регресійного рівняння для проєктів, що вдосконалюються. Однак прийнятні значення MMRE та PRED(0,25) (не більше 0,25 та не менше 0,75 відповідно) для нелінійної регресії з використанням нормалізуючого перетворення у вигляді десяткового логарифму не досягнуті, що свідчить про необхідність застосування у майбутньому більш складних перетворень, наприклад, нормалізуючого перетворення Джонсона.

Перспектива подальших досліджень полягає у побудові нелінійних регресійних моделей із врахуванням впливу додаткових чинників, таких як сфера використання програмного продукту (банківська справа, виробництво, телекомунікації тощо), функціональний розмір тощо. Слід також дослідити використання нелінійних моделей, включення інших змінних у процес моделювання та використання інших видів нормалізуючих перетворень.

3 РОЗРОБКА ПРОЄКТУ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ ЧАСУ РОЗРОБКИ JAVA-ЗАСТОСУНКІВ ДЛЯ ПЛАТФОРМИ MIDRANGE

3.1 Розробка ескізного проєкту програми для оцінювання часу розробки Java-застосунків для платформи midrange

1. Ескізний проект представляє результати зовнішнього проектування програмного забезпечення: розробляються структури вхідних та вихідних даних, потоки і інтерфейси їхнього введення і виведення, загальна технологія розв'язання задачі з використанням обчислювальної техніки і необхідна для цього інформаційна модель. Спочатку розробляється діаграма варіантів використання, виконується структуризація варіантів використання та їх специфікація, проводиться уточнення варіантів використання на діаграмах діяльності [38]. Також створюється логічний проект користувальницького інтерфейсу. Для розробки діаграм було використано UML – уніфіковану мова моделювання, що використовується у парадигмі об'єктно-орієнтованого програмування [39, 40] Для формалізації представлення сценаріїв варіантів використання побудовано діаграму діяльності, спроектований логічний проект користувальницького інтерфейсу, а також розроблена інформаційна модель за допомогою діаграми класів.

3.1.1 Моделювання варіантів використання програми для оцінювання часу розробки Java-застосунків для платформи midrange

Діаграма прецедентів (діаграма варіантів використання) – діаграма, на якій зображено відношення між акторами та прецедентами в системі. Діаграма прецедентів є графом, що відображає елементи моделі варіантів використання

та зв’язки між ними. На діаграмі прецедентів проектована програмна система представляється у вигляді множини сутностей чи акторів, що взаємодіють із системою за допомогою варіантів використання. Варіант використання використовують для описання послуг, які система надає актору [39].

Наш проєкт передбачає наявність лише однієї діючої особи – користувача програми, який буде повноцінно використовувати всі функції програмного продукту.

Виявлені варіанти використання представлені в таблиці 3.1.

Таблиця 3.1 – Реєстр варіантів використання програми

Варіант використання	Формулювання
1	2
Аутентифікація	Авторизація (реєстрація) у системі користувача
Внести назву проєкту	Внесення інформації про назву проекту
Внести трудомісткість проєкту	Внесення інформації про трудомісткість проєкту
Виконати оцінювання тривалості	Виконання оцінювання тривалості розробки програмного продукту на основі нелінійної регресії та її інтервалу передбачення та довірчого інтервалу
Зберегти результати оцінювання тривалості	Збереження результатів оцінювання тривалості розробки програмного продукту в базу даних
Переглянути результати попередніх оцінювань тривалості розробки програмних проєктів	Перегляд збережених результатів оцінювання тривалості розробки програмних проєктів

Розробимо діаграму варіантів використання програми для оцінювання часу розробки Java-застосунків для платформи midrange. Діаграма варіантів використання представлена на рисунку 3.1.

Рисунок 3.1 – Діаграма варіантів використання програми для оцінювання часу розробки Java-застосунків для платформи midrange

3.1.2 Специфікація варіантів використання програми для оцінювання часу розробки Java-застосунків для платформи midrange

Конкретизуємо варіанти використання для програми для оцінювання часу розробки Java-застосунків для платформи midrange, шляхом побудови відповідних специфікацій.

Специфікації варіантів використання

1 Аутентифікація

Короткий опис. авторизація у системі існуючих користувачів**.**

Основні дійові особи: Користувач.

Інші учасники прецеденту: немає.

Зв'язки з іншими варіантами використання: відсутні.

Передумови: відсутні.

Основний потік :

- Користувач запускає програму;
- у вікні авторизації вказує свій логін та пароль;
- при успішній авторизації працює з програмою.

Постумови: аутентифікація пройдена. При успішному завершенні варіанту використання «Аутентифікація» користувач може перейти до інших варіантів використання

Альтернативні потоки : хибний пароль або логін.

2 Внести назву проєкту

Короткий опис. Внесення інформації про назву проекту**.**

Основні дійові особи: Користувач.

Інші учасники прецеденту: немає.

Зв'язки з іншими варіантами використання: відсутні.

Передумови: необхідна попередня «Аутентифікація».

Основний потік:

-Користувач проходить аутентифікацію;

- Користувач вносить інформацію про назву проекту.

Постумови: відсутні.

Альтернативні потоки: відсутні.

3 Внести трудомісткість проєкту

Короткий опис. Внесення інформації про трудомісткість проєкту**.**

Основні дійові особи: Користувач.

Інші учасники прецеденту: немає.

Зв'язки з іншими варіантами використання: відсутні.

Передумови: необхідна попередня «Аутентифікація» та внесення інформації про назву проекту.

Основний потік:

-Користувач проходить аутентифікацію;

- Користувач вносить інформацію про трудомісткість проєкту.

Постумови: при успішному завершенні варіанту використання «Внести трудомісткість проєкту» користувач може перейти до варіанту використання «Виконати оцінювання тривалості».

Альтернативні потоки: хибний формат даних про трудомісткість проєкту.

4 Виконати оцінювання тривалості

Короткий опис. Виконання оцінювання тривалості розробки програмного продукту на основі нелінійної регресії та її інтервалу передбачення**.**

Основні дійові особи: Користувач.

Інші учасники прецеденту: немає.

Зв'язки з іншими варіантами використання: відсутні.

Передумови: необхідна попередня «Аутентифікація», «Внесення інформації про назву проекту» та «Внести трудомісткість проєкту».

Основний потік:

-Користувач проходить аутентифікацію;

- Користувач вносить інформацію про назву проєкту та трудомісткість проєкту;

- Користувач отримує оцінку середнього значення тривалості розробки програмного проєкту, значення нижньої і верхньої границь 95% довірчого інтервалу та 95% інтервалу прогнозування тривалості.

Постумови: при успішному завершенні варіанту використання стає активною кнопка «Зберегти оцінку».

Альтернативні потоки: відсутні.

5 Зберегти результати оцінювання тривалості

Короткий опис. Збереження результатів оцінювання тривалості розробки програмного продукту в базу даних

Основні дійові особи: Користувач.

Інші учасники прецеденту: немає.

Зв'язки з іншими варіантами використання: відсутні.

Передумови: необхідна попередня «Аутентифікація».

Основний потік:

-Користувач проходить аутентифікацію;

- Користувач виконує оцінювання тривалості проєкту;

- Користувач натискає кнопку «Зберегти оцінку»;

- Користувач отримує інформацію про результати оцінювання тривалості розробки збережених програмних проєктів.

Постумови: відсутні.

Альтернативні потоки: відсутні.

6 Переглянути результати попередніх оцінювань тривалості розробки програмних проєктів

Короткий опис. Перегляд збережених результатів оцінювання тривалості розробки програмних проєктів

Основні дійові особи: Користувач.

Інші учасники прецеденту: немає.

Зв'язки з іншими варіантами використання: відсутні.

Передумови: необхідна попередня «Аутентифікація».

Основний потік:

-Користувач проходить аутентифікацію;

- Користувач обирає пункт меню «Інші проєкти»;

- Користувач отримує інформацію про результати оцінювання тривалості розробки збережених програмних проєктів.

Постумови: відсутні.

Альтернативні потоки: відсутні.

3.1.3 Розробка діаграм діяльності прецедентів програмного забезпечення

Для формалізації представлення варіантів використання (див. рис. 3.1) в магістерській роботі були розроблені діаграми діяльності. На рисунках 3.2 – 3.4 представлені діаграми діяльності для деяких з основних варіантів використання.

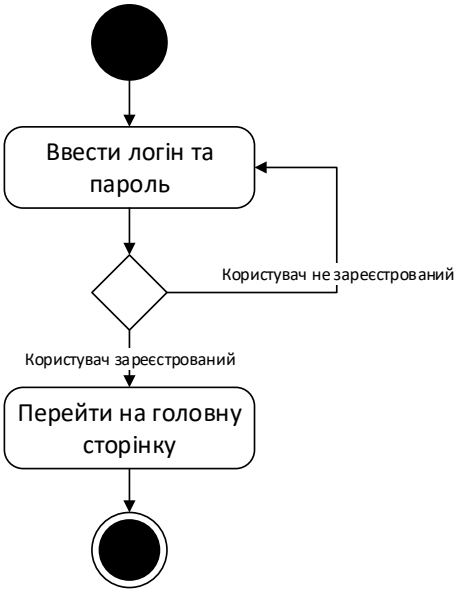

Рисунок 3.2 – Діаграма діяльності варіанту використання «Аутентифікація» програми для оцінювання часу розробки Java-застосунків для платформи midrange

Рисунок 3.3 – Діаграма діяльності варіанту використання «Виконати оцінювання тривалості» програми для оцінювання часу розробки Java-застосунків для платформи midrange

3.1.4 Проект користувацького інтерфейсу

Інтерфейс користувача є засобом зручної взаємодії користувача з програмною системою. В цьому підрозділі представлені схеми екранних форм, систем меню, діалогів і т.д. для проектування зовнішнього інтерфейсу. Розробимо проект користувацького інтерфейсу веб-додатку для оцінювання часу розробки Java-застосунків для платформи midrange.

При завантаженні браузера та переходу на сайт, користувач бачить сторінку аутентифікації (реєстрація або авторизація) (рисунок 3.4), яка призупиняє роботу програми до тих пір, поки користувач не буде аутентифікований в системі.

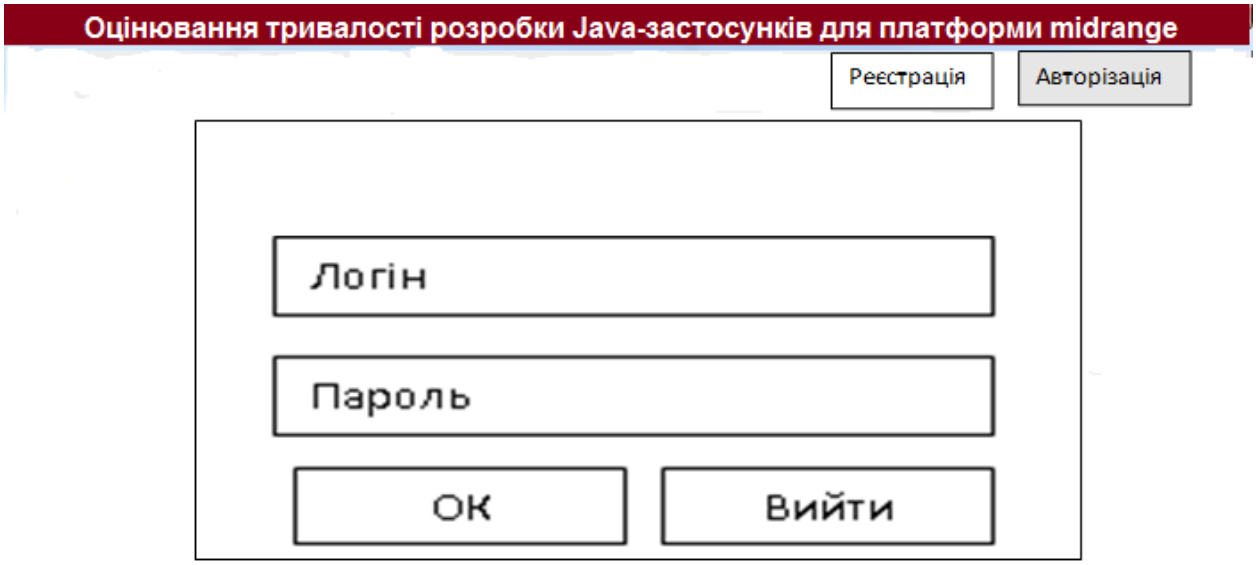

Рисунок 3.4 — Ескіз екранної форми авторизації користувача – аутентифікація

Після успішної аутентифікації користувача буде відображена головна сторінка веб-додатку для оцінювання часу розробки Java-застосунків для платформи midrange (див. рис. 3.5).

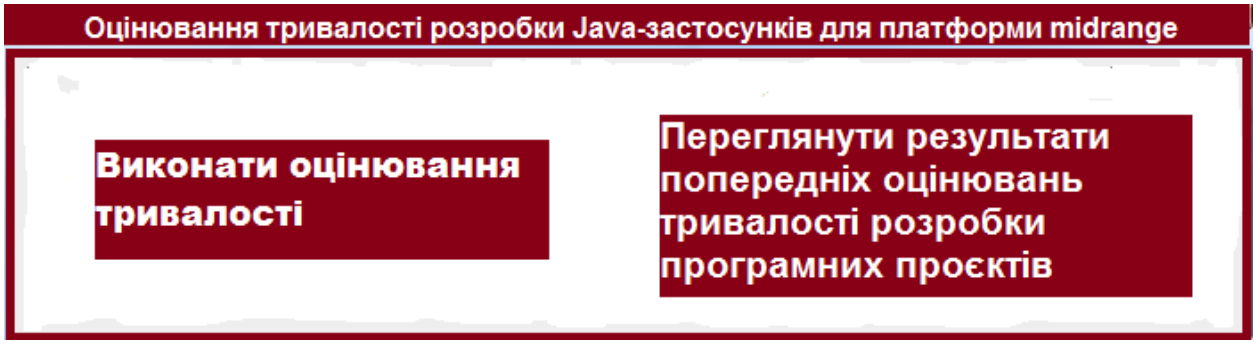

Рисунок 3.5 – Ескіз екранної форми з меню

Для введення початкових даних розрахунку було розроблено ескіз екранної форми, що представлено на рисунку 3.6.

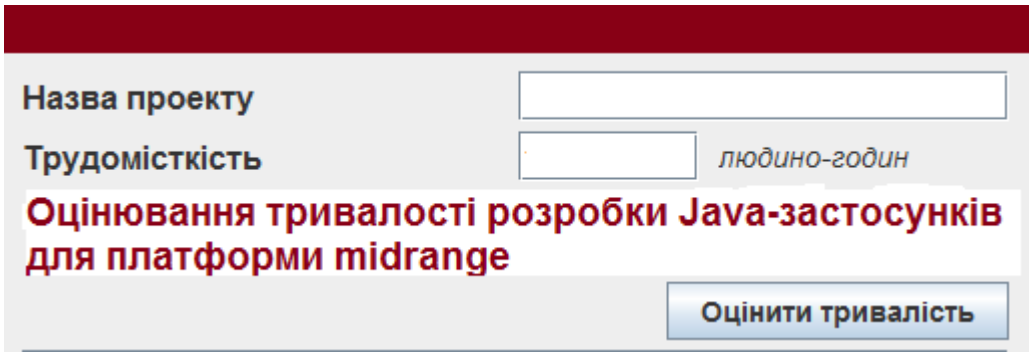

Рисунок 3.6 – Ескіз екранної форми для введення початкових даних розрахунку

Сторінка з можливістю перегляду попередніх результатів оцінювань тривалості розробки програмних проєктів наведена на рис. 3.7

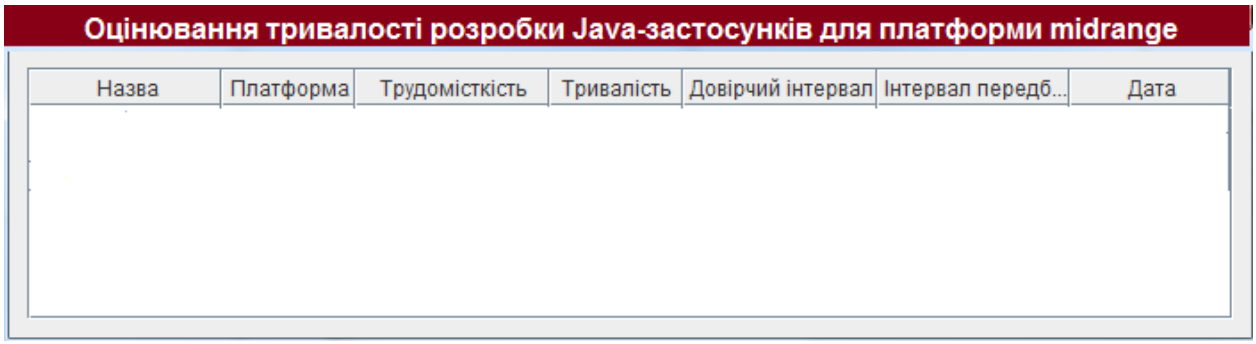

Рисунок 3.7 – Ескіз екранної форми для перегляду попередніх результатів оцінювань тривалості розробки програмних проєктів

Ескіз екранної форми для перегляду результатів розрахунку для оцінювання часу розробки Java-застосунків для платформи midrange зображений на рисунку 3.8.

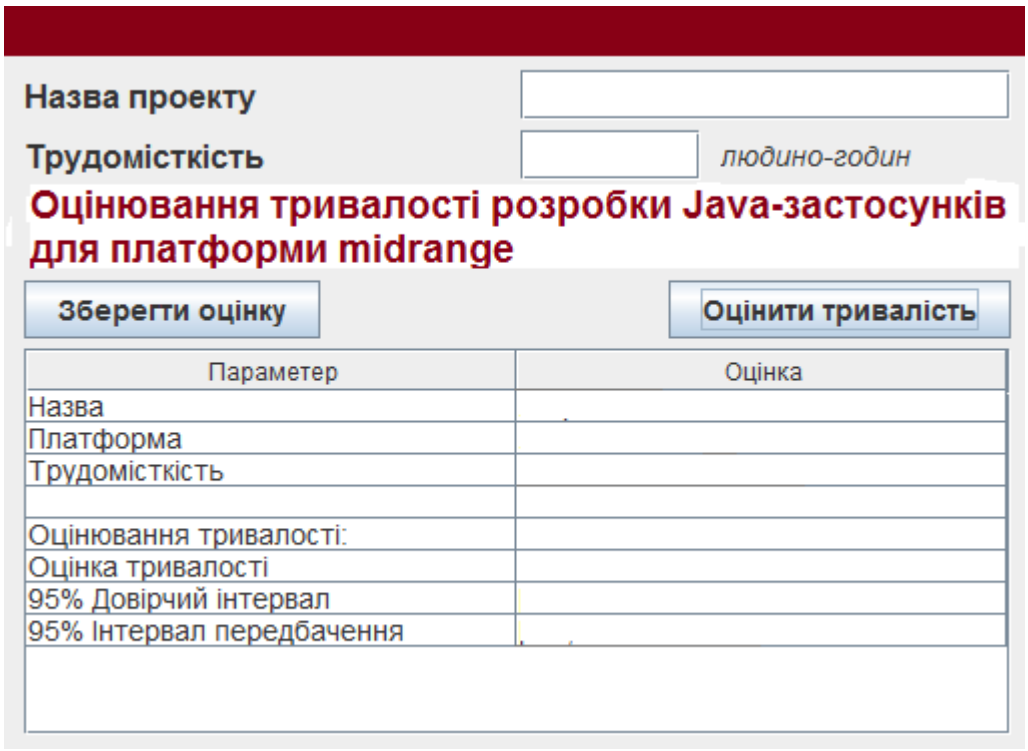

Рисунок 3.8 – Ескіз екранної форми для перегляду результатів розрахунку для оцінювання часу розробки Java-застосунків для платформи midrange

3.2 Розробка технічного проєкту програми для оцінювання часу розробки Java-застосунків для платформи midrange

Проаналізувавши ескізний проект було розроблено технічний проект програмного забезпечення веб-додатку для оцінювання часу розробки Java-застосунків для платформи midrange. Статична модель представлена діаграмами класів, а динамічна - діаграмами взаємодії.

3.2.1 Статична модель програмного забезпечення

Діаграма класів – статичне представлення структури моделі. Вона служить для представлення статичної структури моделі ПЗ в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти і кооперації, а також їхні відносини. Діаграма класів також може містити позначення для пакетів або для вкладених пакетів, деяких елементів поведінки. Однак на діаграмі класів не відображується способи поведінки цих сутностей [38, 39].

На основі моделі варіантів використання, моделі інтерфейсу програмного забезпечення була розроблена статична модель програмного забезпечення «Оцінювання тривалості програмних проектів» показана на рисунку 3.9.

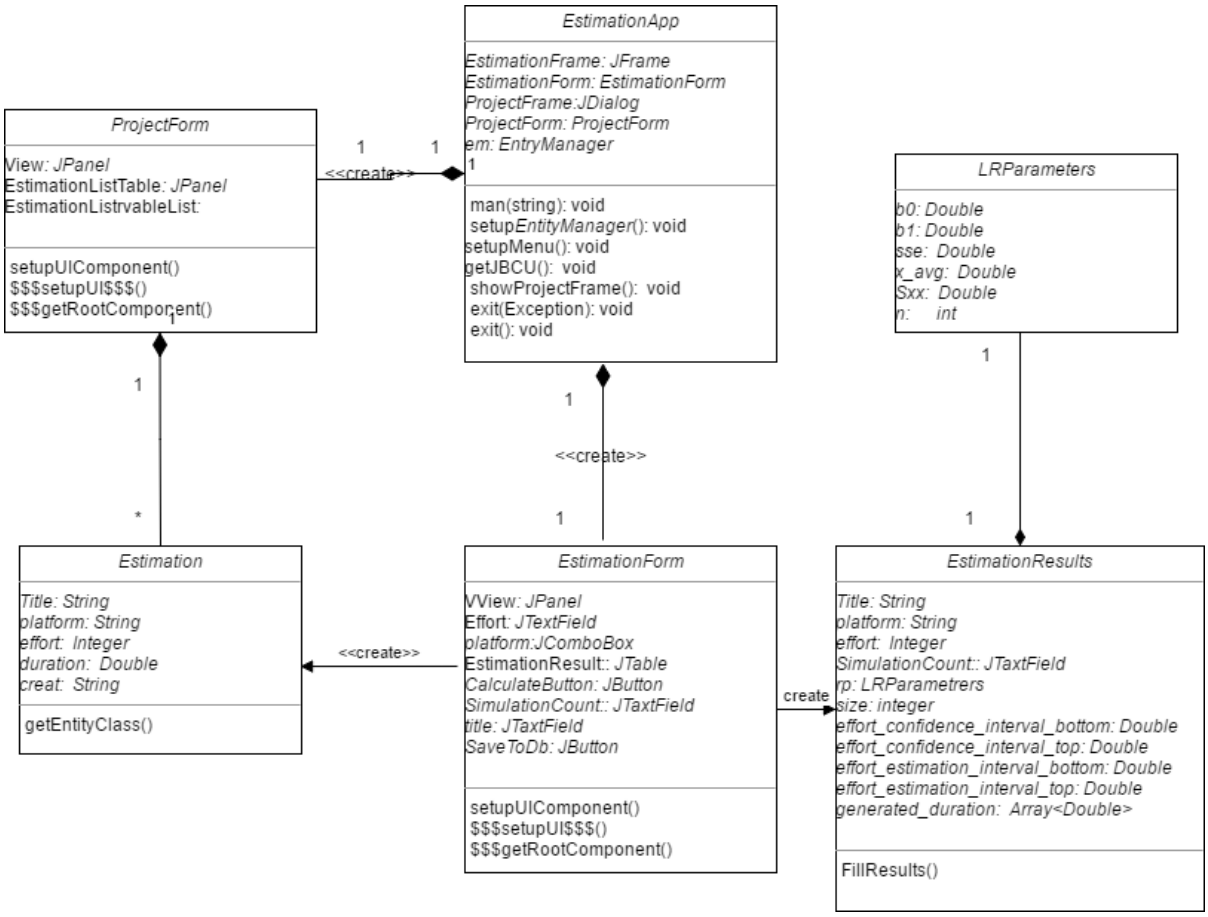

Рисунок 3.9 – Діаграма класів програмного забезпечення для оцінювання часу розробки Java-застосунків для платформи midrange

3.2.2 Логічна модель бази даних

Логічна модель таблиці параметрів математичної моделі для оцінювання часу розробки Java-застосунків для платформи midrange:

- ідентифікатор запису – числовий тип даних, первинний ключ;

- назва проєкту – строковий тип даних, заборонені дублювання;

- параметр $b0$ моделі – числовий тип даних;

- параметр $b1$ моделі– числовий тип даних;

- параметр n моделі – числовий тип даних;

- параметр S^2_{zD} – числовий тип даних;

- параметр $\bar{Z}_E$ моделі – числовий тип даних;

- параметр S_{zE} моделі – числовий тип даних.

3.2.3 Побудова динамічної моделі програмного забезпечення

Динамічна модель відтворює зміни об'єкта, які відбуваються з плином часу, або особливості функціонування об'єкта. На основі моделі сценаріїв варіантів використання і статичної моделі програми була створена динамічна модель програмного забезпечення у вигляді діаграм послідовностей. На рисунках 3.10, 3.11 наведені діаграми послідовності деяких варіантів використання, які реалізують цю модель.

У варіанті використання «Аутентифікація» користувач вводить логін та пароль. Система перевіряє чи існує такий користувач та повертає результат. Діаграма послідовності для варіанту використання «Аутентифікація» представлена на рисунку 3.10.

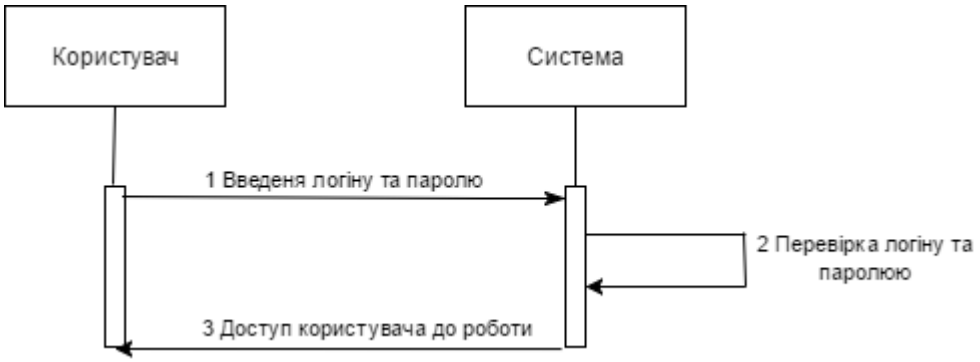

Рисунок 3.10 – Діаграма послідовності для варіанту використання «Аутентифікація»

У варіанті використання «Виконати оцінювання тривалості» користувач вводить назвупроєкту та його трудомісткість. Система перевіряє наявність та допустимість введених даних. Якщо дані коректні, повертає розраховані параметри точкової оцінки часу розробки ПЗ та границі довірчого інтервалу та інтервалу передбачення. Діаграма послідовності для варіанту використання «Виконати оцінювання тривалості» представлена на рисунку 3.11.

Рисунок 3.11 – Діаграма послідовності для варіанту використання «Виконати оцінювання тривалості»

3.3 Розробка робочого проєкту програми для оцінювання часу розробки Java-застосунків для платформи midrang

3.3.1 Обґрунтування вибору засобів розробки програми

Мовою програмування для розробки програмного забезпечення було обрано об'єктно-орієнтовану мову програмування Java. Це одна із найбільш

затребуваних мов програмування. Одночасно Java є найбільш потужною платформою розробки сучасних програмних продуктів, особливо в області web-технологій. [41].

У мови Java є суттєві переваги перед іншими мовами програмування, що дозволяє вирішувати з її допомогою практично будь-які завдання, [41] а саме:

- мова Java проста для вивчення. При розробці Java було приділено велику увагу простоті мови, тому програми на Java, в порівнянні з програмами на інших мовах, простіше писати, компілювати, налагоджувати і вивчати;
- це об'єктно-орієнтована мова, що дозволяє створювати модульні програми, вихідний код яких може використовуватися багаторазово;
- доступність компонентів для формування ергономічного візуального інтерфейсу;
- - існування математичних бібліотек, необхідних для розробки програми;
- незалежність від платформи. Це надає можливість перенесення програм з однієї системи в іншу. Оскільки програми на Java не залежать від платформи їх можна запускати в різних системах, що особливо важливо для програм, призначених для World Wide Web.
- незалежність від платформи та наявність вбудованих функцій захисту роблять цю мову програмування однією з кращих для створення додатків для Internet;
- доступність безкоштовних IDE (Integrated Development Environment – інтегрованих середовищ розробки прикладних програм) дозволяє скоротити час розробки програмного забезпечення;

В якості середовища розробки передбачається використання Intellij Idea, як найбільш багатофункціональне середовище розробки.

Для зберігання даних наша система потребує бази даних та системи управління базою даних (СУБД). Найпопулярнішими СУБД є Oracle, MySQL, MsSQL та DB2.

Oracle – об'єктно-реляційна система керування базами даних від Oracle Corporation. База даних Oracle пропонує широкий спектр можливостей та функцій у сферах доступності, масштабованості, аналітики, продуктивності, безпеки, керування та інтеграції. Вони спрямовані на вдосконалення та доповнення існуючих функціональних можливостей баз даних для задоволення конкретних вимог клієнтів . Всі параметри бази даних доступні лише для Enterprise Edition та пропонуються за додаткову плату.

MySQL — вільна система керування реляційними базами даних. MySQL був розроблений компанією «TcX» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL — одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

Microsoft SQL Server — комерційна система керування базами даних, що розповсюджується корпорацією Microsoft. Мова, що використовується для запитів — Transact-SQL, створена спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту ANSI / ISO щодо структурованої мови запитів SQL із розширеннями. Використовується як для невеликих і середніх за розміром баз даних, так і для великих баз даних масштабу підприємства. Багато років вдало конкурує з іншими системами керування базами даних.

IBM Db2 – набір серверних продуктів для баз даних, розроблених IBM. Ці продукти підтримують реляційну модель, але в останні роки деякі продукти були розширені, щоб підтримувати об'єктно-реляційні функції та не реляційні структури, такі як JSON і XML. Історично, і на відміну від інших постачальників баз даних, IBM випустив для кожного з основних операційних систем платформу Db2. Тим не менш, в 1990-х років IBM змінила шлях і випустила загальний продукт Db2, розроблений з єдиною базою коду для роботи на різних платформах.

В таблиці 3.2. наведено порівняльний аналіз популярних СКБД.

Таблиця 3.2 – Порівняльний аналіз СКБД

СКБД / Характеристика	Oracle	MySQL	MsSQL Server	DB2
1	2	3	4	5
Мова програмування	Java, Delphi PL/SQL	C / C++, SQL	Transact-SQL	Java, SQL 2000
Об’єктно-орієнтоване проектування	+	+	-	+
Мультимедійні типи даних	+	+	-	+
Максимальний розмір бази даних	Не обмежений	Не обмежений	524TB	Не обмежений
Максимальний розмір таблиці	4GB * розмір блоку	64TB (256TB)	524TB	2ZB
Максимальний розмір рядка	8KB	64KB	Не обмежений	32677B
Максимум полів у записі	1000	4096	30000	1012
Максимальний розмір типу "BLOB"	128TB	4GB	2GB	2GB
Кросс-платформенність	+ / -	+	-	+ / -
Шифрування	+	-	+	-

Проаналізувавши всі недоліки та переваги різних СКБД, було прийнято рішення використовувати СКБД MySQL. Це насамперед обумовлено зручністю використання засобів MySQL для створення програми з для оцінювання часу розробки Java-застосунків для платформи midrange, оскільки сервер MySQL забезпечується підтримкою з боку різноманітних мов програмування, зокрема і Java. MySQL задовольняє основні вимоги до СКБД

у процесі розробки програмного забезпечення, так як вона є надійною, має високу швидкість, просту у встановленні та зручну у використанні. Завдяки своїй доступності, швидкості та безпеці MySQL забезпечує ефективний доступ до баз даних.

3.3.2 Розробка фізичної моделі бази даних

Фізична модель даних описує дані засобами конкретної СКБД, в даному випадку MySQL, а також визначає розміри атрибутів та обов'язковість їх заповнення. У ході виконання роботи було побудовано фізичну модель бази даних веб-додатку для програмної реалізації моделі для оцінювання часу розробки Java-застосунків для платформи midrange.

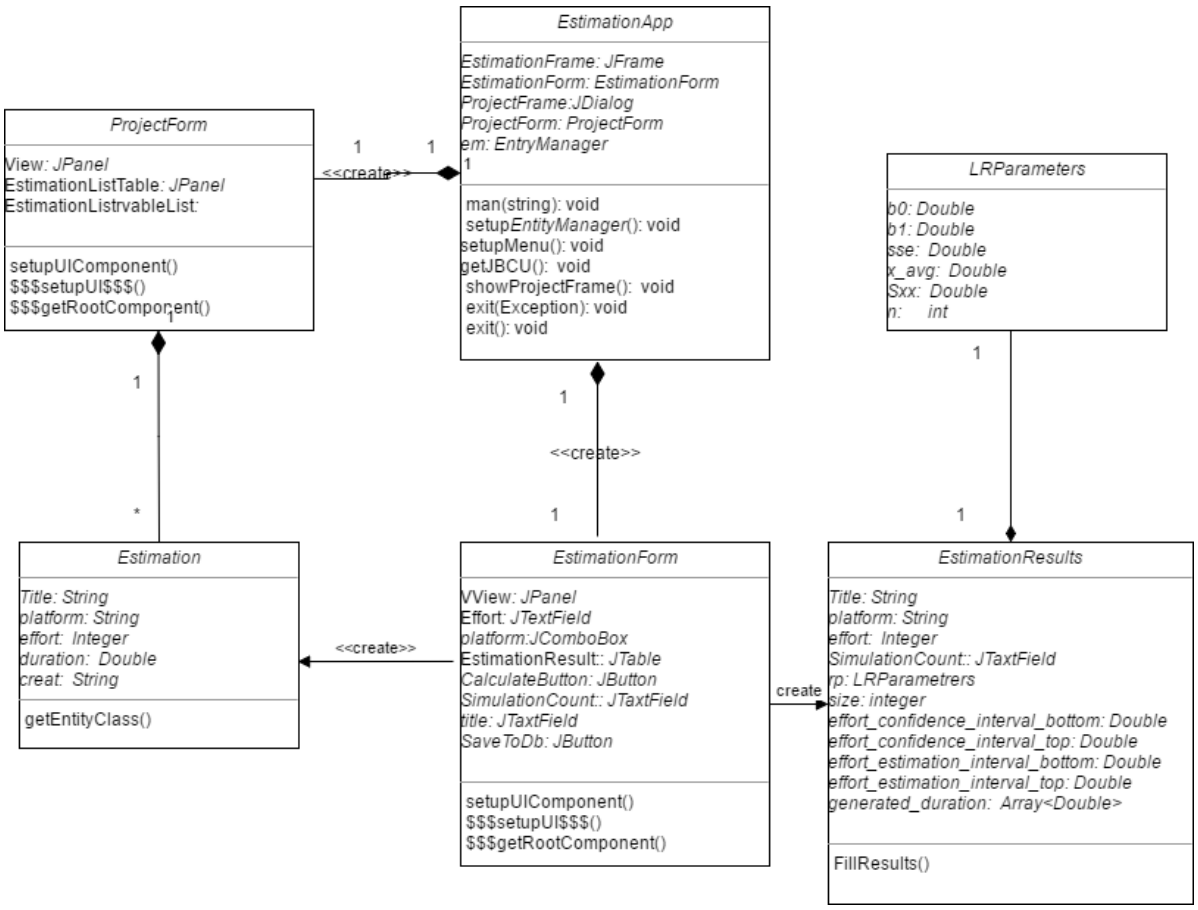

Рисунок 3.12 – Фізична модель бази даних

Фізична модель таблиці параметрів нелінійної регресійної моделі параметрів оцінювання часу розробки Java-застосунків для платформи midrange представлена в табл. 3.3.

Таблиця 3.3 - Таблиця «LRParametrs»

Поле	Тип поля	Ключ	Розмір поля	Обов. заповнення
id	Числовий	PK	Лічильник	Not null
b0	Числовий		З плаваючою точкою (8 байт)	Not null
b1	Числовий		З плаваючою точкою (8 байт)	Not null
n	Числовий		Ціле (4 байта)	Not null
sse	Числовий		З плаваючою точкою (8 байт)	Not null
avg	Числовий		З плаваючою точкою (8 байт)	Not null
Sxx	Числовий		З плаваючою точкою (8 байт)	Not null

Фізична модель таблиці результатів оцінювань та моделювань оцінювання часу розробки Java-застосунків для платформи midrange представлена в табл. 3.4.

Таблиця 3.4 - Таблиця «EstimationResult»

Поле	Тип поля	Ключ	Розмір поля	Обов. заповнення
1	2	3	4	5
id	Числовий	PK	Лічильник	Not null
title	Текстовий		Коротка строка	Not null
size	Числовий		Довге ціле	Not null
effort	Числовий		З плаваючою точкою (8 байт)	Not null
effort_estimation_interval_bottom	Числовий		З плаваючою точкою (8 байт)	Not null
effort_estimation_interval_top	Числовий		З плаваючою точкою (8 байт)	Not null
effort_confidence_interval_bottom	Числовий		З плаваючою точкою (8 байт)	Not null
effort_confidence_interval_top –	Числовий		З плаваючою точкою (8 байт)	Not null
effort_confidence_interval_top	Числовий		З плаваючою точкою (8 байт)	Not null
created	Дата		Дата	Not null

3.3.3 Діаграма компонентів програмного забезпечення

Діаграма компонентів – діаграма фізичного рівня, яка служить для подання програмних компонентів і залежностей між ними. Діаграма компонентів – статична структурна діаграма, показує розбиття програмної системи на структурні компоненти та зв'язок (залежності) між компонентами.

Для представлення компонентів системи було обрано клієнт-серверну архітектуру, яка визначає розподіл обов'язків між клієнтом та сервером.

Клієнт-серверна архітектура передбачає взаємодію двох програмних модулів – клієнтського та серверного. Так модель «тонкого» клієнта, характеризується тим, що вся логіка застосування та керування даними зосереджена на сервері. На відміну від «тонкого» клієнта, модель «товстого клієнта», передбачає те, що сервер тільки керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта. Таким є робоча станція або персональний комп'ютер, що працюють під управлінням власної дискової операційної системи і мають необхідний набір програмного забезпечення. До мережевих серверів «товсті» клієнти звертаються в основному за додатковими послугами (наприклад, доступ до web-серверу або корпоративної бази даних).

Діаграма компонентів веб-застосунку оцінювання часу розробки Java-застосунків для платформи midrange побудована відповідно до архітектури «товстого клієнту».

Для розробки було обрано архітектурний стиль MVC. Модель-вигляд-контролер - архітектурний шаблон, який використовується під час проєктування та розробки програмного забезпечення.

Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін

Діаграму компонентів представлена на рисунку 3.13.

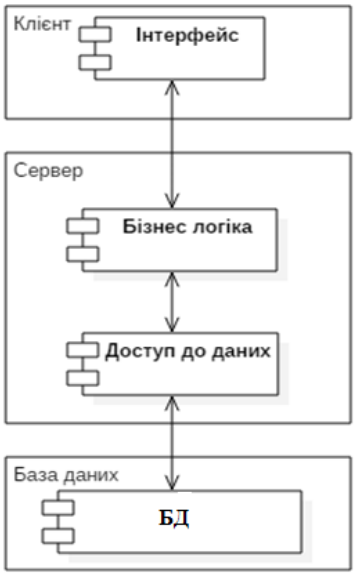

Рисунок 3.13 – Діаграма компонентів веб-застосунку для реалізації удосконаленою моделі оцінювання часу розробки Java-застосунків для платформи midrange

3.3.4 Діаграма розгортання

Модель реалізації визначає розміщення даних, методи доступу і техніку індексування. Модель реалізації задається діаграмами реалізації. Діаграми реалізації призначені для відображення складу компільованих і виконуваних модулів системи, а так само зв'язків між ними. Діаграми реалізацій поділяються на два конкретні види: діаграми компонентів і діаграми розгортання.

Діаграма розгортання показує робочі екземпляри компонент, а діаграма компонент, відображає зв'язки між типами компонент.

Діаграма розгортання моделює фізичне розгортання артефактів на вузлах. Цей вид діаграм призначений для аналізу апаратної частини системи.. Для кожної моделі створюється тільки одна така діаграма, що відображує процесори (Processor), пристрої (Device) та їх сполуки.

Для функціонування програмного забезпечення веб-додатку оцінювання часу розробки Java-застосунків для платформи midrange:

- персональний комп’ютер, ноутбук, планшет тощо з наступними характеристиками:
  - процесор ARM або x86 від 1.3GHz і вище;
  - 1ГБ оперативної пам’яті та більше;
  - можливість підключення до мережі інтернет;
- інтернет мережа

Структура технічного забезпечення системи представлено за допомогою діаграми розгортання, що показана на рисунку 3.14.

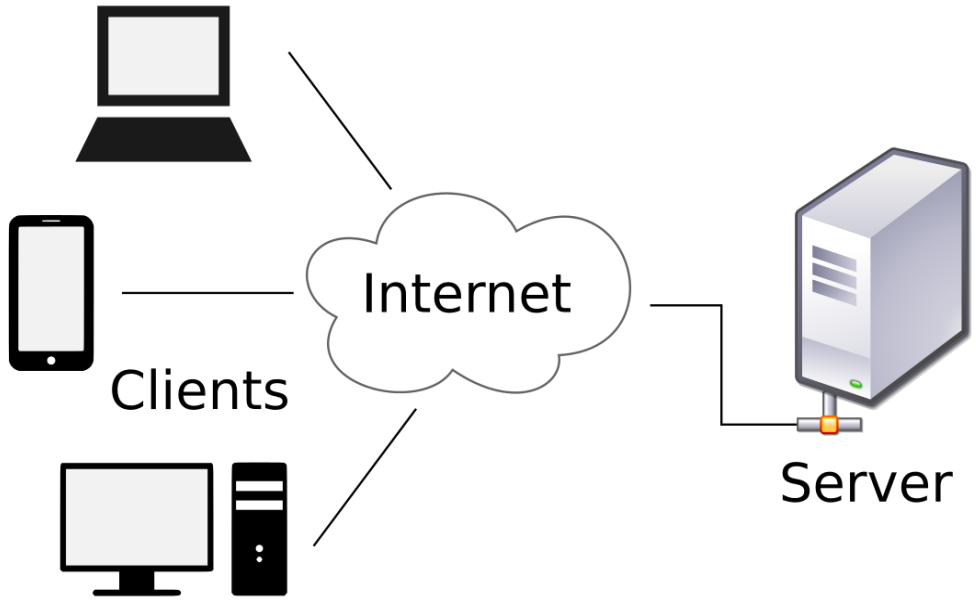

Рисунок 3.14 – Діаграма розгортання веб-додатку для реалізації удосконаленою моделі оцінювання часу розробки Java-застосунків для платформи midrange

3.3.5 Реалізація програмного забезпечення

Програмне забезпечення було реалізовано за допомогою мови програмування Java, використовуючи IDE Intellij Idea. Лістинг програмних

компонентів надано в Додатку Б «Тексти програмних модулів». Екземпляр класу EstimationResults, який реалізує оцінювання часу розробки Java-застосунків для платформи midrange приведено у Лістингу 3.1.

Лістинг 3.1 – Код классу EstimationResults

```
package EstimationApp.forms;
import DataProcess.DataAnalyze;
import DataProcess.LinearRegression;
import ISBSG.ISBSG;
import Util.MathUtil;
import javax.swing.table.DefaultTableModel;
import java.util.ArrayList;
public class EstimationResults extends DefaultTableModel {
     public String title;
     public Integer size;
     public ISBSG.LRParameters rp;
     public Double effort;
     public Double effort_confidence_interval_bottom;
     public Double effort_confidence_interval_top;
     public Double effort_estimation_interval_bottom;
     public Double effort_estimation_interval_top;

     public EstimationResults() {
          super(new String[]{"Параметер", "Оцінка"}, 0);
          addRow(new String[] {"1. Вкажіть значення
     трудомісткості ПЗ", "(ціле число)"});
          addRow(new String[] {"3. Натисніть кнопку
     \"Розрахувати\"", ""});
     }

     public void fillResults() {
          setRowCount(0);

          rp = platform.getLRParameters();
          double tStudent = DataAnalyze.getTStudent(rp.n -
     LinearRegression.k, 0.025); //p = 0.95

          double e_z = Math.log10(size.doubleValue());
          double t_z_estimate = LinearRegression.estimateY(e_z,
     rp.b0, rp.b1);
          double t_z_estimate_l1 =
     LinearRegression.estimateYConfidence(e_z, rp.b0, rp.b1,
     false, false, rp.sse, rp.x_avg, rp.Sxx, rp.n, tStudent);
          double t_z_estimate_r1 =
     LinearRegression.estimateYConfidence(e_z, rp.b0, rp.b1,
     true, false, rp.sse, rp.x_avg, rp.Sxx, rp.n, tStudent);
          double t_z_estimate_l2 =
     LinearRegression.estimateYConfidence(e_z, rp.b0, rp.b1,
     false, true, rp.sse, rp.x_avg, rp.Sxx, rp.n, tStudent);
```

```
        double t_z_estimate_r2 =
LinearRegression.estimateYConfidence(e_z, rp.b0, rp.b1,
true, true, rp.sse, rp.x_avg, rp.Sxx, rp.n, tStudent);
        effort = MathUtil.round(Math.pow(10, t_z_estimate), 1);
        effort_confidence_interval_bottom =
MathUtil.round(Math.pow(10, t_z_estimate_l1), 1);
        effort_confidence_interval_top =
MathUtil.round(Math.pow(10, t_z_estimate_r1), 1);
        effort_estimation_interval_bottom =
MathUtil.round(Math.pow(10, t_z_estimate_l2), 1);
        effort_estimation_interval_top =
MathUtil.round(Math.pow(10, t_z_estimate_r2), 1);

        addRow(new String[]{"Назва", this.title});
        addRow(new String[]{"Трудомісткість", size.toString() +
" функціональних точок"});
        addRow(new String[]{"", ""});
        addRow(new String[]{"Оцінювання тривалості:", ""});
        addRow(new String[]{"Оцінка тривалості",
effort.toString() + " людино-годин"});
        addRow(new String[]{"95% Довірчий інтервал", "[" +
effort_confidence_interval_bottom + ", " +
effort_confidence_interval_top + "] людино-годин"});
        addRow(new String[]{"95% Інтервал передбачення", "[" +
effort_estimation_interval_bottom + ", " +
effort_estimation_interval_top + "] людино-годин"});
        }
}
```

Інструкція користувача програми приведена в Додатку В.

3.3.6 Тестування та випробування програмного забезпечення

Тестування програмного забезпечення веб-додатку для реалізації удосконаленою моделі оцінювання часу розробки Java-застосунків для платформи midrange проведемо за допомогою прийому тестування методом «чорного ящика» - класів еквівалентності.

Відповідно до даної методики необхідно розбити множину значень вхідних даних на кінцеве число підмножин (які називаються класами еквівалентності), щоб кожний тест, що є представником певного класу, був еквівалентним будь-якому іншому тесту цього класу. Два тести є еквівалентними, якщо вони виявляють ті самі помилки [42].

Проектування тестів за методом класів еквівалентності проводиться у два етапи:

— виділення за специфікацією класів еквівалентності;

— побудова множини тестів.

На першому етапі відбувається вибір зі специфікації кожної вхідної умови та розбиття її на дві або більше групи, що відповідають так званим – правильним класам еквівалентності та – неправильним класам еквівалентності, тобто множинам допустимих для програми й недопустимих значень вхідних даних. Цей процес залежить від вигляду вхідної умови.

Для тестування варіанта використання «Аутентифікація» був використаний метод еквівалентної розбивки. Виділені класи еквівалентності, представлені в таблиці 3.5. Результати тестувань представлені в таблицях 3.6 та 3.7

Таблиця 3.5 – Класи еквівалентності

№	Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
1	Логін	1. Введено вірний логін	2. Введено неіснуючий логін 3. Логін не було введено
2	Пароль	4. Введено вірний пароль	5. Введено неіснуючий пароль 6. Пароль не було введено

Таблиця 3.6 – Тести із правильних класів еквівалентності

№ тесту	№ класу еквівалентності	Вхідні умови	Вихідні данні	Результат
1	1, 4	Логін: admin Пароль:admin47	Перехід до розрахунку	пройдено

Таблиця 3.7 – Тести із неправильних класів еквівалентності

№ тесту	Покритий клас	Вхідні дані	Вихідні данні	Результат тестування
3	2,5	Логін: natali Пароль: natali47	Повідомлення про помилку введення даних	Тест пройдено
4	3,6	Логін: - Пароль: -	Повідомлення про помилку введення даних	Тест пройдено

За результатами тестування варіанта використання «Аутентифікація», застосунок працює правильно, не допускаючи до введення будь-яких інших елементів, окрім допустимих символів вводу. Тестування інших класів виконується аналогічно.

Для тестування варіанта використання «Виконати оцінювання тривалості» був використаний метод еквівалентної розбивки. Виділені класи еквівалентності, представлені в таблиці 3.8. Результати тестувань представлені в таблицях 3.9 та 3.10.

Таблиця 3.8 – Класи еквівалентності

№	Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
1	Назва проєкту	1. Введена назва проєкту	Відсутні
2	Трудомісткість	2. Число (позитивне)	3. Від’ємне число 4. Усі крім числового типу (символьні, логічні, графічні зображення). 5. Трудомісткість не було введено

Таблиця 3.9 – Тести із правильних класів еквівалентності

№ тесту	№ класу еквівалентності	Вхідні умови	Вихідні данні	Результат
1	1, 2	В поле «Назва проєкту» введено «Пр122» В поле «Трудомісткість» введено число 1100	Відображено результат розрахунку	Тест пройдено

Таблиця 3.10 – Тести із неправильних класів еквівалентності

№ тесту	Покритий клас	Вхідні дані	Вихідні данні	Результат тестування
2	3	Трудомісткість: -3400	Повідомлення про помилку введення даних	Тест пройдено
3	4	Трудомісткість: М45м	Повідомлення про помилку введення даних	Тест пройдено
4	5	Трудомісткість: -	Повідомлення про помилку введення даних	Тест пройдено

За результатами тестування варіанта використання «Виконати оцінювання тривалості», застосунок працює правильно, не допускаючи до введення будь-яких інших елементів, окрім допустимих символів вводу.

Випробування програмного забезпечення було виконано згідно із програмою і методикою випробувань, що наведена в Додатку Г .

Випробування показали, що розроблене програмне забезпечення повністю відповідає вимогам Технічного завдання (див. Додаток А).

4 РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ ЧАСУ РОЗРОБКИ JAVA-ЗАСТОСУНКІВ ДЛЯ ПЛАТФОРМИ MIDRANGE

В результаті виконання кваліфікаційної роботи отримала подальший розвиток модель ISBSG для оцінювання часу розробки Java-застосунків для платформи midrange в залежності від трудомісткості розробки за рахунок створення моделі тільки для Java-застосунків та побудови рівнянь границь довірчого інтервалу та рівнянь границь інтервалу прогнозування нелінійної регресії, що дозволяє підвищити достовірність оцінювання часу розробки застосунків для платформи midrange.

Практичним результатом кваліфікаційної роботи є програма для оцінювання часу розробки Java-застосунків для платформи midrange, на мові Java, що дозволяє скоротити час відповідного оцінювання, забезпечує зберігання результатів оцінювання, а також надає користувачу швидкий доступ до результатів попередніх оцінювань. 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:

— аутентифікація користувача;
— внесення інформації про проєкт: назву проекту, в трудомісткість проєкту;
— виконання оцінювання тривалості розробки Java-застосунків для платформи midrange: оцінювання 95% довірчого інтервалу середнього значення тривалості розробки Java-застосунків для платформи midrange; - оцінювання 95% інтервалу прогнозування тривалості розробки Java-застосунків для платформи midrange;
— збереження результатів оцінювання тривалості розробки програмного продукту в базу даних;

— перегляд збережених результатів оцінювання тривалості розробки програмних проєктів.

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

Була розроблена необхідна програмна документація:

− технічне завдання (Додаток А);

− тексти програмних модулів (Додаток Д)

− опис програми (Додаток Б);

− інструкція користувача (Додаток В);

− програма та методика випробувань ПЗ (Додаток Г).

Програмне забезпечення було розроблено на мові програмування Java в середовищі Intellij Idea. Під час тестування ПЗ показало повну працездатність.

Робота із програмою здійснюється з використанням зручного користувальницького інтерфейсу й розрахована на роботу в режимі діалогу з користувачем, що не є професійним програмістом.

Як результат розробки, на рисунку 4.1 зображено головне вікно програмного забезпечення веб-додатку "Оцінювання часу розробки Java-застосунків для платформи midrange"

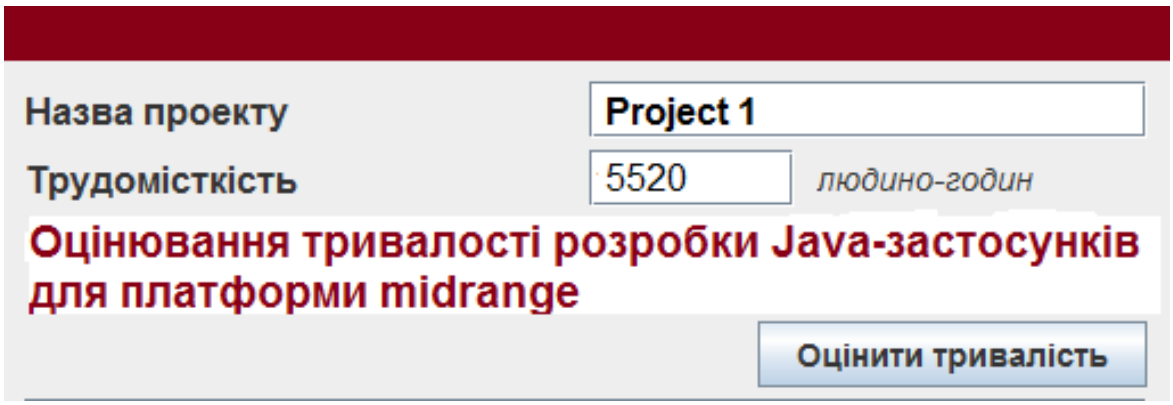

Рисунок 4.1 – Головне вікно програмного забезпечення веб-додатку "Оцінювання часу розробки Java-застосунків для платформи midrange"

На рисунку 4.2 представлено вікно з результатами розрахунку.

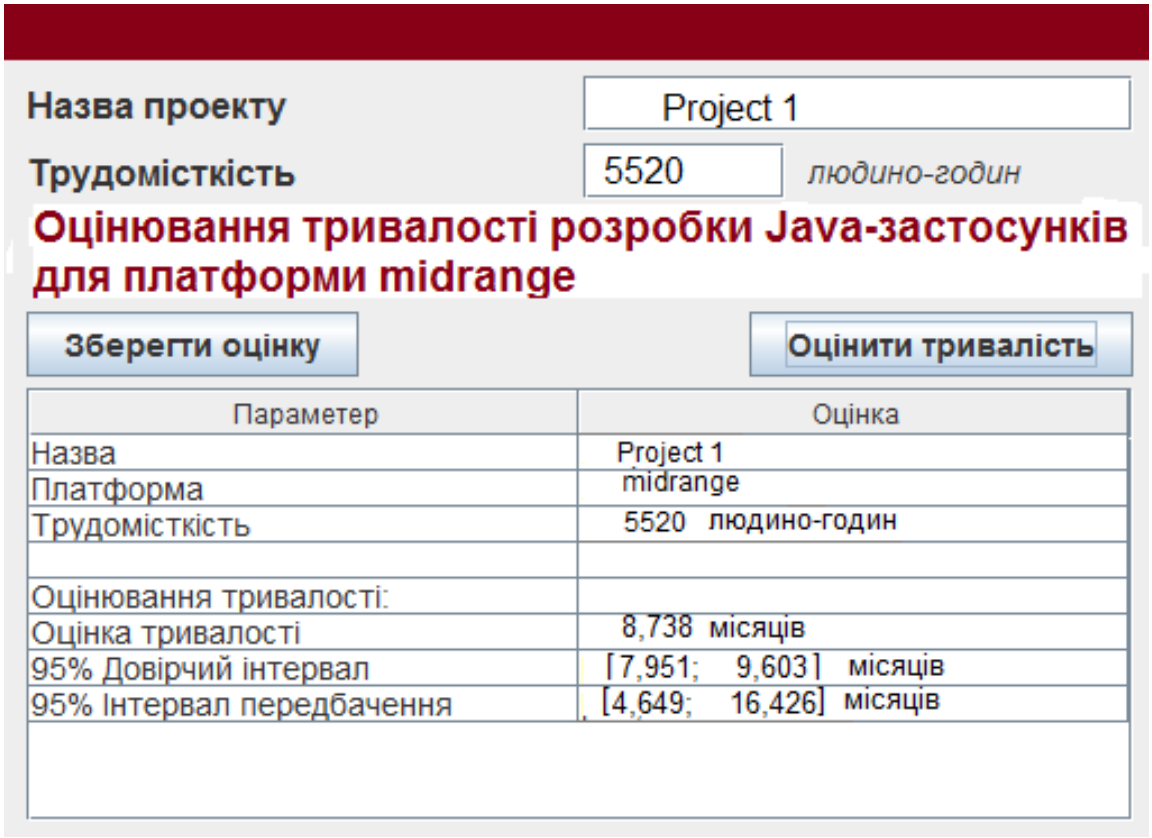

Рисунок 4.2 – Вікно результататів розрахунку програмного забезпечення веб-додатку "Оцінювання часу розробки Java-застосунків для платформи midrange"

Однією з переваг розробки такого програмного продукту є те, що він представлятиме собою веб-додаток. Веб-додатки — невеликі за обсягом, швидко завантажуються і швидко відповідають на дії користувачів. Програмний продукт під управлінням будь-якої операційної системи. Для роботи з продуктом необхідно використовувати сучасний браузер (Mozzilla FireFox, Opera, Google Chrome, Internet Explorer) та мати можливість виходу в мережу Інтернет.

5 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВІД РОЗРОБКИ ТА ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Вступ

Найбільш важливим моментом при проектуванні програмного забезпечення для розробника, з економічної точки зору, є процес формування її майбутньої вартості. Створення програмного забезпечення вимагає одноразових витрат на її розробку, придбання необхідних технічних засобів, а також поточних витрат на функціонування продукту. Економія від функціонування програмного продукту визначається з урахуванням витрат на його експлуатацію. Відношення цієї економії до витрат на створення програмного продукту характеризує економічну ефективність капітальних вкладень.

Економічні показники визначаються по діючим на момент розрахунку оптовим цінам, тарифам і ставкам заробітної плати.

5.1 Розрахунок витрат на створення та експлуатацію програмного забезпечення

Витрати на розробку програмного продукту складаються з витрат на заробітну платню розробника, на збірку (або закупівлю) комп'ютера для програмування, іншої необхідної техніки, витрат на експлуатацію ЕОМ, на засоби розробки та витрати на матеріали.

Розробка програмного забезпечення виконується програмістом, місячний оклад якого складає приблизно 19000 грн [42]. Додаткова заробітна плата складає 20% від окладу.

Виходячи з вищевказаного, основна і додаткова заробітна плата розробника складає 22800 грн./міс, а вартість сучасного ПК - 13200 грн.

(приведена вартість на базі процесору Intel Core i7 9700). При вартості кіловат -години електроенергії рівної 3,93 грн. (згідно діючого тарифу) [43], розраховується вартість розробки програми. Витрати на допоміжні матеріали приведені в таблиці. 5.1.

Таблиця 5.1 - Витрати на допоміжні матеріали

Пункти витрат	Торба (грн.)
Папір А4 (500 арк, упаковка, клас А)	120
Заправлення картріджа до принтера	160
CD –диск	15
Непередбачені витрати	300
Разом матеріали і комплектуючі	595

Вартість програми оцінювання тривалості розробки Java-застосунків для платформи midrange розрахуємо за наступною формулою:

$$Спр = (Ззп + Зсз + Взг + Зе) * Т + Зм, \quad (5.1)$$

де

Т - тривалість розробки (міс);

Ззп - основна і додаткова заробітна плата, грн. :

Зсз - відрахування на соціальні заходи (38% від основної і додаткової заробітної плати, грн.);

Взг - загальногосподарські виграти (10% від основної заробітної плати, грн.);

Зе - витрати на електроенергію;

Зм - витрати на основні і допоміжні матеріали;

Зе при споживанні потужності 0,5 кВт, тривалості роботи за місяць, рівної 21*8=168 рік., вартості кіловат -години електроенергії 3,93 грн. складає:

$$Зе = 168 * 3,93 * 0,5 = 330,12 грн.$$

Всі витрати на розробку ПЗ можна побачити в таблиці 5.2.

Таблиця 5.2 - Витрати на розробку ПЗ.

Найменування витрат	Одиниця	Кількість
Тривалість розробки	міс.	1,5
Основна і додаткова заробітна	грн.	22800
Відрахування на соціальні заходь	грн.	8664
Загальногосподарські витрати	грн.	1900
Витрати на основні і допоміжні матеріали	грн.	595
Витрати на електроенергію	грн.	330,12

Відповідно вартість програми:

$Спр$ = (22800 + 8664 + 1900 + 330,12) * 1,5 + 595 = 51136,18 *грн*.

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

$$Аоб = 13200 * 0,6 = 7920 \text{ грн}.$$

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

$$Вм = 13200 * 0,05 = 660 \text{ грн}.$$

Річний обсяг робіт ПК у годинах визначається в такий спосіб:

$$Фм = 264,5 * Тз$$

де $Тз$ - середнє місячне завантаження устаткування (близько 4 годин);

264,5 – середня кількість робочих днів у році.

Отже, річний обсяг роботи ПК складі:

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

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

$$Зе = 1058*0,5*3,93 = 2078,97 \text{ грн}.$$

Експлуатаційні витрати для ПК за рік складатимуть:

$$Ззр = 7920+660+2078,97 = 10658,97 \text{ грн}.$$

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

$$Зсе = 51136{,}18 + 10658{,}97 = 61795{,}15 \text{ грн}.$$

5.2 Економічна ефективність розробки та впровадження програмного забезпечення оцінювання часу розробки Java-застосунків для платформи midrange

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

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

- високі витрати на складування паперових документів (сейфи, шафи, теки й ін.);
- підвищені витрати на канцтовари;
- витрати, пов' язані з роботою виявлення раніше допущених помилок (людський чинник).

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

- підвищення продуктивності праці;
- вивільнення робочого годині.

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

Обов'язковою умовою визначення економічної ефективності програми є порівнянність усіх показників у часі, за цінами й іншими нормами,

використовуваними для визначення показників, за змістом і кілком елементів витрат.

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

Зарплата 0,6 працівника в рік складає:

$$22800 * 12 * 0{,}6 = 164160{,}00 \text{ грн.}$$

Річний економічний ефект розраховується по формулі:

$$Эгод = \Delta Cn - En * k$$

де ΔCn - вивільнені кошти після впровадження програмного продукту 164160,00 грн.) мінус експлуатаційні витрати (10658,97 грн.);

En - коефіцієнт ефективності (дорівнює коефіцієнту амортизації - 0,6);

k - одноразові витрати на впровадження продукту (*18614,68 грн.).*

$$Эгод = 164160{,}00 - 10658{,}97 - (0{,}6 * 51136{,}18) = 122819{,}32 \text{ грн}.$$

Термін окупності програми розраховується по формулі:

$$T = k/\Delta Cn = 51136{,}18 / 153501{,}03 = 0.33 \text{ (року)} \approx 4 \text{ (місяці)}$$

Отже рядків окупності програмного продукту складає приблизно 4 місяці.

Висновки

У цьому розділі були здійснені розрахунки витрат, економічної ефективності та терміну окупності програми для оцінювання тривалості розробки Java-застосунків для платформи midrange. Проаналізувавши результати можна зробити висновок про економічну доцільність розробки. Впровадження даного програмного забезпечення окупить себе протягом 4 місяців. Таким чином, його розробка є економічно обґрунтованою.

6 ОХОРОНА ПРАЦІ

Під охороною праці розуміється сукупність способів, засобів і дій, спрямованих на скорочення у рамках підприємств або галузей травматизму, ситуацій, які можуть надати шкоду здоров'ю простого робітника.

Законодавство по охороні праці складається з дійсних Законів України «Про охорону праці», «Про охорону здоров'я», «Про пожежну безпеку», «Про забезпечення санітарного та епідеміологічного благополуччя населення», Кодексу законів про працю України та інших нормативних актів.

У випадку, коли міжнародними договорами чи угодами, у яких бере участь Україна, установлені більш високі вимоги до охорони праці, чим ті, які передбачені законодавством України, застосовуються правила міжнародного договору тощо.

Визначення терміну «Охорона праці» наведене в першій статті закону України «Про охорону праці». Охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів і заходів, спрямованих на збереження здоров'я і працездатності людей в процесі праці. Закон України визначає основні напрямки по реалізації конституційного права на охорону життя і здоров'я в процесі трудової діяльності, що регулюються при участі відповідних державних органів, відносини між власником підприємства, установи, організації, органом і роботодавцем з питань зовнішнього середовища і встановлює єдиний порядок організації охорони праці в Україні [44].

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

Цілком безпечних і нешкідливих виробництв не існує. Головна завдання охорони праці – звести до мінімальної ймовірність ураження або захворювання працівника, а також виявлення і вивчення шкідливих факторів, їхній вплив на людину і навколишнє середовище.

Небезпечним виробничим чинником називається такий виробничий чинник, вплив котрого на працюючого у визначених умовах призведе до травми або до іншого раптового, різкого погіршення самопочуття. Прикладами небезпечних факторів можуть служити відкриті струмоведучі частини устаткування, що рухають деталі машин та інше. Прикладами шкідливих факторів являються шкідливі домішки в повітрі, несприятливі метеорологічні умови, шум, вібрації, недостатнє освітлення. [45]

Рівень безпеки є результатом взаємодії людини і того середовища, системи безпеки, що діє на виробництві. До факторів, які впливають на виконання виробничого процесу відносять мікроклімат приміщення, шум та вібрацію, освітлення, електробезпеку, пожежну безпеку, пив та інші.

Шумом є всякий небажаний для людини звук. Шум на виробництві завдає великої шкоди, шкідливо впливаючи на організм людини і знижуючи продуктивність праці. Шум навіть коли невеликий (при рівні 50-60 дБа), створює значне навантаження на нервову систему людини, роблячи на нього психологічний вплив. Під впливом шуму, що перевищує 85-90 дБа, у деру чергу знижується слухова чутливість на високих частотах.

Нормою виробничого шуму є рівень звуку до 85 дБ. Якщо рівень перешкод становить 20 дБ, то такий шум не заважає розбірливості мови. З підвищенням рівня перешкод до 70 дБ та вище мова стає нерозбірливою.

Небезпечним чинником для роботи програміста є також робота за комп'ютером. Найбільшому ризику піддаються зорова, опорно-рухова та нервово-психічна системи. Монітор випускає випромінювання декількох

видів: рентгенівське, ультрафіолетове, інфрачервоне, електромагнітне. Для кожного з цих випромінювань розроблені гранично допустимі норми. Норми передбачають, що опроміненню піддається верхня частина тулуба. Згадані норми встановлені з розрахунку на кожен вид опромінення в окремо, хоча реально всі поля діють одночасно, а їх комплексний вплив досі не досліджено.

Відеоапаратура порушує рівновагу між позитивно і негативно зарядженими іонами в повітрі. Електростатичне поле дисплея притягає негативні іони, порушуючи загальний баланс атмосфери. Це також шкодити здоров'ю. Вже через годину роботи біля монітора спостерігається майже повне зникнення негативних іонів. Вісь чому необхідно, щоб до робочого місця за комп'ютером проникало свіже повітря.

Освітленість робочої зони комп'ютера повинна відповідати стандартам та бути вище яскравості монітора. Відстань від очей до екрана повинна бути не менше ніж півметра. Висота крісла має бути такою, щоб очі булі на одному рівні з центром монітора. Фахівці підтверджують, що саме очі найбільш страждають при роботі з комп'ютером. Занадто відсунутий монітор призводить до перенавантаження м' язів голови та шиї, через що тиск на їх роботові зростає приблизно в три вражай, судини шиї стискаються, погіршуючи кровопостачання до голови. Крім того, людині, що сидить у такій незручній позі, доводитися щоразу відкидати голову назад, для фіксування уваги на ближчих об' єктах. Це посилює вигин шийного відділу хребта. Через деякий годину це може призвести до головної болі та болю у кінцівках, оскільки нерви, що відходять від спинного мозку в області шиї, простягаються до кінчиків пальців.

Малорухома та тривала сидяча поза за комп'ютером шкідлива для опорно-рухового апарату, що веде до застою крові в органах людини. Це особливо проявляється при фізіологічному положенні різних частин тіла і тривало повторюваних одноманітних рухах. Під час роботи за комп'ютером людина сидить кілька годин поспіль в незручному становищі. Це не тільки загрожує втомою і загальним знесиленням організму, а й може призвести до

розвитку остеохондрозу різних ділянок хребта - шийного, грудного, попереково-крижового.

Суттєвий вплив на стан організму ІТ-спеціаліста та його працездатність здійснює мікроклімат (метеорологічні умови), під яким розуміють клімат внутрішнього середовища цих приміщень, що визначається діючою на організм людини сукупністю температури, вологи, руху повітря та теплового випромінювання нагрітих поверхонь. Несприятливі метеорологічні умови приводять до швидкої втоми працівника, частішу його захворюваність та зниження продуктивності праці.

Світло впливає не лише на функції органів зору, а й на діяльність організму в цілому. При поганому освітленні людина швидко втомлюється, працює менш продуктивно, зростає потенційна небезпека помилкових дій і нещасних випадків. Для створення оптимальних умів зорової роботи слід враховувати не лише кількість та якість освітлення, а й кольорове оточення. Так, при світлому пофарбуванні інтер'єру завдяки збільшенню кількості відбитого світла рівень освітленості підвищується на 20-40%, різкість тіней зменшується, покращується рівномірність освітлення.

При надмірній яскравості джерел світла та оточуючих предметів може відбутися засліплення працівника. Нерівномірність освітлення та неоднакова яскравість оточуючих предметів призводять до частої переадаптації очей під година виконання роботи, і як наслідок цього – до швидкого їх втомлення. Тому поверхні, що добрі освітлюються і знаходяться в полі зору, краще фарбувати в кольори середньої світлості.

При освітленні ІТ-відділу використовують природне освітлення, що створюється світлом неба, штучне, здійснюване електричними лампами, і сполучене, при якому у світлий година доби недостатній за нормами, природне освітлення доповнюється штучним. [32]

Для забезпечення здоров'я працюючих приймаються необхідні запобіжні міри захисту, що повністю або частково ізолюють доступ до зони, в

якій діють шкідливі фактори, та виключають їх дію в разі проникнення людини у простір, де сморід виникають. [45]

Велика кількість шуму негативно впливає на розумову роботові мозку, значно знижує продуктивність роботи програміста.

До методів боротьби із шумом відносяться :

- зменшення шуму в джерелі виникнення;
- зміна напрямку випромінювання шуму;
- акустична обробка приміщень;
- установка звукоізолюючих огороджень, кожухи, екрани, кабіни;
- установка глушителів шуму;
- використання засобів індивідуального захисту (вкладиші, навушники, шоломи).

Заходи, щодо зниження негативного впливу мікроклімату:

- впровадження раціональних технологічних процесів;
- механізація та автоматизація виробничих процесів;
- захист працівників різними типами екранів;
- раціональна теплова ізоляція устаткування;
- раціональне розміщення устаткування;
- раціональне планування та конструкторське рішення виробничих приміщень;
- раціональні вентиляція та опалювання.

Для збереження здоров'я очей, та запобіганню перевтоми, робоче приміщення спеціаліста ІТ-відділу повинне відповідати наступним вимогам:

- створювати на робочій поверхні освітленість, що відповідає характеру зорової роботи і не є нижчою за встановлені норми;
- не повинне чинити засліплюючі дії як від самих джерел освітлення, так і від інших предметів, що знаходяться в полі зору;
- забезпечити достатню рівномірність освітлення;
- не створювати на робочій поверхні тіней;
- повинне бути надійним в експлуатації, економічним та естетичним.

Робоче місце має відповідати сучасним вимогам ергономіки і забезпечувати оптимальне розміщення на робочій поверхні використовуваного обладнання (дисплея, клавіатури, принтера) і документів :

1. Висота робочої поверхні робочого столу має регулюватися в межах 680-800 мм, а ширина та глибина забезпечувати можливість виконання операцій у зоні досяжності моторного поля (рекомендовані розміри: 600-1400мм, глибина - 800-1000мм).

2. Робочий стіл повинний мати простір для ніг заввишки не менше ніж 600мм, завширшки не менше ніж 500мм, завглибшки (на рівні колін) не менше ніж 450мм, на рівні простягнутої ноги не менше ніж 650мм. Робочий стілець має бути підйомно-поворотним, регульованим за висотою, з кутом і нахилу сидіння та спинки і за відстанню від спинки до переднього краю сидіння поверхня сидіння має бути плоскою, передній край - заокругленим.

3. Робочі місця слід розташовувати відносно світових прорізів так, щоб природне світло падало переважно з лівого боку.

4. Монітор має розташовуватися на оптимальній відстані від очей користувача, що становить 600-700мм, але не ближче ніж за 600мм з урахуванням розміру літерно-цифрових знаків і символів. Розташування екрана монітору має забезпечувати зручність зорового спостереження у вертикальній площині під кутом +30 градусів до нормальної лінії поглядупрацівника.

5. Клавіатуру слід розташовувати на поверхні столу на відстані 100-300 мм від краю, звернутого до працюючого. У конструкції клавіатури має передбачатися опорний пристрій (виготовлений із матеріалу з високим коефіцієнтом тертя, що перешкоджає мимовільному її зсуву), який дає змогу змінювати кут нахилу поверхні клавіатури у межах 5-15 градусів.

6. Для забезпечення захисту і досягнення нормованих рівнів комп'ютерних випромінювань необхідно застосування приекранних фільтрів, локальних світлофільтрів та інших засобів захисту, що пройшли випробування в акредитованих лабораторіях і мають гігієнічний сертифікат.

На даний час, це поки що найреальніша можливість захистити людину, що працює біля комп'ютера від професійного захворювання.

Електрика широко застосовується у всіх галузях. Тому питанню електробезпечності необхідно приділяти велику увагу. Електробезпечність – система організаційних і технічних заходів та засобів, що забезпечують захист людей від шкідливого і небезпечного впливу електричного струму, електричної дуги, та інше. Проходячи через організм, електричний струм робить термічні, електролітичні і біологічні дії. Це різноманіття дій електричного струму нерідко приводити до різних електричних травм, що умовно можна звести до двох видів: місцеві електричні травми та загальні електричні травми.

Основні поразки струмом наступні:

- випадковий дотик або наближення на небезпечну відстань до струмопровідних частин, що знаходиться під напругою;

- поява напруги на металевих корпусних частинах електроустаткування;

- поява напруги на відключених струмоведучих частинах, на яких працюють люди, унаслідок помилкового включення установки;

Основними заходами захисту від поразки струмом є: забезпечення неприступності струмоведучих частин, що знаходяться під напругою, для випадкового дотику; електричний поділ мережі; усунення небезпеки ураження з появою напруги на корпусах, кожухах і інших частинах електроустаткування, що досягається застосуванням малих напруг, використанням подвійної ізоляції, виявленням потенціалу, захисним заземленням, занулюванням, захисним відключенням і інше; застосування спеціальних електрозахисних засобів - переносних приладів і пристосувань; організація безпечної експлуатації електроустановок.

Занулюванням називається навмисне електричне з' єднання з нулем захисним провідником металевих не струмоведучих частин, що можуть виявитися під напругою. Призначення нульового захисного провідника -

створення струмові короткого замикання ланцюга з малим опором, щоб цей струм був достатнім для швидкого спрацьовування захисту, тобто швидкого відключення ушкодженої установки від мережі.

Захисне відключення – швидкодіючий захист, забезпечує автоматичне відключення при виникненні в ній небезпеки ураження струмом. Така небезпека може виникнути, зокрема, при замиканні фази на корпус електроустаткування; при зниженні опору ізоляції фаз щодо землі нижче визначеної межі; появі в мережі більш високої напруги; дотик людини до струмоведучої частини, що знаходиться під напругою. Установлюють прилад захисного відключення ручний і автоматичний вимикач

Основним нормативним документом, що регламентує вимоги щодо пожежної безпеки є Закон України "Про пожежну безпеку". Цей закон визначає загальні правові, економічні та соціальні основи забезпечення пожежної безпеки на території України, регулює відносини державних органів, юридичних і фізичних осіб у цій галузі незалежно від виду їх діяльності та форм власності. Пожежа – це неконтрольоване горіння поза спеціальним вогнищем, що розповсюджується в часі та просторі, створює загрозу життю і здоров'ю людей, навколишньому середовищу, призводить до матеріальних збитків. Основними причинами пожежі на підприємстві є:

- небезпечне поводження з увігнемо;

- незадовільний стан електротехнічних пристроїв та порушення правив їх монтажу та експлуатації;

- порушення режимів технологічних процесів;

- невиконання вимог нормативних документів з питань пожежної безпеки.

Власник підприємства, повинний:

- розробляти комплексні заходь щодо забезпечення пожежної безпеки;

- організувати навчання працівників правилам пожежної безпеки та пропаганду заходів щодо їх забезпечення;

- мають бути присутні засоби протипожежної безпеки.

Система протипожежного захисту – це сукупність організаційних заходів, а також технічних засобів, спрямованих на запобігання впливу на людей небезпечних факторів пожежі та обмеження матеріальних збитків від неї.

У результаті проведеного дослідження можна зробити наступні висновки. Під час розумової або фізичної праці людина сприймає комплекс шкідливих виробничих чинників, які можуть позитивно або негативно відзначитись на її здоровий' ї та продуктивності праці. Рівень цих факторів залежить від виду робочої діяльності, але можна сказати, що навіть робота в офісі може нашкодити здоров'ю працівника.

Керівництву підприємства необхідно забезпечувати працівників та робітників необхідними засобами індивідуального захисту, вчасно слідкувати за станом робочих місць, що відповідають інструкції з техніки безпеки та не шкодити здоров'ю людини.

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

У приміщені необхідно організувати змішане освітлення, тобто сполучення природного і штучного освітлення. Природне освітлення створюється бічним освітленням через вікно. Штучне освітлення використовується при недостатньому природному освітлені. У даному приміщені використовується загальне штучне освітлення. Розрахунок його здійснюється по методу світлового потоку з урахуванням потоку, відбитого від стін і стелі.

Довжина приміщення (А) = 5 м, ширина (В) = 4 м, висота (Н) = 2,5 м, висота робочої поверхні (*hp*)= 1 м, для освітлення приймаємо світильник типу УПД, мінімальна освітлюваність лампи розжарювання за нормами

Emin=100лк, коефіцієнт відображення стелі ρп =70%, стін ρз = 50%, робочої поверхні ρр=30%. Напруга мережі 210 В.

Розрахуємо відстань від стелі до робочої поверхні:

Но = h - hp = 2.5 - 1 = 1.5 м, (6.1)

де Н - висота приміщення, м;

hp - висота робочої поверхні, м.

Визначимо відстань від стелі до світильника (hc) :

hc= 0.2 * Но = 0.2 * 1.5 = 0.3 м

(6.2)

Висота підвішування світильника над освітлюваною поверхнею (h) :

h = Но - hc = 1.5 - 0.3 = 1.2 м (6.3)

Висота підвішування світильника над підлогою (Hn) :

Hn = h + hp = 1.2 + 1= 2.2 м (6.4)

Для того, щоб досягти найбільшої рівномірності освітлення приймаємо відношення L/h = 1,5. Таким чином відстань між центрами світильників :

L = 1.5 * h = 1.5 * 1.2 = 1.8 м.

Визначимо необхідну кількість світильників:

N = S/l2 = 20 / 1.822 = 6.04 (6.5)

приймаємо за 6 шт. Знаходимо індекс приміщення:

I(h * (A+B)) = 20 / (1.2 * 9) = 1.85.

При *i* = 0,57, коефіцієнтах відображення стелі $\rho_п$ =70%, стін $\rho_з$ = 50%, робочої поверхні $\rho_р$=30% для світильника УПД коефіцієнт використання світлового потоку η = 0,28.

Світловий потік однієї лампи, лм:

$$\Phi = \frac{E_{min} \times S \times Z \times K_з}{N \times \eta \times \gamma},$$ (6.6)

де Emin - рівень мінімальної освітленості за нормами, лк;

S - освітлювана площа приміщення, м2 ;

Кз - коефіцієнт запасу;

Z - коефіцієнт мінімальної освітленості;

N - число світильників;

η - коефіцієнт використання світлового потоку ламп, встановлених у світильнику.

Коефіцієнт запасу Кз враховує експлуатаційне зниження освітленості, в порівнянні із запроектованою, внаслідок забруднення, а також зменшення світлового потоку ламп у процесі їх експлуатації. Для нашого приміщення коефіцієнт мінімальної освітленості (Кз) = 1,3.

Коефіцієнт мінімальної освітленості Z дорівнює відношенню середньої освітленості до мінімальної. Він враховує нерівномірність освітленості і залежить в основному від відношення відстаней між світильниками і від їх типів. Значення цього коефіцієнту для ламп розжарювання приймають Z =1,15.

Ф(6* 0.28) = 1334 лм.

За знайденим світловим потоком вибираємо лампу потужністю 200 Вт, що має світловий потік 3200 лм (Г125-135-200) найбільш близький до розрахункового.

При цьому фактична освітленість (лк) дорівнює:

$$E_{\phi} = E_{\min} \times \frac{\Phi_{л}}{\Phi_{р}}, \qquad (6.7)$$

де *Фл* - світловий потік обраної лампи, лм;

Фр - світловий потік лампи, отриманої розрахунком, лм.

Єф= 75* 3200/1334 = 179,91 лк.

Загальна потужність освітлювальної установки Рз, Вт, визначається за формулою:

$$P_{з} = P_{л} \times N, \qquad (6.8)$$

де Рз - потужність обраної лампи, Вт.

Рз = 200 / 6 = 1200 Вт =1,2 кВт.

Загальна потужність освітлювальної установки становитиме при цьому 1200 Вт.

7 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Вступ

Охорона навколишнього середовища – є регулювання відносин у галузі охорони, використання і відтворення природних ресурсів, забезпечення екологічної безпеки, запобігання і ліквідації негативного впливу господарської та іншої діяльності на навколишнє природне середовище, збереження природних ресурсів, генетичного фонду живої природи, ландшафтів та інших природних комплексів, унікальних територій та природних об'єктів, пов'язаних з історико-культурною спадщиною [46].

Забрудненням навколишнього середовища це дії, які привнесло в екологічну систему не властивих їй живих або неживих компонентів, фізичних або структурних змін, в результаті яких порушуються процеси круговороту і обміну речовин, а також відтоки енергії, унаслідок чого знижується продуктивність або руйнується дана екосистема.

Забруднюючі речовини зазвичай групуються по їх природі:

- фізичні забруднення, до них відносять: шумове забруднення і низькочастотна вібрація, електромагнітне забруднення, радіоактивні елементи;
- хімічні та біологічні забруднювачі, до них відносять: синтетичні органічні речовини, важкі метали, Фтористі з’єднання;
- механічні, до них відносять: пил та тверді частки.

Для вирішення питань в галузі охорони навколишнього середовища займається наука – екологія. Екологія – це економіка природи й одночасне вивчення усіх взаємин живого з органічними і неорганічними компонентами середовища. Основними задачами екології є всебічна діагностика стану природи, розробка прогнозів по зміні стану природного середовища, формування профілактичних заходів щодо охорони навколишнього середовища.

В цьому розділі будуть розглянуті питання щодо забруднення навколишнього середовища інформаційною технікою зокрема і персональними комп’ютерами, та будуть розроблені ефективні методи та заходи стосовно раціональній утилізації відпрацьованого обладнання не завдаючи шкоду природі.

7.1 Забруднення навколишнього середовища

Під забрудненням навколишнього середовища слід розуміти зміну властивостей середовища (хімічних, механічних, фізичних, біологічних і пов'язаних з ними інформаційних), що відбуваються в результаті природних, або штучних процесів і призводять до погіршення функцій середовища по відношенню до будь-якого біологічного, або технологічного об'єкту. Використовуючи різні елементи навколишнього середовища у своїй діяльності, людина змінює її якість. Часто ці зміни виражаються в не сприятливій формі забруднення.

Сьогоднішній день а, тим більше завтрашній, важко представити без комп'ютерів, телевізорів, та іншої електронної техніки.

Інформаційні та телекомунікаційні технології, включивши в себе екологію в якості гуманних підвалин розвитку, перетворились на ідею інформаційного суспільства, стали способом життя людства, запорукою нового циклу розвитку цивілізації та планети.

Інформаційні технології сьогодні є екологічнішими за більшість інших видів активної людської діяльності, проте їх ще не можна назвати справді екологічними. Скажімо, ефективність інформаційних мереж напряму залежить від кількості користувачів, тобто, від кількості комп’ютерів, включених до мережі. Але для виготовлення одного звичайного персонального комп’ютера потрібно від 15 до 19 тонн матеріалів. Це порівнювано з 25 тоннами, потрібними для виготовлення автомобіля. На кожен функціонуючий комп’ютер (використовуваний в середньому протягом

4 років) припадає 1,5 комп’ютери вироблених. А близько третини комп’ютерів ніколи не буває продано взагалі – через швидкість, з якою вони втрачають технологічну актуальність. Це означає, що затрачувані ресурси справді наближаються до рівня автомобіля.

Електронні пристрої містять дуже токсичні з’єднання, які, потрапляючи в навколишнє середовище, створюють серйозну небезпеку для життя людей. Так, наприклад, 22% ртуті, що видобувається щороку в усьому світі, йде саме на потреби електронної промисловості і, зокрема, міститься в мобільних телефонах. Кадмій, який є канцерогеном, використовується практично в усіх напівпровідникових пристроях. Свинець, особливо токсичний для нервової системи, міститься в акумуляторах і екранах моніторів. У міру розкладання захисних покриттів з електронних пристроїв у навколишнє середовище виділяється діоксин та інші високотоксичні з’єднання.

Стурбованість громадськості проблемами екології, а також нові, більш жорсткі закони по захисту навколишнього середовища змушують великих виробників устаткування створювати мережі по збору, що вийшли з обігу техніку і заводи з її утилізації. Крім того, в конструкції обладнання максимально збільшується частка матеріалів, придатних для переробки. Розміри мережі з утилізації "електронного брухту" залежать від регіону і місцевого законодавства.

Вся оргтехніка включає в свій склад як органічні складові (пластик різних видів, матеріали на основі полівінілхлориду, фенолформальдегіда), так і майже повний набір металів.

Згідно з довідкових даних і на підставі лабораторних хімічних аналізів в таблиці 7.1 наведено усереднені дані про вміст різних металів і матеріалів у персональному комп’ютері.

Отже, звичайний комп'ютер містить як цінні метали, такі як золото, срібло, алюміній, мідь, так і небезпечні, такі як кадмій, свинець, цинк, нікель, а тому при списанні та утилізації обладнання керівнику необхідно керуватися і законодавством в області охорони навколишнього середовища.

7.2 Заходи щодо запобігання забруднення навколишнього середовища

Персональні комп'ютери, ноутбуки та інша інформаційна техніка, як відомо широко використовується в галузі наукових досліджень, промисловості, а також у повсякденному домашньому користуванні. Але будь-яка техніка стрімко застаріває, їй на зміну приходять нові, більш потужні, більш сучасні ПК та оргтехніка. Поступово виникає проблема, що робити зі старою технікою, морально застарілою або з тих чи інших причин, що вийшла з ладу, яка захаращує підсобні приміщення та склади.

Таблиця 7.1 – Шкідливі речовини які містяться в ПК

Найменування	Дорогоцінні метали		Кольорові і чорні метали			Полімери і скло	
	Au,	Ag, г	Al, г	Cu, г	Fe, г	Пластик, кг	Скло, кг
Персональний комп’ютер (монітор, системний блок, клавіатура, маніпулятор)	0,053-0,072	0,8-1,1	0,1-0,4	0,1-0,2	3-4	3-3,5	10-20

Утилізація оргтехніки та комп'ютерів це процес, який проводиться в кілька етапів. Найперше дію це списання обладнання безпосередньо з підприємства. Етап другий це розбір техніки і сортування отриманих матеріалів. Якщо деталі здатні служити вихідною сировиною, наприклад, кінескоп, деталі, в складі яких є дорогоцінні метали, то їх відправляють на очищення.

Як нам відомо до складу комп'ютера входить безліч металів таких як золото, срібло, алюміній, мідь та інших. Ще етапом по утилізації персональних комп'ютерів можна досягнути завдяки вторинній переробці.

Сутність даного процесу полягає в тому, що можна витягти з цієї сировини частку корисних та рідких матеріалів таких як іридію, міді та інших. Цей процес відтворити набагато легше аніж наприклад видобути тонну міді, яка міститься в тисячотонних гірських породах.

Одне з нововведень для утилізації друкованих плат придумали Співробітники з Національної фізичної лабораторії Великобританії, продемонстрували можливість спеціального розчину який розчинюють у гарячій воді. Дія якого зумовлює відшарування електронних компонентів.

Таким чином 90% компонентів нових друкованих плат можна використовувати знову, тоді як у випадку звичайним методам – тільки 2%.

Практично жодне підприємство не зможе самостійно утилізувати комп'ютери та оргтехніку, так як цей процес вимагає сучасного обладнання та специфічних знань. Тому довірити таку роботу можна тільки професіоналам, які мають великий досвід у даній сфері.

Проблема утилізації використаних комп’ютерів, периферійного обладнання, стає гострішою з кожним роком. Обсяги виробництва продуктів інформаційно-телекомунікаційних технологій та частота їх заміни на нові моделі примушують компанії замислюватись над проблемою біодеградації. Успіхи в цій галузі допоможуть, серед іншого, компаніям-виробникам зменшити податки, котрі вони сплачують зараз за утилізацію застарілих моделей. Останнє тим більше важливо, оскільки робить екологізацію економічно вигідною, тож спрямовує у цю сферу дедалі більше зусиль дослідників та довгострокових капіталовкладень. Таким чином, подальше поширення інформаційних технологій не збільшить, а навпаки – зменшить техногенне навантаження на довкілля.

В кінцевому результаті виконаної роботи можна стверджувати, що вдосконалення сучасних інформаційних технологій, слід направляти не тільки для того щоб створювати людині максимально комфортні умови життя теперішнього часу. Але і для досягнення безвідходного процесу утилізації відпрацьованої техніки, не завдаючи шкоду навколишньому середовищу.

Висновки

В спеціальному розділі з охорони навколишнього середовища було розглянуто питання щодо забруднення навколишнього середовища інформаційною технікою зокрема і персональними комп’ютерами, та запропоновані методи та заходи стосовно раціональної утилізації відпрацьованого обладнання не завдаючи шкоду природі.

ВИСНОВКИ

В кваліфікаційній роботі було досягнуто поставленої мети: підвищення достовірності оцінювання часу розробки Java-застосунків для платформи midrange за рахунок побудови нелінійної регресійної моделі.

В процесі проведення досліджень одержані наступні результати:

1. На основі аналізу існуючих методів та моделей оцінювання часу розробки програмних проєктів виявлені їх особливості, переваги та недоліки; визначено задачі, які необхідно вирішити для досягнення мети дослідження, а саме – підвищення достовірності оцінювання часу розробки Java-застосунків для платформи midrange. Аналіз показав необхідність побудови нелінійної регресійні моделі та використання нормалізуючих перетворень для емпіричних даних (трудомісткості та тривалості виконання програмних проєктів).

2. Розроблені нелінійні регресійні моделі тривалості розробки Java-застосунків для платформи midrange в залежності від трудомісткості (з урахування типів ПЗ), шляхом побудови рівняння нелінійної регресії, рівняння границь довірчого інтервалу та рівняння границь інтервалу прогнозування нелінійної регресії для тривалості розробки Java-застосунків для платформи. Це дозволило, у порівнянні з моделлю ISBSG, підвищити достовірність оцінювання (*PRED*(0,25)) часу розробки Java-застосунків для платформи midrange на 27%, зменшити відносну похибку (*MMRE*) на 57% .

3. Однак прийнятні значення *MMR*E та *PRED*(0,25) (не більше 0,25 та не менше 0,75 відповідно) для нелінійної регресії з використанням нормалізуючого перетворення у вигляді десяткового логарифму не досягнуті, що свідчить про необхідність застосування у майбутньому більш складних перетворень, наприклад, нормалізуючого перетворення Джонсона. Також необхідно в майбутньому дослідити вплив додаткових чинників, таких як

сфера використання програмного продукту (банківська справа, виробництво, телекомунікації тощо), функціональний розмір та ін.

4. Розроблене програмне забезпечення для оцінювання часу розробки Java-застосунків для платформи midrange дозволить скоротити час відповідного оцінювання, забезпечить зберігання результатів оцінювання, а також надасть швидкий доступ до результатів попередніх оцінювань.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Лавріщева, К.М. Програмна інженерія. Київ: Академперіодика, 2008. 319 с.

2 Standish Group 2015 Chaos Report - Q&A with Jennifer Lynch // S. Hastie, S. Wojewoda. URL: https://www.infoq.com/articles/standish-chaos-2015 (дата звернення: 15.10.2021).

3 Chow A. W. Engaging a corporate community to manage technology and embrace innovation. IBM Systems Journal. 2007. Vol. 46, No. 4. P. 639 – 650.

4 S. McConnell. Rapid Development: Taming Wild Software Schedules. Redmond, WA: Microsoft Press, 2013. 652 p.

5 A guide to the project management body of knowledge. Fifth edition. Project Management Institute, 2013. 589 p.

6 Chapter 12: Software Engineering Economics. URL: http://swebokwiki.org/ Chapter_12:_Software_Engineering_Economics (дата звернення: 15.10.2021)

7 Приходько С. Б., Пухалевич А. В. Розробка нелінійної регресійної моделі тривалості програмних проектів на основі нормалізуючого перетворення Джонсона. Радіоелектронні і комп'ютерні системи. Харків: Харківський авіаційний інститут, 2012. № 4 (56). С. 90–93. ISSN 1814-4225.

8 Приходько С. Б., Пухалевич А. В. Розробка нелінійних регресійних моделей тривалості програмних проектів на основі перетворення Джонсона. Збірник наукових праць НУК. Миколаїв: НУК, 2014. № 2 (2014). С. 76–80.

9 Oligny S. Bourque P., Abran A. An empirical assessment of project duration models in software engineering. 8th.European Software Control and Metrics Conference ESCOM. In Proc. Berlin, 1997.

10 Oligny S., Bourque P., Abran A., Fournier B. Exploring the relation between effort and duration in software engineering projects. In proc. of the World Computer Congress, 2000. P. 175–178.

11 Puentes M., Méndez M., Bohorquez L. Romero S. Estimation metrics in software projects URL: https://www.researchgate.net/publication/329471092_ Estimation_metrics_in_softssof_projects (дата звернення 15.10.2021)

12 ChowdaryV., Vamsi K. Software Effort Estimation: A Comparative Analysis URL: https://ijpsat.ijsht-journals.org/index.php/ijpsat/article/view/27/17 (дата звернення 12.10.2021)

13 Jørgensen M. A Review of studies on expert estimation of software development effort .The Journal of Systems and Software. 2004. Vol. 70, No. 1-2,

14 Bittner K. Spence I. Managing Iterative Software Development. Addison-Wesley Professional, 2006. 640 p.

15 Кульдин С. П. Генетический подход к проблеме оценки сроков и трудоемкости разработки программного обеспечения с заданными требованиями к качеству. Прикладная информатика. №5, 2010. С. 30-42.

16 Putnam L. H. A general empirical solution to the macrosoftware sizing and estimating problem. IEEE Transactions on Sofiware Engineering. Vol. 4, No. 2, July 1978. Pp. 345–361.

17 Lee D. Norden-Raleigh Analysis: A Useful Tool for EVM in Development Projects. The Measurable News, March, 2002. P. 21–24.

18 Ahonen J., Savolainen P., Merikoski H., Nevalainen J. Reported project management effort, project size, and contract type. Journal of Systems and Software. Vol. 109, 2015. P. 205–213.

19 McConnell S. 7 Theses on Software Estimation. URL: https://stevemcconnell.com/17-theses-software-estimation (дата звернення 10.10.2021).

20 Galorath D. D., Evans M. W. Software sizing, estimation, and risk management: when performance is measured performance improves. CRC Press, 2019. 576 p.

21 Gingnell, L.; Franke, U.; Lagerström, R.; Ericsson, E.; Lilliesköld, J. Quantifying success factors for IT projects. Inf. Syst. Manag, 2014, Vol. 31, P. 21–36.

22 Zarour, A., Zein, S. Software development estimation techniques in industrial contexts: An exploratory multiple case-study. International Journal of Technology in Education and Science (IJTES), 2019 3(2), 72-84.

23 The IFPUG Guide to IT and software measurement. CRC Press, 2012. 848p.

24 Suelmann, H. Putnam's effort-duration trade-off law: is the software estimation problem really solved. Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement. Rotterdam, 2014. P. 79–84.

25 Приходько, С. Б., Пухалевич А. В. Інтервальне оцінювання математичного сподівання часу затримок виконання програмних проектів на основі перетворення Джонсона. Вестник ХНТУ. Херсон: ХНТУ, 2010. №2 (38). С. 402–404.

26 Приходько, С. Б., Пухалевич А. В. Розробка нелінійної регресійної моделі тривалості програмних проектів на основі нормалізуючого перетворення Джонсона. Радіоелектронні і комп’ютерні системи. – Харків: Харківський авіаційний інститут, 2012. № 4 (56). С. 90–93.

27 Приходько, С. Б., Пухалевич А. В. Розробка нелінійних регресійних моделей тривалості програмних проектів на основі перетворення Джонсона. Збірник наукових праць НУК. Миколаїв : НУК. 2014. № 2 (2014). С. 76–80.

28 Приходько, С. Б., Пухалевич А. В. Confidence interval estimation of PC software project duration regression based on Johnson transformation Радіоелектронні і комп’ютерні системи. Харків: Харківський авіаційний інститут, 2014. № 2 (66). С. 104–107.

29 Приходько С. Б., Фаріонова Т.А. Нелінійні регресійні рівняння для оцінювання тривалості розробки Java-застосунків для платформи midrange. Інформаційні технології: моделі, алгоритми, системи (ITMAS – 2021): матеріали II Всеукраїнської науково-практичної інтернет конференції. Миколаїв: НУК імені адмірала Макарова, 2021. С. 20 – 22.

30 Оценка программного обеспечения. URL: https://ce-na.ru/articles/o-nematerialnykh-aktivakh/ocenka_programmnogo_obespecheniya (дата звернення: 15.11.2021).

31 Дударин П.В., Тронин В.Г., Святов К.В., Белов В.А., Шакуров Р.А. Подход к оценке трудоемкости задач в процессе разработки программного обеспечения на основе нейронных сетей. Автоматизация процессов управления, 2019. № 3 (57), С. 65 – 72.

32 Сидоров Н.А., Баценко Д.В., Василенко Ю.Н., Щебетин Ю.В. Модели, методы и средства оценки стоимости программного обеспечения. Методи та засоби програмної інженерії. С. 290–298. URL: http://dspace.nbuv.gov.ua/bitstream/handle/123456789/1541/36-Sidorov.pdf (дата звернення: 17.11.2021).

33 Azzeh, M., Marwan A. Analogy-Based Effort Estimation: A New Method to Discover Set of Analogies from Dataset Characteristics. URL: https://arxiv.org/ftp/arxiv/papers/1703/1703.04564.pdf (дата звернення: 17.10.2021)

34 Parthasarathi P. A new software cost estimation model for small software organizations: An empirical approach. International Journal of Applied Engineering Research, 2015, 10(15). P. 76 – 82.

35 Midrange computer. URL: https://en.wikipedia.org/wiki/Midrange_computer (дата звернення: 17.11.2021).

36 Пухалевич, А. В. Моделі та інформаційна технологія переробки інформації для оцінювання тривалості проектів з розробки програмного забезпечення: дис. канд. техн. наук: 05.13.06 / НУК ім. адм. Макарова. Миколаїв, 2017. 179 с.

37 Приходько С. Б., Макарова Л. М., Приходько Н. В., Пухалевич А. В. Методичні вказівки та завдання до виконання лабораторних робіт з дисципліни «Емпіричні методи програмної інженерії». Миколаїв: НУК, 2020. 60 с.

38 Ларман К. Применение UML и шаблонов проектирования: пер. с англ. М.: Издательский дом «Вильяме», 2013. 736 с.

39 Верлань А.Ф., Чмырь И.А., Кузниченко С.Д., Коваленко Л.Б. Императивное программирование и объектно-ориентированное моделирование: JAVA, UML, OSL. Одесса: Экология. 2013. 432 с.

40 Якобсон А., Буч Г., Рамбо Дж. Унифицированный процесс разработки програм. URL: https://krupoderovakr.files.wordpress.com/2014/02/d0b0-d18fd0bad0bed0b1d181d0bed0bd-d0b3-d0b1d183d187-d0b4d0b6-d180d0b0d0bcd0b1d0be-d183d0bdd0b8d184d0b8d186d0b8d180d0bed0b2d0b0d0bd.pdf (дата звернення 15.11.2021 р.).

41 Кадомський К.К., Ніколюк П.К. Java. Теорія і практика: навчальний посібник. Вінниця: Донну, 2019. 197 с.

42 Куликов С.С. Тестирование программного обеспечения. Базовый курс: практ. пособие. 2-е изд.. Минск: Четыре четверти, 2018. 120 с.

43 Середня зарплата за категорією «IT, комп'ютери, інтернет» в Україні. URL: https://www.work.ua/salary-web-%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%81%D1%82/?setlp=ua) (дата звернення: 27.11.2021).

44 Торгова електрична компанія. Ціни. URL: https://tek.energy/electricity/prices (дата звернення: 27.11.2021).

45 Про охорону праці: Закон України від 14.10.1992р № 2694-XII. (зі змінами від 15.07.2021 р.). URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text (дата звернення: 25.11.2021).

46 Про охорону навколишнього природного середовища: Закон України. Відомості Верховної Ради України (ВВР), 1991. № 41. С. 546 (зі змінами від 15.07.2021 р.). URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text (дата звернення: 25.11.2021).

ДОДАТОК А.
ТЕХНІЧНЕ ЗАВДАННЯ

Вступ

Повне найменування теми розробки: програмне забезпечення для оцінювання часу розробки Java-застосунків для платформи midrange.

Коротке найменування програми: Software DEJM.

Коротка характеристика галузі застосування: програмне забезпечення застосовується для автоматизації розрахунків з оцінювання часу розробки Java-застосунків для платформи midrange, що дозволить підвищити ефективність роботи аналітика та менеджера проєкту на початкових стадіях створення програмного забезпечення.

1. Підстави для розробки

Розробка виконується на підставі завдання на магістерську кваліфікаційну роботу, що видане кафедрою ПЗАС та затверджене наказом ректора НУК № 1239-уч від 13.10.2021 року.

2. Призначення розробки

2.1 Функціональне призначення програми

Даний програмний продукт призначений для автоматизації розрахунків оцінювання часу розробки Java-застосунків для платформи midrange.

2.2 Експлуатаційне призначення програми

Експлутаційне призначенням є спрощення роботи кінцевих користувачів, а саме аналітиків та менеджерів проєктів, та підвищення точності оцінювання часу розробки Java-застосунків для платформи midrange на початкових стадіях розробки програмного забезпечення.

Програма призначена для некомерційного використання.

3. Вимоги до програми або програмного продукту

3.1. Вимоги до функціональних характеристик

3.1.1. Вимоги до складу виконуваних функцій

Програмне забезпечення повинно виконувати наступні функції:

— аутентифікація користувача

— внесення інформації про проєкт:

  - внесення інформації про назву проекту;
  - внесення інформації про трудомісткість проєкту

— виконання оцінювання тривалості розробки Java-застосунків для платформи midrange:

  - оцінювання 95% довірчого інтервалу середнього значення тривалості розробки Java-застосунків для платформи midrange;
  - - оцінювання 95% інтервалу прогнозування тривалості розробки Java-застосунків для платформи midrange;

— збереження результатів оцінювання тривалості розробки програмного продукту в базу даних;

— перегляд збережених результатів оцінювання тривалості розробки програмних проєктів.

3.1.2. Вимоги до організації вхідних даних

Внесення вхідних даних виконується шляхом їх введення у відповідні поля веб-форм користувачем.

Вихідні дані (результати оцінювання тривалості, довірчий інтервалу та інтервал передбачення) повинні виводитися в таблиці на формі веб-додатку та зберігатися у базі даних.

3.1.3. Вимоги до організації вихідних даних

Опишемо структури даних, які використовуються у програмі:

- інформація про програмний проєкт: назва проєкту (текстове поле), трудомісткість, людино-години (числове поле);

- інформація про користувача: логін, пароль (текстове поле).

3.1.4. Вимоги до часових характеристик

Вимоги до часових характеристик не не висуваються.

3.2 Вимоги до надійності

3.2.1 Вимоги до забезпечення надійного (стійкого) функціонування програми

Надійне (стійке) функціонування програми повинне бути забезпечене виконанням сукупності організаційно-технічних заходів, перелік яких наведений нижче:

1) видавати повідомлення про помилки при невірно введених даних;

2) вчасне живлення використовуваних технічних засобів;

3) стабільний інтернет-зв’язок.

Час, що витрачається на обслуговування програмного продукту не повинен перевищувати 2% від загального часу роботи.

3.2.2 Час відновлення після відмови

Час відновлення після відмови, викликаної несправністю технічних засобів, фатальним збоєм (крахом) операційної системи, не повинне перевищувати часу, необхідного на усунення несправностей технічних засобів та переустановлення програмних засобів.

3.2.3 Відмови через некоректні дії користувача

Для запобігання відмови програми, внаслідок некоректних дій користувача, програма повинна виконувати перевірку введених даних. У випадку, коли ведені дані не відповідають вимогам (порожні або містять неприпустимі символи), програма повинна вивести повідомлення про помилку.

3.3 Умови експлуатації

3.3.1 Кліматичні умови експлуатації

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

3.3.2 Вимоги до видів обслуговування

Програма не вимагає проведення яких-небудь видів обслуговування.

3.3.3 Вимоги до чисельності та кваліфікації персоналу

Для роботи з програмним продуктом необхідна 1 людина - кінцевий користувач (аналітик, менеджер проєкту).

3.4 Вимоги до складу і параметрів технічних засобів

Для реалізації веб-додатку було обрано такі компоненти:

Операційна система, в якій встановлена версія Java Runtime Environment (JRE) не нижче версії Java 1.8.

- Intellij Idea – в якості середовища розробки;
- Java – в якості мови програмування;

Даний програмний продукт вимагає технічний засіб – персональний комп’ютер, ноутбук, планшет чи мобільний телефон з можливістю виходу в мережу інтернет.

3.5 Вимоги до інформаційної та програмної сумісності

Розроблений програмний продукт орієнтований на роботу в веб-браузерах. Тому для коректної роботи програми необхідне стабільне функціонування операційної системи та підключеня до інтернету.

3.6 Вимоги до маркування та пакування

Поширення даного програмного продукту на фізичних носіях не передбачається.

3.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4 Вимоги до програмної документації

Програмна документація повинна містити:

- технічне завдання;
- інструкцію користувача;
- опис програмного забезпечення;

- тексти програмних модулів;
- програму та методику випробувань.

5 Техніко-економічні показники

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

6 Стадії та етапи розробки

Стадії та етапи розробки представлені в таблиці А.1.

7 Порядок прийому та контролю

Прийомка та контроль програми повинні проводитися відповідно до узгоджених заздалегідь із замовником програми і методики випробувань.

Кожна стадія розробки повинна бути представлена в зазначені терміни і узгоджена з замовником.

Хід проведення приймально-здавальних випробувань проводиться відповідно до програми і методики випробувань і документують за допомогою протоколу проведення випробувань.

На підставі протоколу проведення випробувань виконавець спільно з замовником підписують акт приймання-здачі програмного забезпечення в експлуатацію.

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.

Таблиця А.1 – Стадії та етапи розробки

№	Стадії	Етапи розробки	Термін	Зміст робіт
1	2	3	4	5
1	Технічне завдання	1.1Обґрунтуваня необхідності розробки програми	Початок: 05.09.21 Кінець: 11.10.21	Постановка завдання. Збір вихідних матеріалів. Вибір і обґрунтування критеріїв ефективності та якості програми.
		1.2 Науково-дослідні роботи		Визначення структури вхідних та вихідних даних. Аналіз існуючих методів та моделей оцінювання тривалості розробки програмних проєктів. Розробка моделі для оцінювання часу розробки Java-застосунків для платформи midrange. Обґрунтування доцільності розробки. Визначення вимог до технічних засобів. Обґрунтування можливості вирішення завдання.
		1.3 Розробка та затвердження технічного завдання		Визначення вимог до програми. Розробка техніко-економічного обґрунтування розробки програми. Визначення стадій, етапів і термінів розробки програми і документації на неї. Вибір мов програмування. Визначення необхідності проведення науково-дослідних робіт на наступних стадіях. Узгодження і затвердження технічного завдання.

Кінець табл. А.1

1	2	3	4	5
2	Ескізний проект	2.1 Розробка ескізного проєкту	Початок: 12.10.21 Кінець: 20.10.21	Попередня розробка структури вхідних і вихідних даних, уточнення методів рішення завдання, розробка загального опису алгоритму рішення задачі.
		2.2 Затвердження ескізного проєкту		Узгодження і затвердження ескізного проекту
3	Технічний проект	3.1 Розробка технічного проєекту	Початок: 21.10.21 Кінець: 01.11.21	Уточнення структури вхідних і вихідних даних. Розробка алгоритму рішення завдання. Визначення форми представлення вхідних і вихідних даних, визначення семантики і синтаксису мови даних. Розробка структури програми, остаточне визначення конфігурації технічних засобів.
		3.2 Затвердження технічного проєкту		Розробка плану заходів з розробки та впровадження програми, розробка пояснювальної записки, узгодження та затвердження технічного проєкту.
4	Робочий проект	4.1 Розробка програми	Початок: 02.11.21 Кінець: 19.11.21	Програмування та налагодження програми.
		4.2 Розробка програмної документації		Розробка програмних документів
		4.3 Випробування програми		Розробка, погодження та затвердження програми і методики випробувань, коректування програми і програмної документації.
5	Впровадження	Підготовка і передача програми	Початок: 20.11.21 Кінець: 29.11.21	Підготовка і передача програми та програмної документації для супроводу.

ДОДАТОК Б.
ОПИС ПРОГРАМИ ПРОГРАМИ «ОЦІНЮВАННЯ ЧАСУ РОЗРОБКИ JAVA-ЗАСТОСУНКІВ ДЛЯ ПЛАТФОРМИ MIDRANGE»

1 Загальні відомості

1.1 Позначення танайменування програми

Найменування: програмне забезпечення для оцінювання часу розробки Java-застосунків для платформи midrange.

Позначення: Software DEJM.

1.2 Програмне забезпечення, необхідне для функціонування програми

Програмне забезпечення (веб-додаток) для оцінювання часу розробки java-застосунків для платформи midrange є крос-платформним. Основною необхідною вимогою для функціонування є наявність браузеру та доступ в мережу інтернет.

1.3 Мови програмування

Програмне забезпечення для оцінювання часу розробки Java-застосунків для платформи midrange реалізовано на мові високого рівня Java.

2 Функціональне призначення

2.1 Класи вирішуваних завдань і призначення програми

Програмне забезпечення повинно виконувати наступні функції:

— аутентифікація користувача
— внесення інформації про проєкт: назва проєкту, трудомісткість проєкту;
— виконання оцінювання тривалості розробки Java-застосунків для платформи midrange:
  - оцінювання 95% довірчого інтервалу середнього значення тривалості розробки Java-застосунків для платформи midrange;
  - - оцінювання 95% інтервалу прогнозування тривалості розробки Java-застосунків для платформи midrange;
— збереження результатів в базу даних;

— перегляд збережених результатів оцінювання тривалості розробки програмних проєктів.

2.2 Функціональні обмеження

Відсутні

3. Опис логічної структури

Програмне забезпечення розроблене із використанням шаблону проектування MVC (Model-View-Controller). У рамках цього шаблону проектування програма поділяється на три окремі, але взаємопов'язані частини з розподілом функцій між компонентами – модель, представлення та контроллер.

4. Технічні та програмні засоби

До складу технічних засобів висуваються вимоги не нижче наступних:

- операційна система, в якій встановлена версія Java Runtime Environment (JRE) не нижче версії Java 1.8;

- персональний комп’ютер, ноутбук, планшет чи мобільний телефон з можливістю виходу в мережу інтернет. Швидкість роботи веб-додатку на конкретному комп'ютері залежить також від характеристик окремих його комплектуючих (процесора, оперативної пам'яті і ін.) та швидкості підключення до інтеренету.

5 Виклик і завантаження

Запуск ПЗ веб-додатку в графічному режимі здійснюється через браузер.

6 Вхідні і вихідні дані

Вхідні дані – назва проєкту, трудомісткість розробки (оюлдино-год.).

Вихідні дані:

- оцінювання середнього значення тривалості проекту з розробки програмного забезпечення з використанням нелінійної регресійної моделі в залежності від трудомісткості;

- оцінювання 95% довірчого інтервалу середнього значення тривалості проекту з розробки програмного забезпечення та інтервалу прогнозування тривалості проекту з розробки програмного забезпечення.

ДОДАТОК В
ІНСТРУКЦІЯ КОРИСТУВАЧА КОМП'ЮТЕРНОЇ ПРОГРАМИ «ОЦІНЮВАННЯ ЧАСУ РОЗРОБКИ JAVA-ЗАСТОСУНКІВ ДЛЯ ПЛАТФОРМИ MIDRANGE»

Вступ

Ця настанова призначена для ознайомлення користувача з технічними характеристиками і функціональними можливостями комп'ютерної програми «Оцінювання часу розробки Java-застосунків для платформи midrange».

Програма призначена для точкового та інтервального оцінювання тривалості програмних проектів для платформи midrange в залежності від трудомісткості. Програма є вигляді веб-застосунком та має графічний інтерфейс, який дозволяє виконати автентифікацію користувача, введення даних, перегляд результатів оцінювання тривалості програмних проектів та збереження отриманих результатів в базу даних.

2 Загальні відомості про програму

2.1 Позначення і найменування програми

Найменування програми - «Оцінювання часу розробки Java-застосунків для платформи midrange». Позначення програми - " Software DEJM ".

2.2 Мови програмування, на яких написана програма

Програма написана на мові програмування Java.

2.3 Призначення програми

Даний програмний продукт призначений для автоматизації розрахунків оцінювання часу розробки Java-застосунків для платформи midrange.

Експлутаційне призначенням є спрощення роботи кінцевих користувачів, а саме аналітиків та менеджерів проєктів, та підвищення точності оцінювання часу розробки Java-застосунків для платформи midrange на початкових стадіях розробки програмного забезпечення. Програма призначена для некомерційного використання.

2.4 Можливості програми

Програма виконує наступні функції:

- - аутентифікація користувача
- внесення інформації про проєкт:
  - внесення інформації про назву проекту;
  - внесення інформації про трудомісткість проєкту
- виконання оцінювання тривалості розробки Java-застосунків для платформи midrange:
  - оцінювання 95% довірчого інтервалу середнього значення тривалості розробки Java-застосунків для платформи midrange;
  - оцінювання 95% інтервалу прогнозування тривалості розробки Java-застосунків для платформи midrange;
- збереження результатів оцінювання тривалості розробки програмного продукту в базу даних;
- перегляд збережених результатів оцінювання тривалості розробки програмних проєктів.

2.5 Обмеження області застосування програми

-Область застосування програми обмежена сферою розробки програмного забезпечення.

3 Умови застосування програми

3.1 Умови, необхідні для виконання програми

Спеціальні умови для виконання програми відсутні.

3.2 Вимоги до технічних засобів, необхідних для функціонування програми

Рекомендовані вимоги:

— центральний процесор - Intel або AMD від 2.0 GHz і вище;

— жорсткий диск – об’єм не менший 40 Гб;

— оперативна пам’ять – об’єм не менший 2 Гб;

— мережевий адаптер або інші засоби доступу до мережі Internet.

Даний програмний продукт вимагає технічний засіб – персональний комп’ютер, ноутбук, планшет чи мобільний телефон з можливістю виходу в мережу інтернет.

3.3 Програмне забезпечення, необхідне для функціонування програми

Операційна система, в якій встановлена версія Java Runtime Environment (JRE) не нижче версії Java 1.8. Розроблений програмний продукт орієнтований на роботу в веб-браузерах. Тому для коректної роботи програми необхідне стабільне функціонування операційної системи та підключеня до інтернету.

4 Опис програми

4.1 Загальний вигляд стартового вікна програми

Програмне забезпечення розроблено, як веб-застосунок доступний за визначеним посиланням. При зверненні до посилання буде відкрито головну

сторінку програми, що зображена на рисунку В.1.

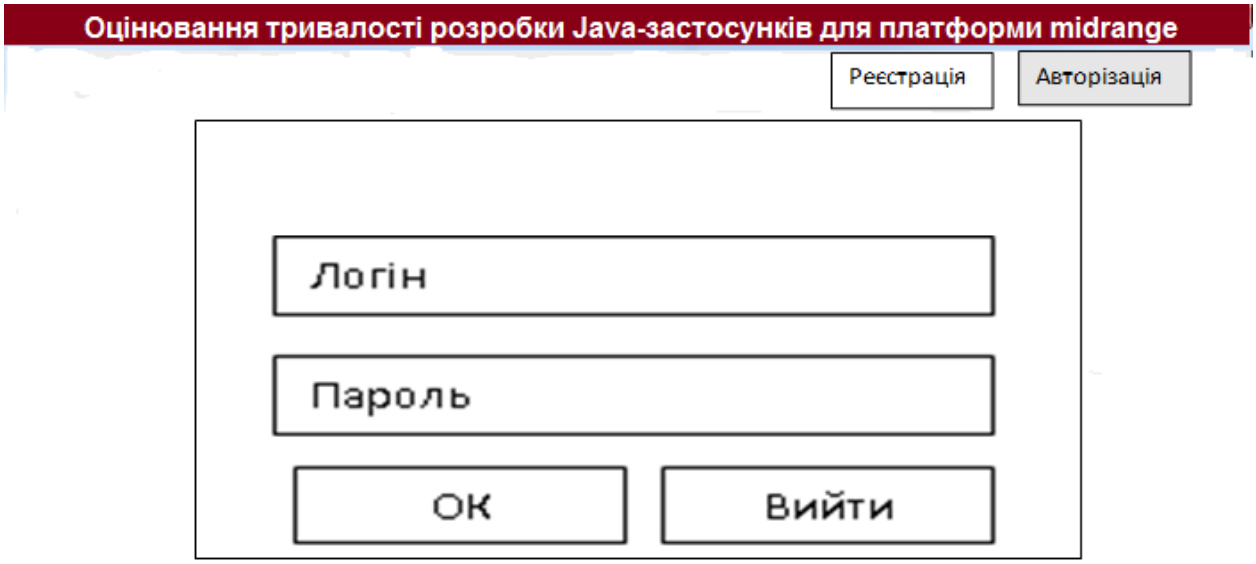

Рисунок В.1 — Екранна форма авторизації користувача – аутентифікація

Після успішної аутентифікації користувача буде відображена головна сторінка веб-додатку, яка містить головне меню зображене на рисунку В.2.

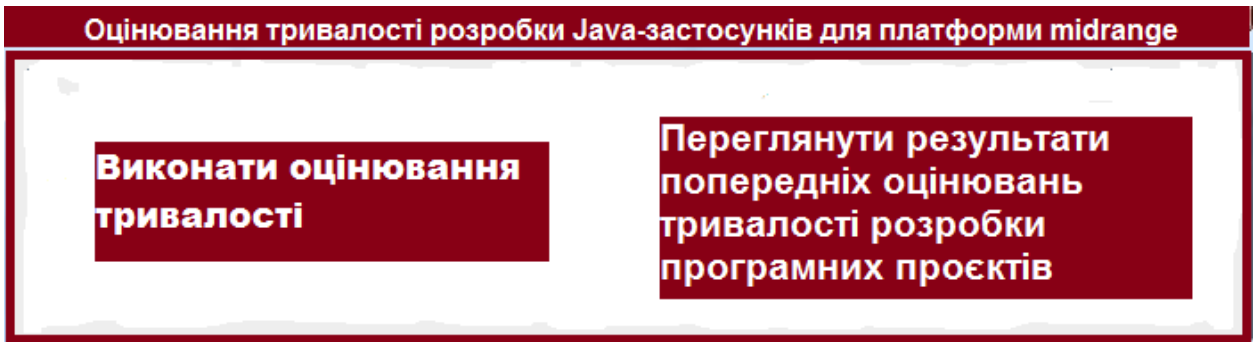

Рисунок В.2 – Екранна форма головного меню

Головна сторінка дозволяє користувачу виконати розрахунки та оцінити параметри нового проєкту або переглянути результати попередніх розрахунків.

Для виконання оцінювання часу розробки проєкту користувачу необхідно обрати «Виконати оцінювання тривалості» (рис. В.2)

Після чого в новому вікні необхідно ввести дані нового прєкту (див. рис.В.3)

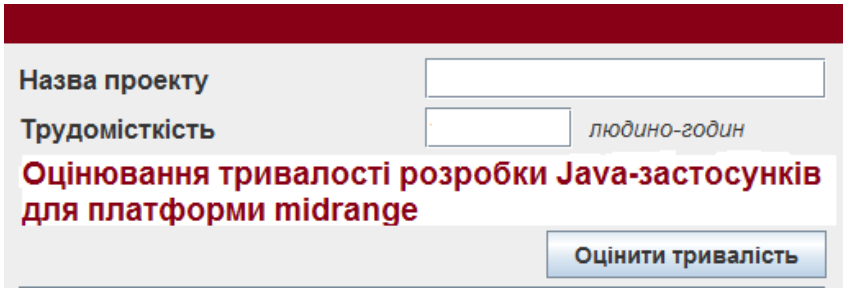

Рисунок В.3 – Вікно для введення початкових даних розрахунку

4.2 Результати оцінювання тривалості програмних проектів

Після введення вхідних даних у вказані поля вікна користувачу потрібно натиснути кнопку «Оцінити тривалість». Результати оцінювання тривалості програмних проектів (оцінка тривалості, довірчий інтервал нелінійної регресії, інтервал прогнозування) будуть показані в новому вікні (рисунку В.4).

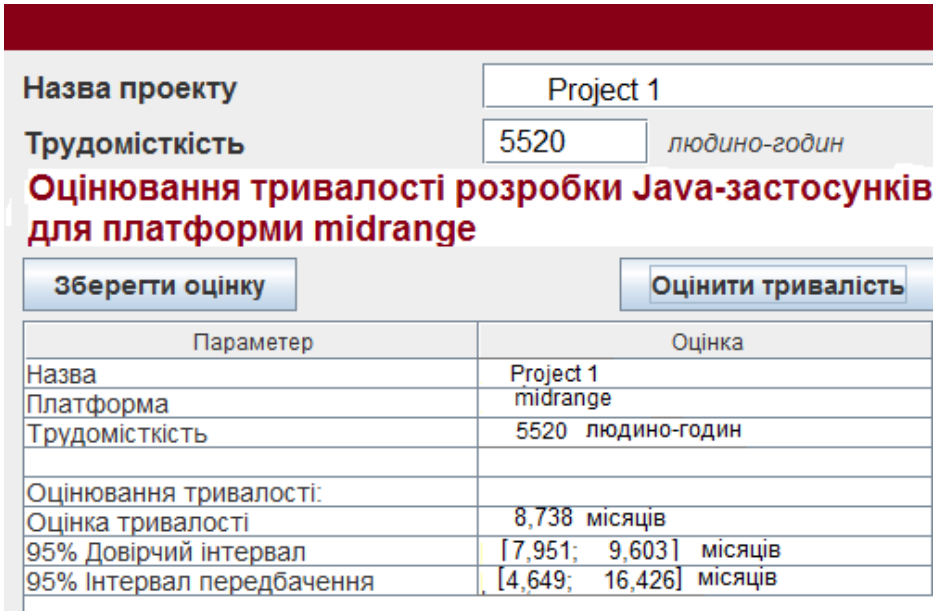

Рисунок В.4 – Вікно з результатами розрахунку

При необхідності результати оцінювання можна зберегти в базу даних. натиснувши на кнопку «Зберегти оцінку».

ДОДАТОК Г.
ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ

1 Об'єкт випробувань

Повне найменування програми: програмне забезпечення для оцінювання часу розробки Java-застосунків для платформи midrange.

Коротка характеристика галузі застосування: : програмне забезпечення застосовується для автоматизації розрахунків з оцінювання часу розробки Java-застосунків для платформи midrange, що дозволить підвищити ефективність роботи аналітика та менеджера проєкту на початкових стадіях створення програмного забезпечення.

2 Мета випробувань

Мета проведення випробувань – перевірка відповідності характеристик розробленого програмного забезпечення функціональним і окремим іншим видам вимог, викладених в документі Технічне завдання.

3 Вимоги до програми

Програма повинна забезпечувати можливість виконання нижче перерахованих функцій:

— аутентифікація користувача

— внесення інформації про проєкт:

внесення інформації про назву проекту;

внесення інформації про трудомісткість проєкту

— виконання оцінювання тривалості розробки Java-застосунків для платформи midrange:

оцінювання 95% довірчого інтервалу середнього значення тривалості розробки Java-застосунків для платформи midrange;

оцінювання 95% інтервалу прогнозування тривалості розробки Java-застосунків для платформи midrange;

— збереження результатів оцінювання тривалості розробки програмного продукту в базу даних;

— перегляд збережених результатів оцінювання тривалості розробки програмних проєктів.

4 Вимоги до програмної документації

4.1 Програмна документація містить:

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

4.2 Спеціальні вимоги

Спеціальні вимоги до програмної документації не висуваються.

5 Засоби і порядок випробувань

В процесі тестування було перевірено функціональність системи. В ході тестування було перевірено функціональні можливості програми.

5.1 Програмні та технічні засоби, що використовуються під час випробувань

Середовище виконання:

- Операційна система Windows 7 64bit Home edition;
- Java 7.
- ПК з процесором Intel I5 2,4GHz.

5.2 Порядок проведення випробувань

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

6 Порядок випробувань:

- робота на різних моніторах;
- перевірка відображення шрифтів;
- перевірка властивостей кожної форми: заголовків, трудомісткості, розташування елементів;
- перевірка змісту кожної форми на відповідність вихідним матеріалам замовника й перевірка правопису на кожній формі;
- перевірка коректності оцінювання тривалості;
- перевірка збереження та зчитування результатів з бази даних.

7 Методи випробувань

Тестування модулів ПЗ було проведено методами «чорної скриньки» та «білої скриньки»

7.1 Результати випробувань

Підтверджені можливості та функції програмного забезпечення «Оцінювання часу розробки Java-застосунків для платформи midrange».

7.2 Відомості про відмови, збої і аварійні ситуації

Відмов, збоїв і аварійних ситуацій в процесі випробувань не спостерігалося.

7.3 Відомості про коригування параметрів об’єкта випробувань та технічної документації

Коригування параметрів об’єкта випробувань та технічної документації в процесі випробувань не проводилося.

ДОДАТОК Д.
ТЕКСТ ПРОГРАМНИХ МОДУЛІВ

Клас EstimationApp

```
package EstimationApp;

import EstimationApp.forms.EstimationForm;
import EstimationApp.forms.ProjectsForm;
import oracle.toplink.essentials.config.TopLinkProperties;

import javax.persistence.EntityManager;
import javax.persistence.Persistence;
import javax.swing.*;
import java.awt.event.*;
import java.io.File;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Properties;

public class EstimationApp {

     static JFrame estimationFrame;
     static EstimationForm estimationForm;

     static JDialog projectsFrame;
     static ProjectsForm projectsForm;

     static public EntityManager em;

     public static void main(String[] args) {
         setupEntityManager();
         estimationFrame = new JFrame("Оцінювання тривалості
програмних проектів");
         estimationForm = new EstimationForm();
         estimationForm.setupUIComponents();
         setupMenu();

     estimationFrame.setContentPane(estimationForm.$$$getRootCompon
ent$$$());

     estimationFrame.setDefaultCloseOperation(WindowConstants.EXIT_
ON_CLOSE);
         estimationFrame.pack();
         estimationFrame.setVisible(true);
         estimationFrame.setLocationRelativeTo(null);
     }

    private static void setupEntityManager() {
         Properties database_properties = new Properties();
```

```
          database_properties.put(TopLinkProperties.JDBC_URL,
EstimationApp.getJDBCUrl());
          em = Persistence.createEntityManagerFactory("dur_est.db",
database_properties).createEntityManager();
     }

    private static void setupMenu() {
          JMenuBar menuBar = new JMenuBar();
          JMenu menuItem;
          menuItem = new JMenu("Інші проекти");
          menuItem.addItemListener(new ItemListener() {
             @Override
             public void itemStateChanged(ItemEvent e) {
                 if (ItemEvent.SELECTED!=e.getStateChange())
                 return;

                  ((JMenu) e.getSource()).setSelected(false);
                  showProjectsFrame();
             }
           });
           menuBar.add(menuItem);

          menuItem = new JMenu("Вихід");
          menuItem.addItemListener(new ItemListener() {
             @Override
             public void itemStateChanged(ItemEvent e) {
             if (ItemEvent.SELECTED!=e.getStateChange())
                 return;

                 exit();
             }
          });
         menuBar.add(menuItem);
         estimationFrame.setJMenuBar(menuBar);
}

     public static String getJDBCUrl(){
          try {
             File db_file = new File("eff_est.db");
             return new
URL("file:jdbc:sqlite:"+db_file.getAbsolutePath()).getPath();
          } catch (MalformedURLException e) {
                 exit(e);
          }
          return null;
     }

     public static void showProjectsFrame() {
          projectsFrame = new JDialog(estimationFrame, "Оцінки
тривалості програмних проектів", true);
          projectsFrame.setResizable(true);
          projectsForm = new ProjectsForm();
          projectsForm.setupUIComponents();
          projectsFrame.setContentPane(projectsForm.$$$getRootCompon
ent$$$());
```

```
     projectsFrame.setDefaultCloseOperation(WindowConstants.DISPOSE
_ON_CLOSE);
          projectsFrame.pack();
          projectsFrame.setVisible(true);
     }

     public static void exit(Exception e) {
          e.printStackTrace(System.err);
          exit();
     }

     public static
 void exit() {
          estimationFrame.setVisible(false);
          estimationFrame.dispose();
     }
}
```

Клас ProjectsForm

```
package EstimationApp.forms;

import EstimationApp.EstimationApp;
import EstimationApp.models.Estimation;
import com.intellij.uiDesigner.core.GridConstraints;
import com.intellij.uiDesigner.core.GridLayoutManager;
import org.jdesktop.observablecollections.ObservableCollections;
import org.jdesktop.observablecollections.ObservableList;

import javax.persistence.Query;
import javax.swing.*;
import java.awt.*;

public class ProjectsForm {

     private JPanel View;
     private JTable estimationsListTable;

     public ObservableList<Estimation> estimationList;


     public ProjectsForm() {
          Query query =
EstimationApp.em.createNamedQuery("Estimation.findAll");
          estimationList =
ObservableCollections.observableList(query.getResultList());

     }
```

```
     public void setupUIComponents() {
         final EstimationsDataModel dataModel = new
EstimationsDataModel(estimationList);
     estimationsListTable.setModel(dataModel);

     estimationsListTable.getColumnModel().getColumn(1).setMaxWidth
(80);
     estimationsListTable.getColumnModel().getColumn(3).setMaxWidth
(80);
     estimationsListTable.getColumnModel().getColumn(6).setMaxWidth
(90);
     estimationsListTable.getColumnModel().getColumn(6).setPreferre
dWidth(90);
     }

     /**
     * @noinspection ALL
     */
      private Font $$$getFont$$$(String fontName, int style, int
size, Font currentFont) {
         if (currentFont == null) {
            return null;
         }
         String resultName;
         if (fontName == null) {
            resultName = currentFont.getName();
         } else {
            Font testFont = new Font(fontName, Font.PLAIN, 10);
            if (testFont.canDisplay('a') &&
testFont.canDisplay('1')) {
                 resultName = fontName;
            } else {
                resultName = currentFont.getName();
            }
     }
         return new Font(resultName, style >= 0 ? style :
currentFont.getStyle(), size >= 0 ? size : currentFont.getSize());
     }

     {
        // GUI initializer generated by IntelliJ IDEA GUI Designer
        // >>> IMPORTANT!! <<<
        // DO NOT EDIT OR ADD ANY CODE HERE!
        $$$setupUI$$$();
      }

     /**


* Method generated by IntelliJ IDEA GUI Designer
```

```
     * >>> IMPORTANT!! <<<
     * DO NOT edit this method OR call it in your code!
     *
     * @noinspection ALL
     */
     private void $$$setupUI$$$() {
         View = new JPanel();
         View.setLayout(new GridLayoutManager(1, 1, new Insets(10,
10, 10, 10), -1, -1));
         Font ViewFont = this.$$$getFont$$$(null, -1, 16,
View.getFont());
         if (ViewFont != null) {
            View.setFont(ViewFont);
         }
         View.setPreferredSize(new Dimension(1000, 600));
         final JScrollPane scrollPane1 = new JScrollPane();
         View.add(scrollPane1, new GridConstraints(0, 0, 1, 1,
GridConstraints.ANCHOR_CENTER, GridConstraints.FILL_BOTH,
GridConstraints.SIZEPOLICY_CAN_SHRINK |
GridConstraints.SIZEPOLICY_WANT_GROW,
GridConstraints.SIZEPOLICY_CAN_SHRINK |
GridConstraints.SIZEPOLICY_WANT_GROW, null, null, null, 0, false));
         estimationsListTable = new JTable();
         estimationsListTable.setAutoCreateRowSorter(true);
         estimationsListTable.setAutoResizeMode(4);
         estimationsListTable.setFillsViewportHeight(true);
         Font estimationsListTableFont = this.$$$getFont$$$(null, -
1, 14, estimationsListTable.getFont());
         if (estimationsListTableFont != null) {

     estimationsListTable.setFont(estimationsListTableFont);
         }
         estimationsListTable.setShowHorizontalLines(true);
         scrollPane1.setViewportView(estimationsListTable);
     }
     /**
     * @noinspection ALL
     */
     public JComponent $$$getRootComponent$$$() { return View; }
}
```

Клас EstimationResults

```
package EstimationApp.forms;

import DataProcess.DataAnalyze;
import DataProcess.LinearRegression;
import ISBSG.ISBSG;
import Util.MathUtil;
```

```
import javax.swing.table.DefaultTableModel;
import java.util.ArrayList;

public class EstimationResults extends DefaultTableModel {

     public String title;
     public ISBSG.type platform;
     public Integer size;
     public ISBSG.LRParameters rp;

     public Double effort;
     public Double effort_confidence_interval_bottom;
     public Double effort_confidence_interval_top;
     public Double effort_estimation_interval_bottom;
     public Double effort_estimation_interval_top;

     public EstimationResults() {
            super(new String[]{"Параметер", "Оцінка"}, 0);
            addRow(new String[] {"1. Вкажіть значення
трудомісткості ПЗ", "(ціле число)"});
            addRow(new String[] {"2. Виберіть платформу ПЗ", ""});
            addRow(new String[] {"3. Натисніть кнопку
\"Розрахувати\"", ""});
     }

     public void fillResults() {
          setRowCount(0);
          rp = platform.getLRParameters();
          double tStudent = DataAnalyze.getTStudent(rp.n -
LinearRegression.k, 0.025); //p = 0.95
          double e_z = Math.log10(size.doubleValue());
          double t_z_estimate = LinearRegression.estimateY(e_z,
rp.b0, rp.b1);
          double t_z_estimate_l1 =
LinearRegression.estimateYConfidence(e_z, rp.b0, rp.b1, false,
false, rp.sse, rp.x_avg, rp.Sxx, rp.n, tStudent);
          double t_z_estimate_r1 =
LinearRegression.estimateYConfidence(e_z, rp.b0, rp.b1, true,
false, rp.sse, rp.x_avg, rp.Sxx, rp.n, tStudent);
          double t_z_estimate_l2 =
LinearRegression.estimateYConfidence(e_z, rp.b0, rp.b1, false,
true, rp.sse, rp.x_avg, rp.Sxx, rp.n, tStudent);
          double t_z_estimate_r2 =
LinearRegression.estimateYConfidence(e_z, rp.b0, rp.b1, true, true,
rp.sse, rp.x_avg, rp.Sxx, rp.n, tStudent);
          effort = MathUtil.round(Math.pow(10, t_z_estimate), 1);
          effort_confidence_interval_bottom =
MathUtil.round(Math.pow(10, t_z_estimate_l1), 1);
          effort_confidence_interval_top =
MathUtil.round(Math.pow(10, t_z_estimate_r1), 1);
```

```
        effort_estimation_interval_bottom =
MathUtil.round(Math.pow(10, t_z_estimate_l2), 1);
        effort_estimation_interval_top =
MathUtil.round(Math.pow(10, t_z_estimate_r2), 1);



        addRow(new String[]{"Назва", this.title});
        addRow(new String[]{"Платформа",
this.platform.toString()});
        addRow(new String[]{"Трудомісткість", size.toString() + "
функціональних точок"});
        addRow(new String[]{"", ""});
        addRow(new String[]{"Оцінювання тривалості:", ""});
        addRow(new String[]{"Оцінка тривалості", effort.toString()
+ " людино-годин"});
        addRow(new String[]{"95% Довірчий інтервал", "[" +
effort_confidence_interval_bottom + ", " +
effort_confidence_interval_top + "] людино-годин"});
addRow(new String[]{"95% Інтервал передбачення", "[" +
effort_estimation_interval_bottom + ", " +
effort_estimation_interval_top + "] людино-годин"});
    }
}
```

Клас LinearRegression

```
package DataProcess;

import Util.MathUtil;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class LinearRegression {
    List<Double> x;
    public DataAnalyzeGauss ds_x;

    List<Double> y;
    public DataAnalyzeGauss ds_y;

    List<Double> predicted_y;

    List<Double> residuals;
    DataStat ds_residuals;

    public Double b0;
    public Double b1;

    public static final int k = 2;
    public int n;
```

```
     public double tStudent;

     public LinearRegression(List<Double> x, List<Double> y) {
          setVariables(x, y);
          calculateCoefficients();
          calculateResiduals();
     }

     public LinearRegression(List<Double> x, List<Double> y, double
b0, double b1) {
          setVariables(x, y);
          this.b0 = b0;
          this.b1 = b1;
          calculateResiduals();
     }

     protected void setVariables(List<Double> x, List<Double> y) {
this.x = x;
          ds_x = new DataAnalyzeGauss(x);
          this.y = y;
          ds_y = new DataAnalyzeGauss(y);
          n = x.size();
          tStudent = DataAnalyze.getTStudent(n - k, 0.025); //p =
0.95
     }

     public void calculateResiduals() {
        predicted_y = new ArrayList<Double>();
        residuals = new ArrayList<Double>(n);
        for (int i = 0; i < n; i++) {
            double yi = b0 + b1 *x.get(i);
            predicted_y.add(yi);
            residuals.add(y.get(i) - predicted_y.get(i));
        }
        ds_residuals = new DataStat(residuals);
}

     protected void calculateCoefficients() {
          double x_avg = ds_x.getAvg();
          double y_avg = ds_y.getAvg();


          double s = 0;
          for (int i = 0; i < n; i++)
             s += (x.get(i) - x_avg) * (y.get(i) - y_avg);
          b1 = s / (ds_x.getMoment(2) * n);

          b0 = y_avg - b1 * x_avg; 93

     }
```

```
     /** Sum of squares due to regression */
     public double SSR() {
         if (true)
            return SST() - SSE();

         double avg_y = ds_y.getAvg();
         double S = 0;
         for (double y : predicted_y)
             S += Math.pow(y - avg_y, 2);

         return S;
}

      /** Sum of squares due to error */
public double SSE() { return ds_residuals.getSumPow(2); }

     /** Total sum of squares. SST = SSR + SSE */
     public double SST() { return ds_y.getMoment(2) * n; }

     public double MSR() { return SSR() / (k - 1) ; }

     public double MSE() { return SSE() / (n - k); }

     public double standardError() { return Math.sqrt(MSE()); }

     public double FRatio() { return MSR() / MSE(); }

     /** R-squared */
     public double R2() { return 1 - SSE()/SST(); }

     /** R-squared adjusted */
     public double R2Adjusted() { return 1 - (1-R2()) * (n - 1) /
(n - k); }


     public double t_b0() {
         return Math.abs(b0) * Math.sqrt(n -
k)/ds_residuals.getSqDerivation();
     }

     public double t_b1() {
         return Math.abs(b1) * Math.sqrt(n -
k)/ds_residuals.getSqDerivation()*ds_x.getSqDerivation();
     }

     /** hi */
     public double leverage(int i) {
         return 1./n + Math.pow(x.get(i)-ds_x.getAvg(), 2) / (n-1)
/ ds_x.getDisp();
}
```

```
     /** mi */
     public double getResidualSE(int i) { return Math.sqrt(MSE() *
(1 - leverage(i))); }

     public double getStandardizedResidual(int i) { return
residuals.get(i) / Math.sqrt(MSE()); }

     /** rti */
     public double getInternallyStudentizedResidual(int i) { return
residuals.get(i) / getResidualSE(i); }

     /** rt(i) */
     public double getExternallyStudentizedResidual(int i) {
         double i_stud_residual =
getInternallyStudentizedResidual(i);
         int v = n - k;
         return i_stud_residual * Math.sqrt((v-1)/(v-
i_stud_residual*i_stud_residual));
     }

     public double getCriticalLeverage() { return 2.5 * k / n; }

     public double getCriticalExternallyStudentizedResidual() {
return 2.; }

     public double estimateY(double x) {
         return estimateY(x, b0, b1);
     }

     public static double estimateY(double x, double b0, double b1)
{
         return b0 + b1 * x;
     }

        public double estimateYConfidence(double x, boolean
left_side, boolean for_values) {
        double x_avg = ds_x.getAvg();
        double Sxx = ds_x.getMoment(2) * n;
        return estimateYConfidence(x, b0, b1, left_side,
for_values, SSE(), x_avg, Sxx, n, tStudent);
     }

     public static double estimateYConfidence(double x, double b0,
double b1, boolean top, boolean for_values, double sse, double
x_avg, double Sxx, int n, double tStudent) {
         double sign = top ? +1 : -1;
         double k1 = for_values ? 1 : 0;
         return estimateY(x, b0, b1) + sign * tStudent *
Math.sqrt(sse/(n - k)) * Math.sqrt(k1 + 1./n + Math.pow(x - x_avg,
2) / Sxx);
      }

public void printIsolatedEntries () {
```

```
         double criticalLeverage = getCriticalLeverage();
         System.out.println("Critical Leverage="+
criticalLeverage);
         for (int i = 0; i < n; i++) {
            final double leverage = leverage(i);
            if (leverage < criticalLeverage )
            continue;
            System.out.println(i + String.format(" %.6g",
leverage));
         }

         System.out.println();
          double criticalExternallyStudentizedResidual =
getCriticalExternallyStudentizedResidual();
          System.out.println("Critical Studentized Residual="+
criticalExternallyStudentizedResidual);
          for (int i = 0; i < n; i++) {
          final double stud_residual =
getExternallyStudentizedResidual(i);
            if (stud_residual <
criticalExternallyStudentizedResidual)
                    continue;

                System.out.println(i + String.format(" %.6g",
stud_residual));
            }
         }

     public List<Integer> getIsolatedIndexes() {
         double criticalLeverage = getCriticalLeverage();
         double criticalExternallyStudentizedResidual =
getCriticalExternallyStudentizedResidual();
         List<Integer> isolated = new ArrayList<Integer>();
         for (Integer i = 0; i < n; i++) {
            final double leverage = leverage(i);
            final double stud_residual =
getExternallyStudentizedResidual(i);
            if (leverage < criticalLeverage && stud_residual <
criticalExternallyStudentizedResidual)
                  continue;

           isolated.add(i);
           if (true)
                continue;
           System.out.print(i);
           if (!(leverage < criticalLeverage))
                System.out.print(" High leverage");
           if (!(stud_residual <
criticalExternallyStudentizedResidual))
                System.out.print(" High residual");
           System.out.println();

         }
```

```
          return isolated;
      }
}
```

Клас Estimation

```
package EstimationApp.models;

import javax.persistence.*;

@Entity
@Table(name = "estimations", catalog = "", schema = "")
@NamedQueries({
    @NamedQuery(name="Estimation.findAll", query="SELECT e FROM
Estimation e"),
})


      public class Estimation extends AppModel {
      public String title;
      public String platform;
      public Integer size;
      public Double effort;
      public Double effort_confidence_interval_bottom;
      public Double effort_confidence_interval_top;
      public Double effort_estimation_interval_bottom;
      public Double effort_estimation_interval_top;
      public String created;


      @Override
      protected Class getEntityClass() {
          return Estimation.class;
      }
}
```

Клас AppModel

```
package EstimationApp.models;

import javax.persistence.*;
import java.io.Serializable;

@MappedSuperclass
public abstract class AppModel implements Serializable {
      @Id
      public Integer id;

      public AppModel() {
```

```
    }
    public AppModel(Integer id) {
        this.id = id;
}

    public Integer getId() {
        return id;
}

    public void setId(Integer id) {
        System.out.println("Id set to "+id);
        this.id = id;
}

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (id != null ? id.hashCode() : 0);
        return hash;
    }

     @Override
     public boolean equals(Object object) {
// TODO: Warning - this method won't work in the case the id fields
are not set
        if (!this.getEntityClass().isInstance(object))
return false;
        AppModel other = (AppModel) object;
        return !((this.id == null && other.id != null) || (this.id
!= null && !this.id.equals(other.id)));
     }

    protected abstract Class<? extends AppModel> getEntityClass();

    @Override
    public String toString() {
return getClass().getSimpleName()+"[id=" + id + "]";
   }
}
```

Клас LRParameters

```
    public class LRParameters {

        public double b0;
        public double b1;
        public double sse;
        public double x_avg;
        public double Sxx;
        public int n;
```

```
          LRParameters(double b0, double b1, double sse, double
x_avg, double Sxx, int n) {
             this.b0 = b0;
             this.b1 = b1;
             this.sse = sse;
             this.x_avg = x_avg;
             this.Sxx = Sxx;
             this.n = n;
          }
      }
```