МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

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

Програмного забезпечення автоматизованих систем
(повна назва кафедри)

Пояснювальна записка
до кваліфікаційної (магістерської) роботи

за темою: «Удосконалення регресійної моделі для оцінювання розміру програмних застосунків на TypeScript та розробка програми для її реалізації»

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

Чебаков Д.В.
(підпис, прізвище та ініціали)
Керівник Латанська Л.О.
(підпис, прізвище та ініціали)
Рецензент Приходько С.Б.
(підпис, прізвище та ініціали)
Завідувач кафедри Приходько С.Б.
(підпис, прізвище та ініціали)

м. Миколаїв – 2020 р.

Міністерство освіти і науки України
Національний університет кораблебудування
імені адмірала Макарова

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

ЗАТВЕРДЖУЮ
Завідувач кафедри

" 26 " 10 2020 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ (МАГІСТЕРСЬКУ) РОБОТУ СТУДЕНТУ

Чебакову Дмитру Вадимовичу
(прізвище, ім'я, по батькові)

1.Тема магістерської роботи Удосконалення регресійної моделі для оцінювання розміру програмних застосунків на TypeScript та розробка програми для її реалізації

керівник роботи Латанська Людмила Олексіївна, к.ф.-м. н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від " 26 " жовтня 2020 року №1037-уч
2. Строк подання студентом роботи 01.12.2020 року
3. Вихідні дані до роботи

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

5. Перелік графічного матеріалу

_____
_____
_____
_____
_____
_____

6. Консультанти розділів магістерської роботи

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

7. Дата видачі завдання___

КАЛЕНДАРНИЙ ПЛАН

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

Студент _____ _____
(підпис) (прізвище та ініціали)

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

РЕФЕРАТ

Чебаков Дмитро Вадимович

«Удосконалення регресійної моделі для оцінювання розміру програмних застосунків на TypeScript та розробка програми для її реалізації»

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

Обсяг роботи: 102 стор., 20 таблиць, 16 рисунків, 21 використане джерело, 5 додатків.

Актуальність теми: оцінка розміру програмного забезпечення є основою для більщості важливих оцінок програмного продукту. Незважаючи на постійні зусилля дослідників в напрямку оцінювання розміру програмного забезпечення, розробники так і не отримали ефективного, універсального інструменту прогнозування розміру майбутньої програми.

Таким чином, удосконалення методів та моделей оцінювання розміру програмного забезпечення в цілому, в тому числі програмних застосунків на TypeScript, є актуальною задачею інженерії програмного забезпечення.

Мета і завдання дослідження. Метою даної кваліфікаційної роботи є підвищення достовірності оцінювання розміру програмних застосунків на TypeScript. Для досягнення поставленої мети необхідно виконати аналіз існуючих методів та моделей оцінювання розміру програмного забезпечення; обґрунтувати необхідності удосконалення регресійної моделі для оцінювання розміру програмних застосунків на TypeScript; розробити нелінійну регресійну модель та програму оцінювання розміру ПЗ в залежності від кількості класів.

Об’єкт дослідження: процес оцінювання розміру програмних застосунків на TypeScript.

Предмет дослідження: нелінійна регресійна модель для оцінювання розміру програмних застосунків на TypeScript.

Методи дослідження: при проведенні досліджень використовувалися методи теорії ймовірностей, математичної статистики та регресійного аналізу.

Наукова новизна одержаних результатів: удосконалено регресійну модель для оцінювання розміру програмних застосунків на TypeScript, що дозволило підвищити достовірність оцінювання в порівнянні з існуючими лінійними моделями.

Практичне значення одержаних результатів: розробка алгоритму і програми для оцінювання розміру програмних застосунків на TypeScript на основі нелінійної регресійної молелі з нормалізуючим перетворенням у вигляді десяткового логарифму, що дозволило отримати більш точну оцінку розміру програмного забезпечення.

Апробація результатів досліджень: результати досліджень, викладених у роботі, були оприлюднені на VI міжнародній науково-технічній конференції «Комп’ютерне моделювання та оптимізація складних систем» (м. Дніпро, 04-06.11.2020).

ABSTRACT

Chebakov Dmytro

«Improving the regression model for estimating the size of software applications in TypeScript and developing the software for its implementation»

Qualification work for obtaining a master's degree in Specialty 121 – Software Engineering. Admiral Makarov National University of Shipbuilding. Mykolaiv, 2020.

The work contains: 102 pages, 20 tables, 16 figures, 21 references, 5 appendices.

Relevance of the topic: estimating software size is the basis for most important software product evaluations. Despite the constant efforts of researchers to estimate the size of the software, the developers have not received an effective, universal tool for predicting the size of the future program.

Thus, improving the methods and models for estimating the size of software in general, including software applications in TypeScript, is an urgent task of software engineering.

The purpose and objectives of the study. The purpose of this qualification work is to increase the reliability of estimating the size of software applications in TypeScript. To achieve this goal it is necessary to perform an analysis of existing methods and models for estimating the size of software; justify the need to improve the regression model to estimate the size of software applications in TypeScript; develop a nonlinear regression model and software size estimation program depending on the number of classes.

Object of research: the process of estimating the size of software applications in TypeScript.

Subject of research: regression equation for estimating the size of dynamic site software in PHP.

Research methods: the methods of probability theory, mathematical statistics and regression analysis were used in the research.

The scientific novelty of the obtained results: improved a regression model for estimating the size of software applications in TypeScript, which increased the reliability of the estimate compared to existing linear models.

The practical value of the results: development of an algorithm and program for estimating the size of software applications on TypeScript based on a nonlinear regression molecule with normalizing transformation in the form of a decimal logarithm, which allowed to obtain a more accurate estimate of software size.

Approbation of study results: The results of the research presented in the paper were published at the 6th International Scientific and Technical Conference "Computer Modeling and Optimization of Complex Systems" (Dnipro, 04-06.11.2020).

ЗМІСТ

ВСТУП

Актуальність теми. Управління розробкою програмного забезпечення, у тому числі і програмних застосунків на TypeScript, – це мистецтво планування і керування проектами з розробки програмного забезпечення. Однією з перших дій планування є оцінювання, яке закладає фундамент для інших дій з планування проекту. На цьому етапі надзвичайно висока ціна помилок. Тому дуже важливо провести оцінку з мінімальним ризиком.

При правильній методології оцінка розміру програмного забезпечення стає основою для більщості важливих оцінок, наприклад, тривалості, трудомісткості та інших. Розмір створюваної системи є найважливішим чинником, що визначає її вартість [1, 2].

Оцінювання розміру базується на знанні вимог до системи. Для такого оцінювання існують два основних підходи:

– За аналогією. Якщо в минулому доводилося мати справу з подібними проектами та їх оцінки відомі, то є можливість приблизно оцінити свій проект.

– Шляхом підрахунку розміру за певними алгоритмами на основі вихідних даних (вимог до системи).

Розмір програмного проекту можна виміряти в:

– об'єктних одиницях;

– функціональних одиницях;

– кількості рядків вихідного коду.

Незважаючи на постійні зусилля дослідників в напрямку оцінювання розміру програмного забезпечення, розробники так і не отримали ефективного, універсального інструменту прогнозування розміру майбутнього продукту.

Таким чином, удосконалення моделей оцінювання розміру програмного забезпечення в цілому, в тому числі програмних застосунків на TypeScript, є актуальною задачею інженерії програмного забезпечення [3, 4].

Мета і завдання дослідження. Метою даної кваліфікаційної роботи є підвищення достовірності оцінювання розміру програмних застосунків на TypeScript.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

– проаналізувати існуючі методи та моделі оцінювання розміру ПЗ, порівняти їх;

– обґрунтувати необхідність удосконалення регресійної моделі для оцінювання розміру програмних застосунків на TypeScript.

– зібрати дані, які можуть бути використані для удосконалення моделі;

– нормалізувати отримані емпіричні дані, використовуючи перетворення у вигляді десяткового логарифму;

– перевірити нормалізовані дані на викиди;

– побудувати лінійну регресійну модель, довірчий інтервал та інтервал прогнозування для нормалізованих даних;

– перейти від лінійної регресії до нелінійної та побудувати регресійну модель, довірчий інтервал та інтервал прогнозування для вихідних даних;

– побудувати лінійну регресійну модель, довірчий інтервал і інтервал прогнозування для вихідних даних без нормалізації в припущенні про нормальність вихідних даних;

– виконати порівняння отриманих лінійної та нелінійної моделей;

– розробити ПЗ для оцінювання розміру програмних застосунків на TypeScript.

Об’єктом досліджень є процес оцінювання розміру програмних застосунків на TypeScript.

Предметом досліджень нелінійна регресійна модель для оцінювання розміру програмних застосунків на TypeScript.

Методи дослідження. Поставлені завдання кваліфікаційної роботи будуть вирішуватися з використанням методів теорії ймовірностей, математичної статистики та регресійного аналізу.

Наукова новизна одержаних результатів полягає в удосконаленні нелінійної регресійної моделі для оцінювання розміру програмних застосунків на TypeScript за рахунок застосування нормалізуючого перетворення у вигляді десяткового логарифму, що дозволяє підвищити достовірність оцінювання в порівнянні з існуючими лінійними моделями.

Практичне значення одержаних результатів полягає в розробці алгоритму і програми для оцінювання розміру програмних застосунків на TypeScript на основі нелінійної регресійної моделі з нормалізуючим перетворенням у вигляді десяткового логарифму, що дозволило отримати більш точну оцінку розміру програмного забезпечення.

Особистий внесок здобувача. Кваліфікаційна робота є самостійно виконаною працею. Усі результати, викладені у роботі, отримані автором особисто. У тезах доповіді [5], написаних у співавторстві, здобувачу належить дослідження питання розрахунку довірчих інтервалів для лінійної та нелінійної регресії.

Апробація результатів досліджень. Основні положення та результати досліджень, викладені у кваліфікаційній роботі, пройшли апробацію на VI міжнародній науково-технічній конференції «Комп’ютерне моделювання та оптимізація складних систем» (м. Дніпро, 04-06.11.2020).

Публікації. Основні результати магістерської роботи викладено у 1 науковій праці – тезах конференції [5].

1 МОВА ПРОГРАМУВАННЯ TYPESCRIPT ТА СФЕРИ ЇЇ ЗАСТОСУВАННЯ

1.1 Сутність, переваги та недоліки мови програмування TypeScript

На сьогодні існує чимало мов програмування, якими користуються веб-розробники. До найбільш поширених мов відносяться: Perl, PHP, Python, JAVA, Ruby, JavaScript. Кожна з перелічених мов має свої переваги та недоліки. Останнім чином зростає популярність мови програмування TypeScript (в 2019 році мова TypeScript попала в топ-10 по популярності). Розглянемо сутність, переваги та недоліки цієї мови.

TypeScript – це мова програмування, розроблена Microsoft. Специфікації мови відкриті і опубліковані в рамках угоди Open Web Foundation Specification Agreement.TypeScript – це мова програмування, в якій виправлено багато недоліків JavaScript.

Мова TypeScript почала активно розвиватися декілька років тому. Такий розвиток їй забезпечує підтримка Microsoft.

Мова TypeScript є надбудовою над JavaScript. Тому в TypeScript можна використовувати існуючий JavaScript-код, включати популярні бібліотеки JavaScript і викликати TypeScript-код, згенерований з інших JavaScript [6]. В даний час TypeScript – одна з головних мов, які компілюються в JavaScript.

Розглянемо переваги та недоліки TypeScript.

До переваг TypeScript можна віднести:

– компіляцію програми в JavaScript код, з яким можна працювати без TypeScript;

– підтримку класів, підкласів;

– можливість описати область видимості властивостей класу;

– підтримку інтерфейсів, які підходять для повторного використання;

– підтримку ООП з наслідуванням закритих членів та інтерфейсів;

– підтримку модулів;

– підтримку строгої типізації, яка дозволяє перевіряти правильність типу під час компіляції;

– багато доступних IDE;

– на TypeScript написаний фреймворк Angular 2 і для використання всіх переваг цього фреймворка необхідно використовувати TypeScript;

– досить потужна підтримка бібліотек;

– інструментарій TypeScript.

Недоліки TypeScript:

– більш складна мова в порівнянні з JavaScript;

– необхідні базові знання з ООП;

– більш складний процес збирання;

– вимагає компіляції.

1.2 Сфери використання TypeScript

Мову TypeScript рекомендують використовувати, якщо:

– досить великикі та комплексні проекти;

Якщо команда працює над проектом з великим кодом, то TypeScript допоможе обійти багато помилок.

– над задачею працює велика команда;

Чим більша команда, тим важче їх контролювати. Тому частину контролю краще передати машині.

– в перспективі потребується великий рефакторінг;

TypeScript краще використовувати для довгострокових проектів, для яких в майбутньому буде необхідний великий рефакторінг.

– команда знайома зі строго типізованими мовами;

Якщо команда знайома з типізованими мовами, – це пришвидшить вивчення TypeScript.

– бібліотека / фреймворк передбачають TypeScript.

Для тих, хто використовує фреймворк з TypeScript.

2 ДОСЛІДЖЕННЯ ІСНУЮЧИХ МЕТОДІВ І МОДЕЛЕЙ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Метрики програмного забезпечення

Для оцінки програмних проектів використовуються метрики. Існує чимало класифікацій метрик. Метрики існують кількісні та якісні. Зазвичай для оцінювання програмного забезпечення використовуються кількісні метрики, які піддаються статистичній обробці. Розглянемо найбільш вживані метрики програмного забезпечення [7]:

– Метрики розміру

Існує чимало метрик розміру, а саме:

- кількість рядків коду;
- кількість порожніх рядків;
- кількість рядків коду, включаючи коментарі;
- середня кількість рядків методів;
- середня кількість рядків функцій;
- середня кількість рядків модулів;
- середня кількість рядків класів [7].

Також до метрик розміру відносяться функціональні точки. Ці метрики базуються на: вимогах користувача, специфікаціях, описах прецедентів і т.д. Найчастіше ці метрики використовуються при обчисленні таких характеристик програмних проектів, як трудомісткість, вартість, тривалість та інших) [7].

– Метрики складності

Метрики складності характеризують складність програмних проектів та використовуються в основному для оцінки складності розроблених програмних проектів. Також можна використовувати метрики складності на

етапі проектування програмного забезпечення. З допомогою метрик складності визначаються найбільш вразливі ділянки проекту, які можуть привести до помилок та являються точками ризику. Серед метрик складності найбільш вживаною є метрика цикломатична складність. Для обчислення цієї метрики необхідно побудувати граф керування з обчислювальними оператори або виразами в вигляді вузлів, а передача управління між вузлами представляється у вигляді дуг [8].

– Об'єктно-орієнтовані метрики

До цієї групи метрик відносяться:

- загальне число класів;
- загальне число методів;
- загальне число атрибутів;
- середнє число методів на клас;
- середнє число атрибутів на клас [7].

Обчислення метрики в ході реалізації проекту дозволяє своєчасно визначити найбільш складні структурні одиниці, що супроводжуються високими ризиками, та вжити заходів щодо усунення ризиків за рахунок внесення коректівів.

Визначення розміру програмного забезпечення є ключовим параметром будь-якої оціночної моделі. Як правило, кількість рядків коду програми є негаусівською випадковою величиною, яка залежить від ряду факторів, у тому числі й метрик програмного забезпечення, які в тому чи іншому випадку впливають на кінцевий результат.

2.2 Методи та моделі для оцінювання розміру програмного забезпечення

Напрямки оцінювання розміру програмного забезпечення можна поділити на наступні категорії [9, 10, 11]:

– Методи, основані на моделях. На основі математичної моделі відповідно до вхідних параметрів, отриманих від проекту, обчислюється його розмір.

– Методи, основані на експертному оцінюванні. Базуються на думках експертів, які мають значний досвід в розробці програмного забезпечення в заданій сфері.

– Методи, які базуються на функціональних точках. Методи призначені для оцінювання кількості функціоналу і основані на логічній моделі обсягу програмного продукту, який вимагає замовник і постачає розробник.

– Методи оцінювання з можливістю навчання Це оновлений метод оцінювання за аналогією з подальшим навчанням системи оцінювання шляхом такої техніки штучного інтелекту, як нейронні мережі.

– Динамічні методи оцінювання. Використовують динаміку зміни параметрів.

– Регресійні методи. Використовуються в поєднанні з методами, основаними на моделях, і включають стандартну регресію, робастну регресію і т.п.

– Композитні методи. Є комбінацією двох і більше методів для формування найбільш точної системи оцінювання.

Дослідження продовжують активно розвиватися, вдосконалюючи вже існуючі методи та моделі, і створюючи нові.

2.3 Перевірка якості рівняння регресії для оцінювання розміру програмного забезпечення

Для перевірки якості рівняння регресії використовується коефіцієнт детермінації R^2 [12, 13]:

$$R^2 = 1 - \left(\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 / \sum_{i=1}^{n} (y_i - \bar{y})^2 \right), \qquad (2.1)$$

де y_i – емпіричне значення y;

$\hat{y}_i$ – розраховане значення y;

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

R^2 показує, наскільки рівняння регресії відповідає реальним даним. Коефіцієнт детермінації змінюється від 0 до 1. Якщо він дорівнює 0, це означає, що зв'язок між змінними в регресійній моделі відсутній. При R^2 =1 отримуємо ідеальну модель. Якщо R^2 близький до 1, то модель має високу значимість, а якщо до 0, – низьку значимість моделі [12].

Для перевірки якості знайденого рівняння регресії, окрім критерію R^2, використовують суму квадратів відхилень S_y:

$$S_y = \sum_{i=1}^{n} [y_i - \hat{y}_i]^2, \qquad (2.2)$$

де y_i – емпіричне значення;

$\hat{y}_i$ – розрахункове значення згідно рівняння регресії.

Сума квадратів відхилень S_y використовується для перевірки якості як нелінійного, так і лінійного рівняння регресії. Цей параметр також використовується для порівняння різних моделей.

Найбільш часто для вимірювання точності оцінки використовуються середня величина відносної похибки MMRE (Mean Magnitude Relative Error) та рівень прогнозування PRED(x) (Prediction Within x). Обидві ці оцінки базуються на значенні відхилення відносної похибки (Magnitude of Relative Errors) [13, 14].

Величина відносної похибки MRE (Magnitude of Relative Errors) для лінійного та нелінійного рівняннь регресії обчислюється за формулою:

$$MRE_i = \left| \frac{y_i - \hat{y}_i}{y_i} \right|, \qquad (2.3)$$

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

y_i – емпіричне значення.

MMRE (Mean of MRE – середня величина відносної похибки) знаходиться за формулою:

$$MMRE = \frac{1}{N}\sum_{i=1}^{N} MRE_i \qquad (2.4)$$

Рівень прогнозування (PRED(l)) визначається за формулою:

$$PRED(l) = \frac{k}{N}, \qquad (2.5)$$

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

2.4 Обґрунтування необхідності проведення досліджень

Виконаний аналіз існуючих методів та моделей оцінювання розміру програмного забезпечення показав, що існуючі на сьогоднішній день методи та моделі мають ряд недоліків, що не дозволяють в повному обсязі оцінити розмір програмного забезпечення з необхідною достовірністю. А отже є необхідність побудувати математичну модель, яка б покращила достовірність передбачення та дала змогу швидко та надійно прогнозувати розмір програмних застосунків на TypeScript.

Для побудови математичної моделі будемо використовувати регресійний аналіз. Зазвичай вихідні дані для оцінювання розміру ПЗ не підпорядковуються нормальному закону розподілу. Це негативно впливає на достовірність отриманих результатів, що в результаті призводить до помилок у розрахунках, у нашому випадку, в оцінюванні розміру ПЗ. Тому перед побудовою регресійної моделі потрібно виконати нормалізацію емпіричних даних.

Для нормалізації негаусівських даних можуть бути використані перетворення на основі десяткового або натурального логарифму,

перетворення Box-Cox, перетворення Джонсона та інші. У даній роботі в якості нормалізуючого перетворення буде використовуватись десятковий логарифм.

Застосування нормалізуючого перетворення дає можливість застосовувати лінійний регресійний аналіз для нормалізованих даних, для них побудувати довірчий інтервал та інтервал прогнозування, і через використання зворотного перетворення перейти до нелінійної регресії з одночасним урахуванням особливостей розробки програмних програмних застосунків на TypeScript та нелінійних зв’язків між негаусівськими даними.

3 УДОСКОНАЛЕННЯ РЕГРЕСІЙНОЇ МОДЕЛІ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНИХ ЗАСТОСУНКІВ НА TYPESCRIPT ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ПРОГРАМИ ДЛЯ ЇЇ РЕАЛІЗАЦІЇ

3.1 Аналіз підходу до побудови нелінійної регресійної моделі та вибір факторів для оцінювання розміру програмного забезпечення

При побудові нелінійної регресійної моделі для даних, які не підпорядковуються нормальному закону, був використаний підхід, який полягає в наступному:

– за допомогою нормалізуючого перетворення виконується перехід до нормалізованих даних (в роботі вибране перетворення десятковий логарифм);

– для нормалізованих даних використовується лінійна регресійна теорія та отримується лінійна модель, прогнозні значення, довірчі інтервали та інтервали прогнозування.

– через зворотнє перетворення отримується нелінійна модель, прогнозні значення, довірчі інтервали та інтервали прогнозування для емпіричних даних.

Грунтуючись на аналізі методів та моделей для оцінювання розміру програмного забезпечення, можна зробити висновок про те, що використання нелінійної регресійної теорії являється актуальним в задачах покращення якості оцінювання розміру [15]. Для оцінювання розміру в якості емпіричних даних будуть використовуватися кількість класів та кількість строк коду.

3.2 Побудова удосконаленої регресійної моделі оцінювання розміру програмних застосунків на TypeScript

Для побудови удосконаленої регресійної моделі оцінювання розміру програмних застосунків на TypeScript були зібрані вихідні дані з завершених

проектів. В якості вихідних даних обрано кількість класів *X* та кількість строк коду *Y*.

Розмір програмного забезпечення буде розраховуватися через кількість класів, тобто в якості незалежнохї змінної розглядаємо кількість класів, а в якості залежної – кількість строк коду:

$$Y = f(X) \quad (3.1)$$

Вихідні дані для побудови регресійної моделі наведені в таблиці 3.1.

Таблиця 3.1 – Вихідні дані для побудови регресійної моделі

№ пр.	X класів	Yстрок
1	1259	40204
2	390	33122
3	2996	120387
4	480	30243
5	3522	162383
6	612	10006
7	918	60171
8	821	75587
9	653	33663
10	709	76468
11	1643	59519
12	1117	86383
13	479	47138
14	1211	95226
15	237	29645
16	971	80929
17	118	21866
18	611	34461
19	871	46123
20	2013	61771
21	97	11920
22	451	33530
23	551	54970
24	791	61697
25	180	13410
26	212	19874
27	387	32972
28	418	53959
29	673	61343
30	872	61935

Для перевірки нормальності розподілу емпіричних даних застосовується критерій згоди Пірсона (критерій χ^2). Встановлено, що вихідні дані не є гаусівськими. Із цього слідує, що для побудови нелінійної регресійної моделі необхідно нормалізувати дані [16].

В якості нормалізуючого перетворення обрано десятковий логарифм. В результаті отримано нормалізовані величини Z_Y та Zx, які представлені в таблиці 3.2.

Таблиця 3.2 – Нормалізовані вихідні дані

№ пр.	Zx	Z_Y
1	1259	40204
2	390	33122
3	2996	120387
4	480	30243
5	3522	162383
6	612	10006
7	918	60171
8	821	75587
9	653	33663
10	709	76468
11	1643	59519
12	1117	86383
13	479	47138
14	1211	95226
15	237	29645
16	971	80929
17	118	21866
18	611	34461
19	871	46123
20	2013	61771
21	97	11920
22	451	33530
23	551	54970
24	791	61697
25	180	13410
26	212	19874
27	387	32972
28	418	53959
29	673	61343
30	872	61935

Вихідний та нормалізований розподіли для вибірки Y представлено на рис. 3.1.а та рис. 3.1.б відповідно.

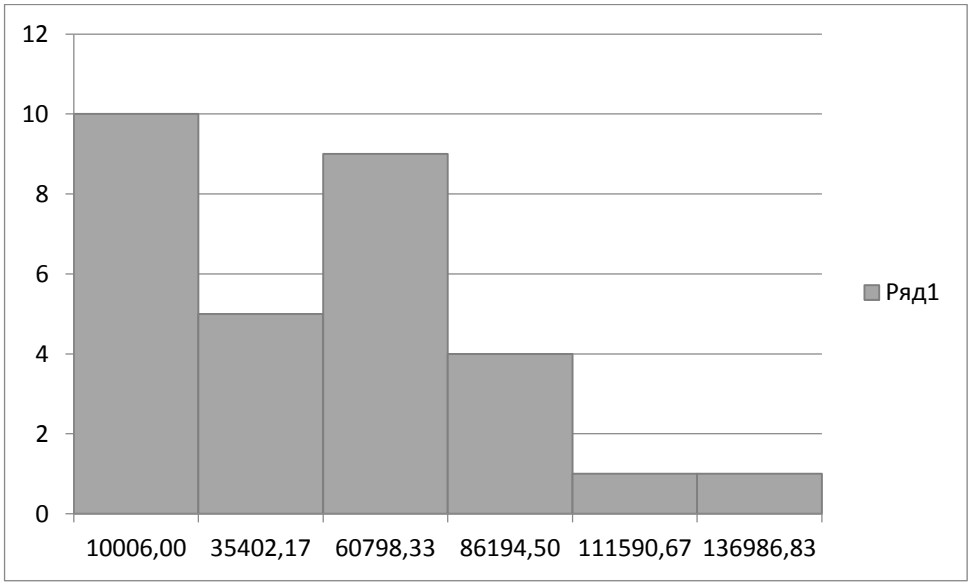

Рисунок 3.1.а – Вихідний розподіл для вибірки Y

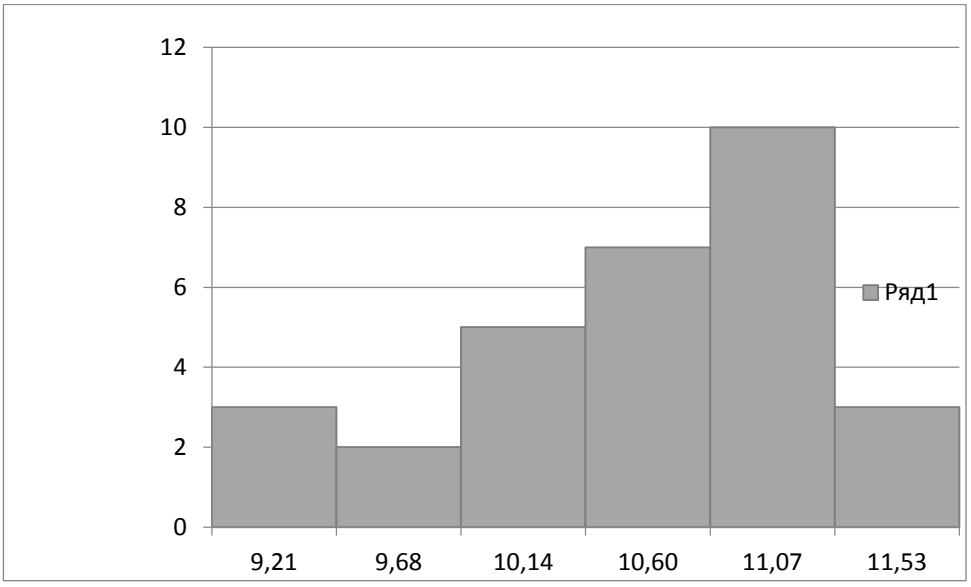

Рисунок 3.1.б – Нормалізований розподіл для вибірки Y

Вихідний та нормалізований розподіли для вибірки X представлено на на рис. 3.2.а та рис. 3.2.б. відповідно.

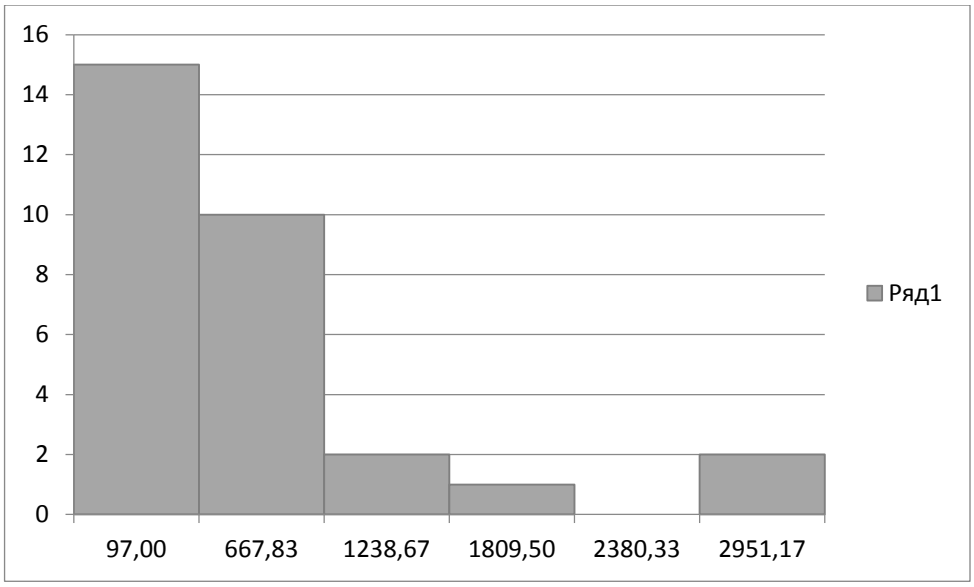

Рисунок 3.2.а – Вихідний розподіл для вибірки X

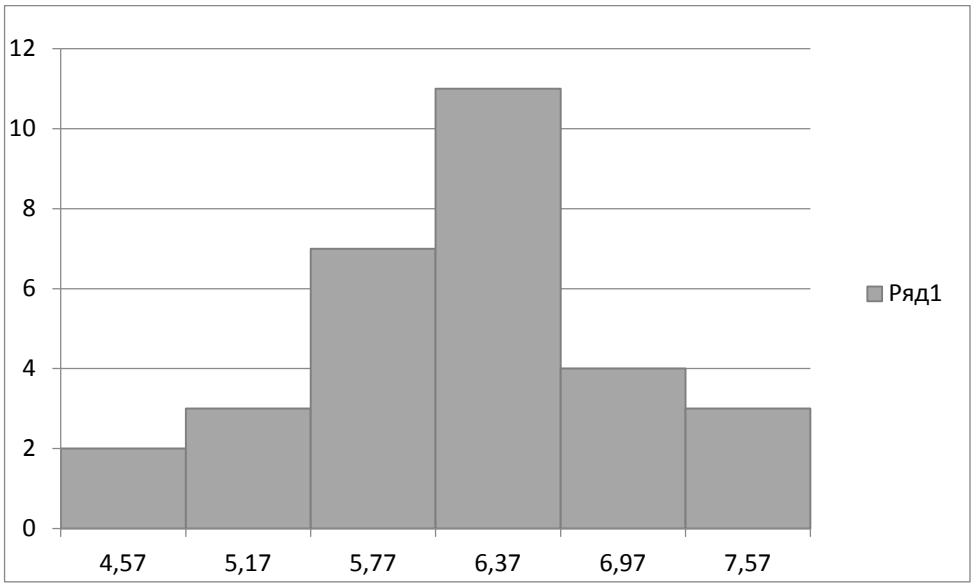

Рисунок 3.2.б – Нормалізований розподіл для вибірки X

Нелінійну регресійну модель будуємо на основі лінійної регресійної моделі. Загальний вигляд лінійної регресійної моделі з двома параметрами в нашому випадку наступний [5, 18]:

$$Z_y = b_0 + b_1 Z_x + \varepsilon. \qquad (3.2)$$

Значення коефіцієнтів були розраховані за методом найменших квадратів:

$$b_1 = 0{,}6278,\ b_0 = 2{,}8887.$$

Звідси

$$\hat{Z}_y = 2{,}8887+0{,}6278Z_x+\varepsilon.$$

Виконавши нормалізацію вихідних даних та отримавши лінійну регресійну модель, будуємо довірчий інтервал та інтервал прогнозування. Довірчий інтервал – це границі прогнозу (верхня і нижня), в рамки яких із заданою ймовірністю потраплять фактичні значення. Інтервал прогнозування – це кількісна оцінка невизначеності в прогнозі. Інтервал прогнозування зазвичай більше довірчого інтервалу, оскільки він повинен враховувати довірчий інтервал і прогнозовану дисперсію вихідної змінної. Довірчий інтервал будуємо за формулою (3.3) [5, 18]:

$$Z_y = \hat{Z}_y \pm t_{\alpha/2,\vartheta}\ S_{Z_y}\sqrt{\frac{1}{n}+\frac{(x-\overline{x})^2}{\sum_n^1(x_i-\overline{x})^2}}, \tag{3.3}$$

де $S_{Z_y} = \sqrt{\frac{1}{n-2}\sum_1^n(y_i - \hat{y}_i)^2}$;

$t_{\alpha/2,\vartheta}$ – квантіль t-розподілу Стьюдента.

Інтервал прогнозування визначається за формулою (3.4) [5, 18]:

$$Z_y = \hat{Z}_y \pm t_{\alpha/2,\vartheta}\ S_{Z_y}\sqrt{1+\frac{1}{n}+\frac{(x-\overline{x})^2}{\sum_n^1(x_i-\overline{x})^2}}, \tag{3.4}$$

де $S_{Z_y} = \sqrt{1+\frac{1}{n-2}\sum_1^n(y_i - \hat{y}_i)^2}$;

$t_{\alpha/2,\vartheta}$ – квантіль t-розподілу Стьюдента.

Для повернення до вихідних даних та отримання нелінійної регресійної моделі необхідно виконати зворотнє перетворення [5, 19]. Формула зворотнього перетворення для функції має вигляд:

$$Y = 10^{\hat{Z}_y} \ . \tag{3.5}$$

Застосовавши аналогічну формулу до границь довірчого інтервалу та інтервалу прогнозування лінійної регресії, отримаємо верхню та нижню границю довірчого інтервалу та інтервалу прогнозування для нелінійної регресійної моделі оцінювання розміру програмних застосунків на TypeScript

Нелінійна регресійна модель оцінювання розміру програмних застосунків на TypeScript має вигляд:

$$10^{b_0+\varepsilon} * X^{b_1}.$$

Таким чином нелінійна регресійна модель має вигляд:

$$Y = 10^{2,8887+\varepsilon} * X^{0,6278}.$$

В таблиці 3.3 знаходяться результати виконаних розрахунків: довірчі інтервали для вихідних даних, нормалізованих за допомогою десяткового логарифму, та для ненормалізованих.

Із таблиці 3.3 можна зробити висновок про те, що довжини довірчих інтервалів в переважній більшості точок менші для нормалізованих даних, чим для вихідних.

Таблиця 3.3 – Довірчі інтервали для вихідних даних, нормалізованих за допомогою десяткового логарифму, та для ненормалізованих

№ п / п	Нижня границя довірчого інтервалу (ненорм)	Верхня границя довірчого інтервалу (ненорм)	Нижня границя довірчого інтервалу (норм)	Верхня границя довірчого інтервалу (норм)
1	58437,9836	76515,6288	54639,6677	85514,1488
2	28166,6821	44347,7380	26822,5042	39997,9101
3	121790,0326	137970,1126	81632,6636	169982,5901
4	31399,7509	47581,3289	31112,8349	44752,2980
5	140687,0590	156867,1197	87680,4720	193893,1753
6	36140,9956	52324,5185	36577,6776	51642,6645
7	47069,3673	63382,7899	46413,9358	67711,4264
8	43610,4023	59872,1327	43630,7708	62607,3745
9	37613,2532	53798,1838	38098,8152	53786,2009
10	39623,1755	55811,9608	40059,4427	56720,6106
11	73182,1280	89362,5659	62112,5121	105079,8801
12	54283,2379	70467,4225	51453,7718	78140,5340
13	31363,8289	47545,3992	31067,8153	44699,9522
14	57661,2730	73843,4542	53594,3855	83029,3445
15	22670,2386	38850,8600	18331,3835	31315,0140
16	49026,8615	65233,4398	47834,9350	70496,9528
17	18395,1442	34575,5932	10467,0311	22849,1719
18	36105,0829	52288,5794	36539,6114	51590,4316
19	40678,8373	66396,2865	45097,0754	65238,7150
20	86474,8185	102655,0321	68250,1319	123406,9377
21	17640,7118	33821,1383	8918,9820	20966,2755
22	30358,0026	46539,3758	29783,5885	43231,2422
23	33950,1121	50132,4437	34166,0821	48459,4221
24	42556,4737	58770,5081	42715,4051	61029,0081
25	20622,5098	36803,0376	14729,6240	27590,0336
26	21772,1134	37952,6908	16784,6476	29733,9716
27	28058,9108	44239,9540	26670,9319	39837,1147
28	29172,5420	45353,7278	28210,0353	41489,5492
29	38331,2625	54517,2099	38813,9086	54833,3273
30	38341,4542	68805,5213	45125,6903	65291,3397

В табл. 3.4 знаходяться результати виконаних розрахунків: інтервали прогнозування для вихідних даних, нормалізованих за допомогою десяткового логарифму та для ненормалізованих.

Таблиця 3.4 – Інтервали прогнозування для вихідних даних, нормалізованих за допомогою десяткового логарифму, та для ненормалізованих

№ п / п	Нижня границя інтервалу прогнозув. (ненорм)	Верхня границя інтервалу прогнозув. (ненорм)	Нижня границя інтервалу прогнозув. (норм)	Верхня границя інтервалу прогнозув. (норм)
1	21719,09	113234,52	32623,3298	143224,6401
2	-9609,77	82124,19	15715,2975	68267,5027
3	79030,50	180729,65	53878,3431	257545,6258
4	-6278,60	85259,68	17967,3225	77494,6218
5	95211,65	202342,53	58967,9149	288303,3112
6	-1428,81	89894,32	20963,4973	90107,5190
7	9648,29	100803,87	26956,8923	116584,4255
8	6162,05	97320,49	25171,7503	108518,7949
9	68,83	91342,61	21834,6130	93850,5542
10	2107,67	93327,47	22985,9279	98851,6130
11	34971,60	127573,09	38184,1501	170928,9141
12	16727,44	108023,22	30364,2752	132413,0142
13	-6315,50	85224,73	17943,3439	77395,2655
14	20037,32	111467,40	31873,9031	139609,7201
15	-15317,93	76839,03	11309,3358	50758,7308
16	11543,28	102717,03	27895,4937	120887,5236
17	-19796,50	72767,24	7011,4886	34110,1592
18	-1465,39	89859,05	20941,9323	90015,3002
19	7961,99	99113,13	26103,3581	112708,6880
20	47411,17	141718,68	42968,7639	196015,4078
21	-20590,33	72052,18	6109,2849	30608,7922
22	-7349,82	84247,19	17262,9791	74586,2879
23	-3664,67	87747,23	19618,1477	84394,7461
24	5079,10	96247,88	24600,2965	105969,4058
25	-17458,91	74884,46	9387,4411	43290,9048
26	-16256,00	75980,80	10491,1855	47570,8140
27	-9721,15	82020,01	15636,6416	67948,9242
28	-8571,29	83097,56	16437,5391	71204,1892
29	797,88	92050,60	22250,6924	95650,7652
30	7997,93	99149,05	26121,7363	112791,7663

Із таблиці 3.4 можна зробити висновок про те, що довжини інтервалів прогнозування в переважній більшості точок менші для нормалізованих даних, чим для вихідних. Також із таблиці 3.4 випливає, що границі інтервалу прогнозування більші нуля для нормалізованих даних, тоді як ті ж самі границі, отримані без використання нормалізації, мають від`ємні значення.

Для ненормалізованих та нормалізованих за методом десяткового логарифму даних були пораховані R^2, MMRE та PRED(25) [20, 21]. Для ненормалізованих даних: R^2 = 0,6824; MMRE = 0,3982; PRED(25) = 0,5. Для даних, нормалізованих за допомогою десяткового логарифму: R^2 = 0,7154; MMRE = 0,3484; PRED(25) = 0,5. Таким чином для нормалізованих даних R^2 більше на 0,333; MMRE менше на 0,0498, ніж для ненормалізованих даних, а PRED(25) співпадають, тобто удосконалена модель дає кращі результати [22].

Таким чином, удосконалена нелінійна регресійна модель має кращу якість в порівнянні з лінійною та може бути рекомендована для оцінювання розміру програмних застосунків на TypeScript.

3.3 Постановка задачі на розробку програмного забезпечення для оцінювання розміру програмних застосунків на TypeScript

Ретельно проаналізувавши існуючі методи оцінювання розміру програмних застосунків на TypeScript, можна сформулювати наступну постановку задачі: розробити проект програмного забезпечення для оцінювання розміру програмних застосунків на TypeScript. Для розробки програми слід побудувати математичну модель, яка допоможе оцінити розмір програмних застосунків на TypeScript за наявності негаусівських емпіричних даних. Нормалізація буде виконуватись за допомогою логарифмування (десятковий логарифм).

Дане програмне забезпечення повинно відповідати наступним вимогам. Вимоги до складу виконованих функцій:

1) внесення даних про проекти (кількість строк коду, класів);

2) виконання нормалізації внесених даних за допомогою перетворення на основі десяткового логарифму;

3) оцінювання розміру коду;

4) побудова довірчого інтервалу;

5) побудова інтервалу передбачення;

6) редагування даних.

Оцінювання розміру програмного забезпечення необхідно виконати для 30 проектів програмних застосунків на TypeScript, які були попередньо завершені.

Детальні вимоги до програми наведені у додатку А.

4 ПРОЕКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНИХ ЗАСТОСУНКІВ НА TYPESCRIPT

4.1 Ескізний проект програмного забезпечення для оцінювання розміру програмних застосунків на TypeScript

В ескізному проекті був проведений аналіз потоків вхідних і вихідних даних, розроблена загальна схема функціонування системи, уточнені вимоги до виконуваних функцій, виконане проектування інтерфейсу.

4.1.1 Розробка контекстної діаграми ПЗ для оцінювання розміру програмних застосунків на TypeScript

Для виявлення схеми функціональної структури задачі, визначення модулів, які входять до її складу і вивчення динаміки функціонування програмного забезпечення в цілому, необхідно побудувати функціональну модель.

Функціональна модель демонструє, яким чином вхідні дані перетворюються у вихідні, не розглядаючи порядок або спосіб реалізації перетворень. Функціональна модель складається з набору діаграм потоків даних, що показують потоки значень від зовнішніх входів через операції і внутрішні сховища даних до зовнішніх виходів.Важливу специфічну роль у моделі грає спеціальний вид діаграм потоків даних – контекстна діаграма, що моделює ПЗ найбільш загальним способом. Контекстна діаграма відбиває інтерфейс програмного забезпечення з зовнішнім світом [23].

Тепер перейдемо до програми, що проектуємо для оцінювання розміру програмних застосунків на TypeScript.

З опису предметної галузі та постановки задачі виходить, що потрібно розробити програмне забезпечення, яке буде автоматизувати: внесення даних

про проекти (кількість строк коду, класів); редагування даних, нормалізацію даних (десятковий логарифм); оцінювання розміру коду; розрахунок довірчого інтервалу та інтервалу прогнозування.

Таким чином задача зводиться до наступних дій:

1) розробити базу даних, в якій буде зберігатися інформація про вихідні дані та результати розрахунку.

2) розробити взаємодію з користувачем (тобто розробити прикладне програмне забезпечення, яке забезпечить доступ користувачів до інформації, що зберігається у БД та надасть можливість формувати відповідні документи).

Контекстна діаграма програмного забезпечення для оцінювання розміру програмних застосунків на TypeScript наведена на рисунку4.1.

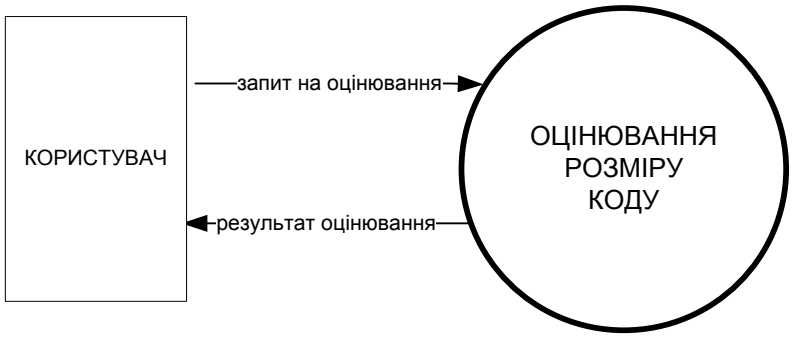

Рисунок 4.1 – Контекстна діаграма ПЗ для оцінювання розміру програмних застосунків на TypeScript

Як бачимо з рисунку 4.1, із системою взаємодіє «Користувач». «Користувач» посилає у контекстний процес інформацію у вигляді запиту на оцінювання, та отримує з контекстного процесу результат оцінювання.

Словник даних та потоки даних, що відповідають побудованій діаграмі наведені у таблиці 4.1.

Таблиця 4.1– Словник даних і потоки даних контекстної діаграми

Ім'я потоку	Структура потоку
Запит на оцінювання	– кількість класів
Результат оцінювання	– оцінка розміру коду; – довірчий інтервал (нижня та верхня межа довірчого інтервалу); – інтервал прогнозування (нижня та верхня межа інтервалу прогнозування).

4.1.2 Розробка концептуальної моделі даних ПЗ для оцінювання розміру програмних застосунків на TypeScript

Розробка концептуальної моделі включає наступні основні етапи:

– Ідентифікація сутностей та їх атрибутів.

– Ідентифікація відношень між сутностями і визначення типів відношень.

На основі аналізу предметної галузі були виділені наступні сутності та їх атрибути:

– Програмний застосунок: назва програмного застосунку на TypeScript;

– Дані: кількість класів;

– Результат: нижня границя довірчого інтервалу, верхня границя довірчого інтервалу, нижня границя інтервалу прогнозування, верхня границя інтервалу прогнозування, прогноз.

Відношення між сутностями та типи відношень приведені в таблиці 4.2.

Таблиця 4.2 – Відношення між сутностями та типи відношень

Зв’язок	Опис зв’язку	Відношення
Програмний застосунок – дані	Один програмний застосунок може мати одні дані, і одні дані можуть належати одному застосунку	Один до одного
Програмний застосунок – результат	Один програмний застосунок може мати один результат, і один результат може належати одному застосунку	Один до одного

На основі виділених сутностей, їх атрибутів а також даних таблиці 4.2 можемо побудувати діаграму «сутність-зв’язок» (рисунок 4.2).

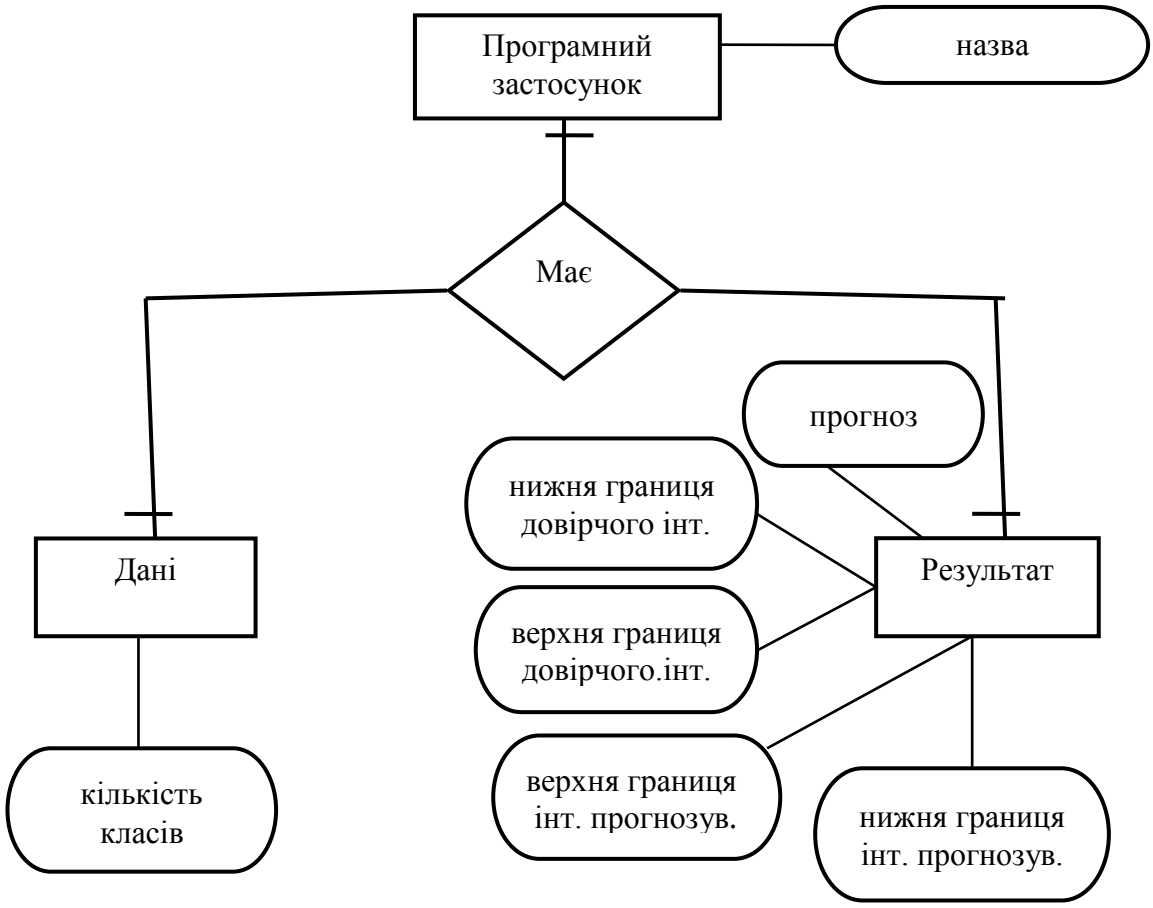

Рисунок 4.2 – Концептуальна модель даних ПЗ

4.1.3 Розробка діаграми станів ПЗ для оцінювання розміру програмних застосунків на TypeScript

Для представлення загального функціонування системи побудуємо діаграму переходів станів, на якій можна побачити, що система у заданий момент часу знаходиться в одному зі станів, та при виконанні тієї чи іншої умови та після виконання деякої дії, вона змінює свій стан, тобто переходить в інший стан [23]. ST -діаграма ПЗ представлена на рисунку 4.3.

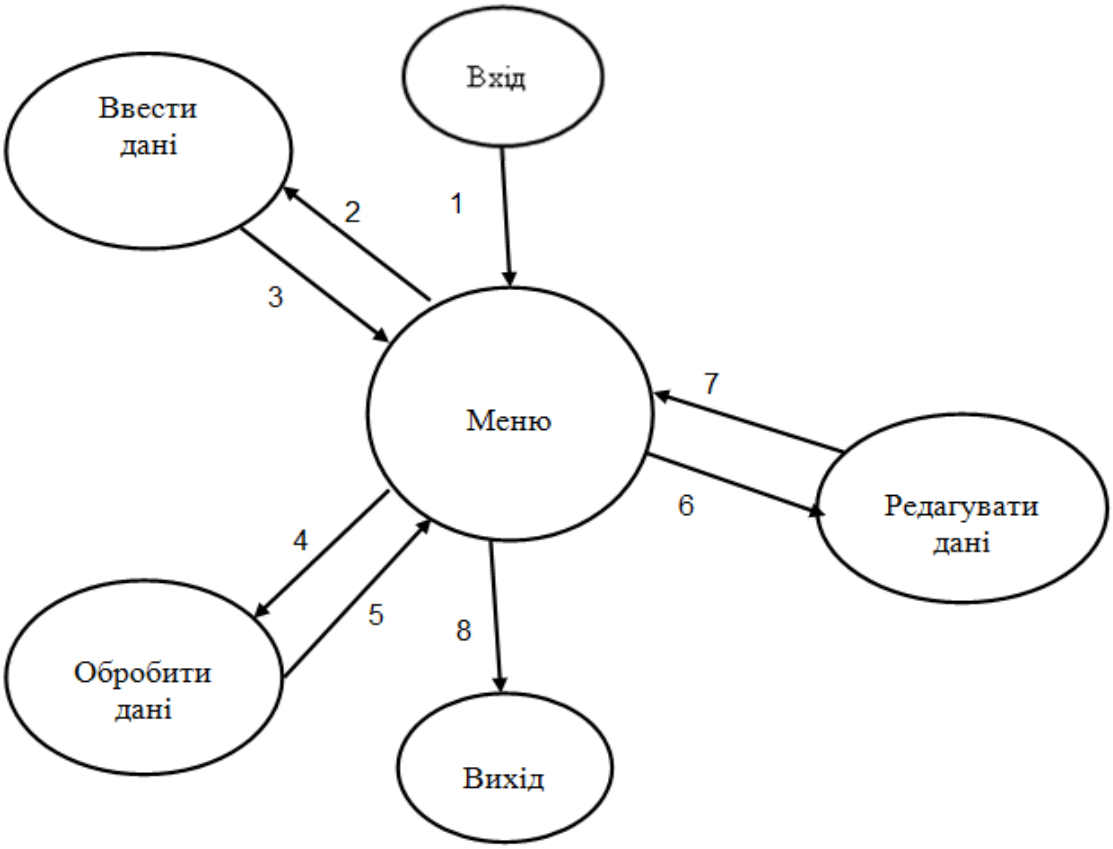

Рисунок 4.3 – Діаграма переходів станів ПЗ для оцінювання розміру програмних застосунків на TypeScript

Можливі події на переходах описані в таблиці 4.3.

Таблиця 4.3 – Можливі події на переходах

N	Події на переходах
1	початкова подія (завантаження програмного забезпечення)
2	вибір дії – введення початкових даних
4	вибір дії –обробка даних
6	вибір дії – редагування даних
3,5,7	
8	користувач завершив роботу з програмним забезпеченням

4.1.4 Розробка проекту графічного інтерфейсу ПЗ для оцінювання розміру програмних застосунків на TypeScript

Для розробки якісної і надійної програми варто приділити особливу увагу створенню екранних форм. Дані на користувальницькому екрані повинні виводитися в зручному для читання виді, їхнє розміщення повинне бути продумане і компактне. Створювані екранні форми повинні мати усі властивості, що характеризують поняття «дружній інтерфейс» [23]. Розглянемо проекти екранних форм для програмного забезпечення оцінювання розміру програмних застосунків на TypeScript.

Для входу в програму необхідно обрати відповідний пункт меню:

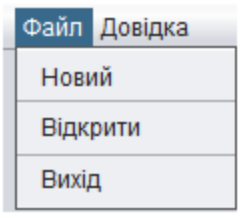

Рисунок 4.4 – Меню файл програми

Меню файл містить наступні пункти:

– завести нові дані;

– відкрити існуючі дані;

– вийти з програми.

При виборі можливості завести нові дані відкриється наступне вікно (рис. 4.5):

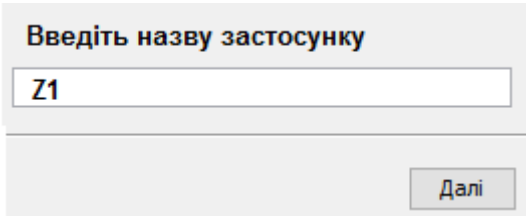

Рисунок 4.5 – Вікно вводу назви застосунку

Після натискання на кнопку «Далі» відкриється вікно для введення параметрів застосунку: кількості класів (рис. 4.6):

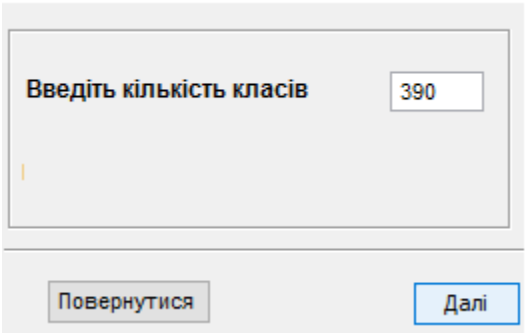

Рисунок 4.6 – Вікно вводу параметрів застосунку

На цьому кроці можна повернутися на попередній крок, або перейти до наступного. Після натискання на кнопку «Далі» користувач отримує інформацію про введені дані та може повернутися на початок, або натиснути «Далі» та отримати результат прогнозу (рис. 4.7):

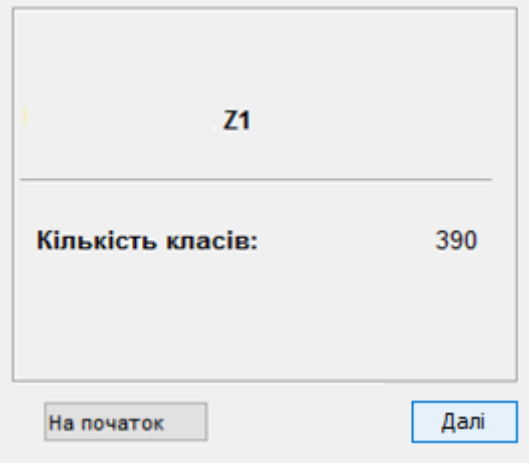

Рисунок 4.7 – Вікно перегляду інформації про введені дані

Результат прогнозування розміру застосунків на TypeScript, довірчий інтервал та інтервал прогнозування буде виводитися у вікні, проект якого представлений на рисунку 4.8.

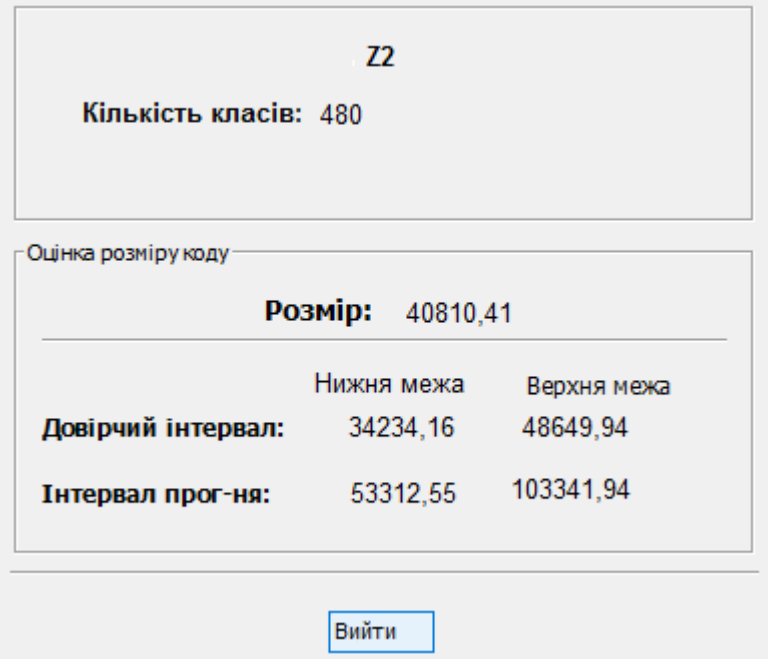

Рисунок 4.8 – Вікно перегляду прогнозного значення розміру застосунку на TypeScript

4.2 Технічний проект програмного забезпечення для оцінювання розміру програмних застосунків на TypeScript

У технічному проекті були визначені процеси і структури даних, необхідні для виконання функцій, побудована логічна модель даних, яка відображає структури представлення даних в програмному забезпеченні для оцінювання розміру програмних застосунків на TypeScript.

4.2.1 Побудова діаграми потоків даних першого рівня для ПЗ оцінювання розміру програмних застосунків

Діаграми потоків даних необхідні для відображення всіх процесів, що відбуваються з даними.

Загальний процес «Оцінювання розміру коду» необхідно розглядати як сукупність трьох більш детальних процесів: «1 Ввести дані», «2 Редагувати дані», «3 Обробити дані». Інформаційні потоки потрібно теж конкретизувати. На рисунку 4.9 представлена діаграма потоків даних першого рівня, яка дає основне уявлення про функціонування системи.

Відповідність потоків даних контекстної діаграми та діаграми першого рівня представлена в таблиці 4.4.

Таблиця 4.4 – Відповідність потоків даних контекстної діаграми та діаграми першого рівня

Потоки даних контекстної діаграми	Потоки даних DFD першого рівня
запит на оцінювання	запит на введення даних
	запит на редагування даних
	запит на обробку даних
результат оцінювання	підтвердження вводу
	підтвердження редагування
	підтвердження обробки

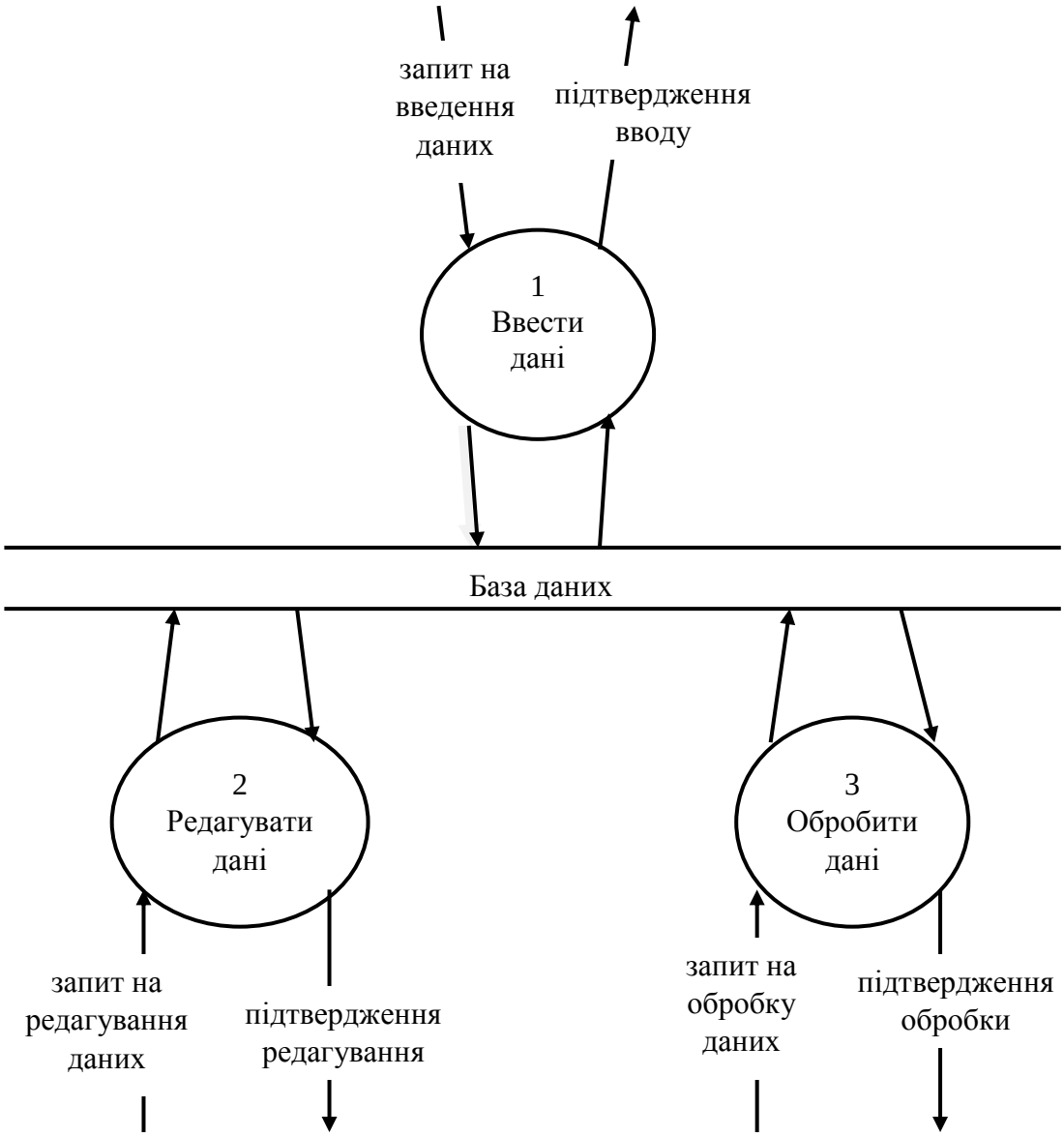

Рисунок 4.9 – Діаграма потоків даних першого рівня ПЗ оцінювання розміру програмних застосунків

Процес 3 «Обробити дані» на діаграмі потоків даних першого рівня являється складним та потребує подальшої декомпозиції.

4.2.2 Декомпозиція процесів до діаграми потоків даних другого рівня для ПЗ оцінювання розміру програмних застосунків

Процес 3 «Обробити дані», представлений на діаграми потоків даних першого рівня, деталізується до діаграми потоків даних другого рівня процесами: «Нормалізувати дані», «Апроксимувати дані», «Знайти прогноз», «Знайти інтервал передбачення», «Знайти довірчий інтервал»(рис. 4.10):

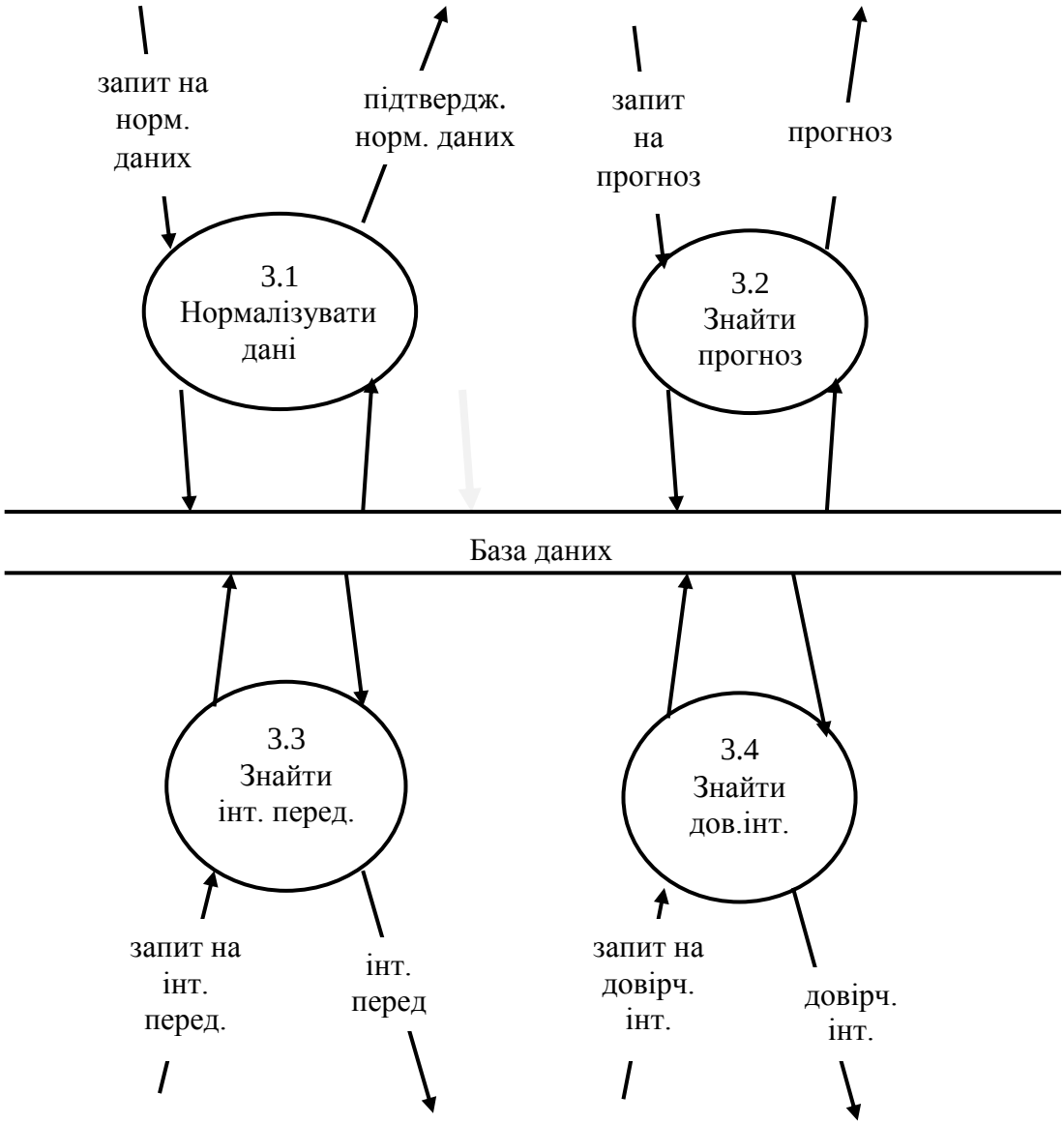

Рисунок 4.10 Декомпозиція процесу 3 «Обробити дані»

Декомпозиція процесів на діаграмах потоків даних доведена до рівня специфікацій програмних модулів.

Відповідність потоків на діаграмах першого та другого рівнів представлена в таблиці 4.5.

Таблиця 4.5 – Відповідність потоків на діаграмах першого та другого рівнів

Потоки даних DFD першого рівня	Потоки даних DFD другого рівня
запит на обробку даних	запит на нормалізацію даних
	запит на прогноз
	запит на інтервал передбачення
	запит на довірчий інтервал
підтвердження обробки	підтвердження нормалізації
	прогноз
	довірчий інтервал
	інтервал передбачення

.

4.2.3 Специфікація модулів програмного забезпечення

№ 1 Ввести дані

Призначення: Введення вихідних даних про програмний застосунок на TypeScript до бази даних.

Вхідні дані: Запит на введення вихідних даних.

Вихідні дані: Вихідні дані, підтвердження вводу.

Умови виконання: Занесення у форму даних правильного формату й натискання кнопки «Далі»

Процедура обробки: При введені даних правильного формату відбувається присвоєння унікального номеру запису та занесення інформації до таблиць «Програмний застосунок» та «Дані» бази даних.

№ 2 Редагувати дані

Призначення: Зміна вихідних даних про застосунок

Вхідні дані: Вихідні дані, запит на редагування.

Вихідні дані: Результати редагування, підтвердження редагування.

Умови виконання: Занесення у форму даних правильного формату й натискання кнопки «Вийти»

Процедура обробки: Прочитати вихідні дані про застосунок із бази даних. При введені оновлених даних правильного формату відбувається оновлення інформації у таблицях «Програмний застосунок», «Дані», «Результат» бази даних.

№ 3.1 Нормалізувати дані

Призначення: Нормалізація вихідних даних.

Вхідні дані: Вихідні дані.

Вихідні дані: Нормалізовані дані.

Умови виконання: Наявність емпіричних даних правильного формату.

Процедура обробки: Відбувається нормалізація даних за допомогою десяткового логарифму.

№ 3.2 Знайти прогноз

Призначення: Розрахунок прогнозу розміру застосунку.

Вхідні дані: Вихідні дані.

Вихідні дані: прогнозне значення розміру застосунку.

Умови виконання: Наявність вихідних даних правильного формату.

Процедура обробки: Відбувається розрахунок розміру програмного застосунку на TypeScript за допомогою нелінійного регресійного рівняння.

№ 3.3 Розрахувати інтервал прогнозування

Призначення: Розрахунок інтервалу прогнозування.

Вхідні дані: Нормалізовані дані, розмір застосунку.

Вихідні дані: Інтервал прогнозування.

Умови виконання: Наявність нормалізованих даних та інформації про розмір застосунку.

Процедура обробки: Відбувається розрахунок інтервалу прогнозування за допомогою формули (3.4). Інформація про інтервал прогнозування заноситься до таблиці «Результат» бази даних.

№ 3.4 Розрахувати довірчий інтервал

Призначення: Розрахунок довірчого інтервалу.

Вхідні дані: Нормалізовані дані, розмір застосунку.

Вихідні дані: Довірчий інтервал.

Умови виконання: Наявність нормалізованих даних та інформації про розмір застосунку.

Процедура обробки: Відбувається розрахунок довірчого інтервалу за допомогою формули (3.3). Інформація про довірчий інтервал заноситься до таблиці «Результат» бази даних.

4.2.4 Розробка логічної моделі даних

На основі концептуальної моделі даних, що була отримана у ескізному проекті, розробимо логічну модель бази даних. Логічна модель відображає логічні зв'язки між елементами даних поза залежністю від їхнього змісту й середовища зберігання. До логічного проектування відноситься опис характеристик тих об'єктів, відомості про які будуть накопичуватися та використовуватися в програмному середовищі [23].

У реляційній моделі бази дані об'єкти представляються у вигляді таблиць. Кожна таблиця представляє один об'єкт і складається з рядків і стовпців. Атрибутом у таблиці є будь-який стовпець, а записом (кортежем) – рядок у таблиці. Ключем є стовпець, значення якого однозначно визначає рядок таблиці [23].

Кожна сутність має ключове поле, що являється первинним ключем, або ж ключове поле є частиною складеного первинного ключа.

Логічна модель даних програмного забезпечення наведена на рис. 4.11.

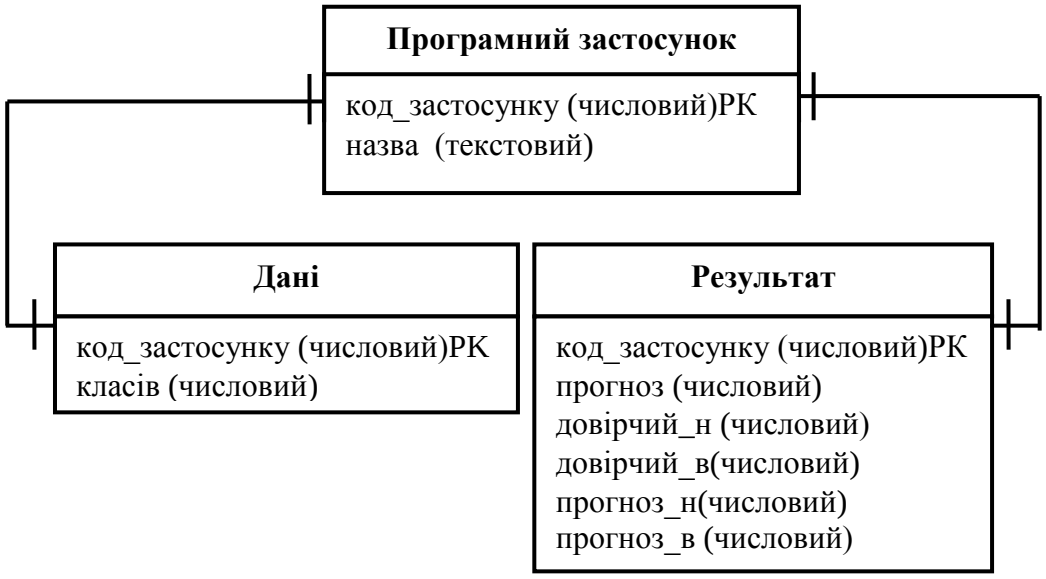

Рисунок 4.11 – Логічна модель даних

Таким чином, логічна модель даних містить три таблиці, які пов’язані між собою зв’язками один до одного.

4.3 Робочий проект програмного забезпечення для оцінювання розміру програмних застосунків на TypeScript

На етапі робочого проектування були розподілені та уточнені елементи інструментальних засобів, якими треба буде користуватись при розробці програмного продукту [23]. З їх допомогою розроблене програмне для оцінювання розміру програмних застосунків на TypeScript

4.3.1 Вибір засобів розробки програмного забезпечення для оцінювання розміру програмних застосунків

В якості мови програмування була обрана Java. Вибір мови Java був обумовлений наступними причинами:

– Багато якісного і безкоштовного інструментарію.

– Багато якісних бібліотек та фреймворків – як старих, перевірених часом, так і нових, що реалізують модні віяння.

– Дуже багато прикладів, книг, відео доповідей і так далі.

– Кросплатформеність.

– Статична типізація.

– Надійна платформа з точки зору зростання проекту.

В якості СКБД обрана MySQL. Ця СКБД є сучасною, поширеною, безкоштовною, добре співпрацює з багатьма мовами програмування.

MySQL – компактний багатопоточний сервер баз даних. Серед переваг сервера MySQL виділимо простоту у встановленні та використанні, підтримку паралельної роботи значної кількість користувачів. Кількість рядків у таблицях даних може сягати 50 млн. Висока швидкість виконання команд, наявність простої та ефективної системи безпеки обумовлюють зростання популярності MySQL. Проте основною перевагою СКБД MySQL є можливість її безкоштовного використання [23].

4.3.2 Розробка фізичної моделі даних для ПЗ програмного забезпечення для оцінювання розміру програмних застосунків

Фізична модель даних представлена у наведених в цьому розділі таблицях 4.6 - 4.8. Кожна таблиця містить назву полів, типи даних, довжину, формат і ключі. Поля та їх типи зображені згідно обраного середовища розробки програми. Таблиці містить п’ять колонок: № п/п, назву полів, тип даних (довжина), опис, ключ.

Таблиця 4.6 – Структура таблиці project (Програмний застосунок)

№ п/п	назва поля	Тип даних	Опис	Ключ
1	id_project	INT	Код застосунку	PK
2	title_project	CHAR(36)	Назва застосунку	

Таблиця 4.7 – Структура таблиці pdanni (Дані)

№ з/п	Ідентифікатор	Тип даних	Описання	Ключ
1	id_pr_danni	INT	Код застосунку	PK
2	class_danni	REAL	Кількість класів в застосунку	

Таблиця 4.8 – Структура таблиці resultat (Результати)

№ з/п	Ідентифікатор	Тип даних	Описання	Ключ
1	id_pr_resultat	INT	Код застосунку	РК
2	size_resultat	INT	Прогноз	
3	n_dov_resultat	REAL	Нижня межа довірчого інтервалу	
4	v_dov_resultat	REAL	Верхня межа довірчого інтервалу	
5	n_prog_resultat	REAL	Нижня межа інтервалу прогнозування	
6	v_prog_resultat	REAL	Верхня межа інтервалу прогнозування	

4.3.3 Реалізація та тестування програмного забезпечення для оцінювання розміру програмних застосунків на TypeScript

Текст програмного забезпечення наведений в додатку Д.

Найбільшої популярності набули дві групи методів тестування програмного забезпечення.

Перша група називається методами «чорного ящика» та називається тестуванням за специфікаціями. Ця група методів включає метод еквівалентного розбиття, метод граничних значень та інші. В роботі використовується метод еквівалентного розбиття. Згідно методу всі вхідні дані розбиваються на класи еквівалентності і на кожен клас обирається один тест. При запуску довільного тесту з класу еквівалентності програма веде

себе однаково. Це дає змосу зменшити кількість тестів. Інакше необхідно було б запускати стільки тестів, скільки є різних вхідних даних [23].

При використанні методів білого ящика досліджується внутрішня структура програми. До методів білого ящика відносяться метод покриття операторів, метод покриття умов, комбінований метод покриття умов та інші [23].

Програма була протестована одним із методів чорного ящика та помилок знайдено не було. Розглянемо тестування функцій програмного забезпечення методом розбиття на класи еквівалентності. В роботі наводиться приклад тестування функції «Ввести дані.

Тестування функції «Ввести дані»

Класи еквівалентності для функції «Ввести дані» представлені в таблиці 4.9.

Таблиця 4.9 – Класи еквівалентності вхідних даних для функції «Ввести дані»

Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
Назва застосунку	1.Текст	2.Пуста строка
Кількість класів	3. Число >0	4. Число<= 0 5. Не число

В таблиці 4.10 наведені тести для правильних класів еквівалентності.

Таблиця 4.10 – Тести для правильних класів еквівалентності

№ тесту	№ покритого класу	Вхідні дані	Вихідні дані
1	1	Z5	Система продовжила роботу по введенню даних. Відсутні повідомлення про помилку. Результат вірний.
	3	10	

В таблиці 4.11 наведені тести з неправильних класів еквівалентності.

Таблиця 4.11 – Тести з неправильних класів еквівалентності для функції «Ввести дані»

№ тесту	№ покритого класу	Вхідні дані	Вихідні дані
1	2	Пуста строка	Повідомлення про невірну назву застосунку
	4	-5	Повідомлення про невірну кількість класів
	5	ae	Повідомлення про невірний формат даних

4.3.4 Випробування програмного забезпечення для оцінювання розміру програмних застосунків на TypeScript

Випробування програмного забезпечення є завершальним етапом розробки й одним з найважливіших етапів життєвого циклу, на якому перевіряються й фіксуються досягнуті показники якості програмного забезпечення. Метою випробувань є експериментальне визначення

фактичних (досягнутих) характеристик випробовуваного програмного забезпечення й визначення ступеня його відповідності технічному завданню.

Програму та методику випробувань можна знайти в Додатку Б. На основі інструкцій, що наведені в цьому додатку провели випробування розробленого програмного забезпечення:

– Функція «Введення даних»

Після вибору в меню «Файл» пункту «Новий» отримали можливість ввести назву застосунку, кількість класів. Ввели необхідні дані. Інформація про новий застосунок записалася до таблиць «Програмний застосунок» та «Дані». Результат відповідає очікуваному. Проблем не виникло.

– Функція «Редагувати дані»

Після вибору в меню «Файл» пункту «Відкрити» обрали потрібний застосунок та отримали можливість змінити назву застосунку, кількість класів. Ввели необхідні дані. Натиснули «Так». Змінена інформація про застосунок записалася до таблиць «Програмний застосунок», «Дані» та «Результат». Результат відповідає очікуваному. Проблем не виникло.

5 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНИХ ЗАСТОСУНКІВ НА TYPESCRIPT

Під час виконання кваліфікаційної роботи роботи були поставлені задачі, які було виконано у повному обсязі:

– виконано порівняльний аналіз існуючих методів та моделей оцінювання розміру ПЗ;

– обґрунтована необхідність удосконалення регресійної моделі для оцінювання розміру програмних застосунків на TypeScript.

– зібрані дані, які можуть бути використані для удосконалення моделі;

– виконана нормалізація емпіричних даних, використовуючи перетворення у вигляді десяткового логарифму;

– перевірені нормалізовані дані на викиди;

– побудовані лінійна регресійну модель, довірчий інтервал та інтервал прогнозування для нормалізованих даних;

– побудована нелінійна регресійна модель, довірчий інтервал та інтервал прогнозування для вихідних даних за допомогою зворотнього перетворення;

– побудувані лінійна регресійна модель, довірчий інтервал і інтервал прогнозування для вихідних даних без нормалізації в припущенні про нормальність вихідних даних;

– виконано порівняння отриманих лінійної та нелінійної моделей;

– розроблене ПЗ для оцінювання розміру програмних застосунків на TypeScript.

За результатами наведених розрахунків можна стверджувати, про доцільність побудови нелінійної регресійної моделі для вихідних даних.

Також була розроблена наступна документація:

– технічне завдання (додаток А);

- програма і методика випробувань програмного забезпечення (додаток Б);
- опис програми (додаток В);
- інструкція користувача (див. додаток Г).

6 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВІД РОЗРОБКИ ТА ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

6.1 Вступ

Дана робота присвячена розробці програми для оцінювання розміру програмних застосунків на TypeScript.

Ефективність програмних систем визначається ступенем їх позитивного впливу на підприємство, що управляється, або процес в результаті використання засобів обчислювальної техніки, програмного забезпечення, зміни інформаційних потоків та організаційної структури управління.

Створення програмного забезпечення вимагає одноразових витрат на розробку, придбання необхідних технічних засобів і поточних витрат на функціонування продукту. Економія від функціонування ПЗ визначається з урахуванням витрат на його експлуатацію. Відношення цієї економії до витрат на створення програмного забезпечення характеризує економічну ефективність капітальних вкладень. Економічні показники визначаються за діючими на момент розрахунку оптовими цінами, тарифами і ставками заробітної плати.

6.2 Розрахунок витрат на створення й експлуатацію програми для оцінювання розміру програмних застосунків на TypeScript

Витрати на розробку системи складаються з витрат:

- на заробітну платню розробника;
- на амортизацію ЕОМ, на якій виконується розробка;
- на експлуатацію цієї ЕОМ;

- на засоби розробки;

- на матеріали і комплектуючі.

Розробка програмного забезпечення виконується програмістом, місячна заробітна плата якого складає 10000 грн.

Вартість сучасного ПК становить 25000 грн.

При вартості кіловат-години електроенергії 1.68 грн, розраховується вартість розробки програми.

Витрати на допоміжні матеріали наведені в табл. 6.1.

Таблиця 6.1 – Витрати на допоміжні матеріали

№	Пункти витрат	Сума, грн
1	Флеш-накопичувач даних	320
2	Папір для принтера	90
3	Картридж та тонер для принтера	240
	Разом	650

Вартість ПЗ знаходиться за формулою:

$$C_{пр} = (З_{зп} + З_{сз} + З_{зг} + З_{е}) * Т + З_{м} \quad (6.1)$$

де Т – тривалість розробки, міс.

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

$З_{сз}$ – відрахування на соціальні заходи (15% від основної і додаткової заробітної плати), грн.

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

$З_{е}$ – витрати на електроенергію, грн.

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

$З_е$ при споживанні потужності 0.5 кВт, тривалості роботи на місяць, рівної 22*8 = 176 годин, вартості кіловат-години електроенергії 1.68 грн становить:

$$З_е = 176 * 1.68 * 0.5 = 113.52 \text{ грн}$$

Витрати на розробку ПЗ наведені в табл. 6.2.

Таблиця 6.2 – Витрати на розробку програмного забезпечення

№	Найменування витрат	Одиниця	Кількість
1	Тривалість розробки (Т)	Міс.	3
2	Основна і додаткова заробітна плата ($З_{зп}$)	Грн.	10000.00
3	Відрахування на соціальні заходи ($З_{сз}$)	Грн.	1200.00
4	Загальногосподарські витрати ($З_{зг}$)	Грн.	900.00
5	Витрати на допоміжні матеріали ($З_м$)	Грн.	660.00
6	Витрати на електроенергію ($З_е$)	Грн.	113.52

За формулою 6.1 розрахуємо вартість програми:

$$С_{пр} = (10000 + 1200 + 900 + 113.52) * 3 + 660 = 37300.56 \text{ грн}$$

Амортизаційні відрахування складають 25% балансової вартості на рік:

$$А_{об} = 15000 * 0.25 = 3750 \text{ грн}$$

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

$$В_м = 15000 * 0.05 = 750 \text{ грн}$$

Річний обсяг робіт ПК у годинах визначається в такий спосіб:

$$Ф_м = 264.5 * ТЗ \qquad (6.2)$$

Тз – це середнє місячне навантаження устаткування (близько 6 годин), 264.5 – середня кількість робочих днів на рік.

Таким чином, річний обсяг роботи ПК складає:

$$Ф_м = 264.5 * 6 = 1587 \text{ годин}$$

Витрати на електроенергію $З_е$ при 1587 годинах роботи устаткування на рік становлять:

$$З_е = 1587 * 1.29 * 0.5 = 1023.62 \text{ грн}$$

Експлуатаційні витрати на ПК на рік складуть:

$$З_{зр} = 3750 + 750 + 1023.62 = 5523.62 \text{ грн}$$

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

$$З_р = 37300.56 + 5523.62 = 42824.18 \text{ грн}$$

6.3 Економічна ефективність розробки і впровадження програми для оцінювання розміру програмних застосунків на TypeScript

Основним показником економічної ефективності програмного забезпечення є зменшення трудових витрат та часу на оцінювання розміру програмного забезпечення, а також ефективне розподілення трудових ресурсів на основі отриманих результатів. Ефективний розподіл ресурсів дає змогу зменшити час оцінювання, і, відповідно, грошові витрати на роботу спеціалістів.

Визначимо пряму економічну ефективність, ґрунтуючись на тому, що впровадження програмного продукту вивільняє 1.5 працівника (за експертною оцінкою фахівців підприємства).

Заробітна плата 1.5 працівника в рік складає:

$$З = 10000 * 12 * 1.5 = 180000 \text{ грн}$$

Річний економічний ефект розраховується за формулою:

$$E_{рік} = \Delta C_п – E_п * k \quad (6.3)$$

де $\Delta C_п$ – вивільнені кошти після впровадження продукту (180000 грн) мінус експлуатаційні витрати (42824.18 грн) = 137175.82 грн;

$E_п$ – коефіцієнт ефективності (дорівнює коефіцієнту амортизації 0.25);

k – одноразові витрати на впровадження продукту (37300.56 грн).

$E_{рік} = 137175.82– 0.25 * 37300.56 = 127850.68$ грн

Термін окупності ПЗ розраховується за формулою:

$$T = k / \Delta C_п = 37300.56 / 137175.82 = 0.27 \text{ року}$$

Отже, термін окупності ПЗ становить приблизно 4 місяці.

6.4 Висновки

У цьому розділі були здійснені розрахунки витрат, економічної ефективності та терміну окупності ПЗ для оцінювання розміру програмних застосунків на TypeScript.

Виходячи з розрахунків, впровадження даного програмного забезпечення окупить себе протягом 4 місяців. Таким чином, його розробка є економічно обґрунтованою.

7 ОХОРОНА ПРАЦІ

7.1 Вступ

Охорона праці – це така система, що піклується про збереження життя та здоров’я працівників у ході їх трудової діяльності через правові, соціально-економічні, організаційно-технічні, санітарно-гігієнічні, лікувально-профілактичні, реабілітаційні та інші заходи.

Законодавство про працю містить норми і вимоги з техніки безпеки і виробничої санітарії, норми, що регулюють робочий час і час відпочинку, звільнення та переведення на іншу роботу, норми праці щодо жінок, молоді, гігієнічні норми і правила тощо.

Загальний нагляд за додержанням норм охорони праці покладено на прокуратуру, спеціальний – на професійні спілки. Контроль за безпекою праці здійснюють також державні й відомчі спеціалізовані інспекції.

Комп’ютерна техніка дуже широко застосовується у житті людини: вдома, на роботі та відпочинку, у різноманітних організаціях та виробництвах – інакше кажучи, комп'ютери міцно увійшли до повсякденного життя людей та масштаби цієї інтеграції з часом тільки збільшуються.

Недотримання вимог безпеки та експлуатації приводить до того, що з часом оператори комп’ютерної техніки починають відчувати перевтому та загальне погіршення стану здоров’я, що проявляється у погіршенні зору, головних болях, проблемах з хребтом, шиєю та кистями рук.

7.2 Аналіз шкідливих та небезпечних факторів на робочому місці програміста

Недостатн’є освітлення

Освітлення – використання світлової енергії Сонця та штучних джерел світла для забезпечення можливості безперешкодного зорового сприйняття оточуючого середовища.

Світло є необхідною умовою житт’єдіяльності людини, воно потрібно для збереження здоров’я та високої продуктивності праці, що виявляється у якості проведених робіт та виробленої продукції.

Виступаючи подразником не тільки для органів зору, а і для всього організму, достатн’є освітлення діє тонізуючи, покращує діяльність нервової системи та емоційний стан, настрій, впливає на формування добового ритму фізіологічних процесів організму людини.

Недостатн’є освітлення приміщень та робочих поверхонь, навпаки, може потягнути за собою цілу купу негативних наслідків, котрі у результаті негативно позначаться на здоров’ї працівників та результатах їх праці. До впливів на здоров’я слід віднести такі прояви, як: втрата зору, головні болі, психологічні розлади тощо.

Згідно СНіП II-4-79, в приміщенні зі встановленими комп’ютерами, необхідно застосовувати систему комбінованого освітлення. У таких приміщеннях висуваються наступні вимоги щодо освітленості: при виконанні зорових робіт високої точності загальна освітленість повинна складати 300лк, а комбінована –750лк; аналогічні вимоги при виконанні робіт середньої точності - 200 і 300лк відповідно. Крім того усе поле зору повинне бути освітлено максимально рівномірно – це основна гігієнічна вимога. Іншими словами, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими, тому що яскраве світло у районі периферійного зору значно збільшує напруженість очей і, як наслідок,

призводить до швидкого стомлення очей, що є однією з головних причин погіршення зору.

Параметри мікроклімату

Параметри мікроклімату можуть змінюватись у широких межах, в той час як необхідною умовою життєдіяльності людини є підтримка сталості температури тіла завдяки терморегуляції, тобто здатності організму регулювати власну температуру через віддачу тепла в навколишнє середовище. Принцип нормування мікроклімату – створення оптимальних умов для теплообміну тіла людини з оточуючим середовищем, що зроблять знаходження людей у приміщені комфортним та приємним.

Обчислювальна техніка є джерелом істотних тепловиділень, так, температура центрального процесора під час роботи може перевищувати 60°С, графічного ядра – більше ніж 50°С, тому для охолодження вузлів комп'ютера використовуються повітряні або водяні системи охолодження, що відводять тепло від елементів системи, розсіюючи його у навколишнє середовище. Такий вплив на мікроклімат приміщення може призвести до підвищення температури і зниження відносної вологості в приміщенні. У санітарних нормах СН-245-71 встановлені величини параметрів мікроклімату, що створюють комфортні умови для більшості працівників. Ці норми встановлюються залежно від пори року, характеру трудового процесу і характеру виробничого приміщення (дивись таблицю 7.1).

Таблиця 7.1 – Параметри мікроклімату для приміщень, де встановлені комп'ютер.

Період року	Параметр мікроклімату	Величина
Холодний	Температура повітря в приміщенні Відносна вологість Швидкість руху повітря	22…24°С 40…60% до 0,1м/с
Теплий	Температура повітря в приміщенні Відносна вологість Швидкість руху повітря	23…25°С 40…60% 0,1…0,2м/с

На кожну людину, що працює у приміщені, об’єм цього приміщення повинен складати не менше ніж 19,5 $м^3$ на особу з урахуванням максимального числа одночасно працюючих у зміну. Норми подачі свіжого повітря в приміщення, де розташовані комп'ютери, приведені в таблиці 7.2.

Таблиця 7.2 – Норми подачі свіжого повітря в приміщення, де розташовані комп’ютери.

Характеристика приміщення	Об'ємна витрата свіжого повітря, що подається в приміщенні м3 /на одну людину в годину
Обсяг до 20$м^3$ на людину	Не менше 30
20…40$м^3$ на людину	Не менше 20
Більше 40$м^3$ на людину	Природна вентиляція

Шум і випромінювання

Шум – це сукупність звуків різної частоти та інтенсивності, що виникають у результаті коливального руху частинок у пружних, тобто шумом можна назвати звуки котрі не не суть ніякого корисного інформаційного навантаження та своїм існуванням заважать працювати та концентруватися на певних задачах.

Шум погіршує умови праці, роблячи шкідливий вплив на організм людини: працюючи в умовах тривалого шумового впливу люди відчувають дратівливість, головний біль, запаморочення, нервову втомленість, крім того, шум значно зменшує об’єм засвоєної інформації та підвищує кількість зроблених помилок під час виконання роботи порушуючи концентрацію.

Тривала дія інтенсивного шуму, рівень якого перевищую 80 дБА, на слух людини приводить до суттєвої втрати працеспроможності.

Рівень шуму на робочому місці не повинен перевищувати 50 дБА. Для зниження рівня шуму стіни і стеля приміщень, де встановлені комп'ютери, можуть бути облицьовані звукопоглинальними матеріалами, а самі робочі машини слід або ізолювати від працівників у інше приміщення, або використовувати якісні корпуса, що самі собі гарно ізолюють працюючі

елементи системи від середовища, виступають бар’єром, та значно зменшують негативні впливи на людей.

Вважається, що як короткочасний, так і тривалий вплив усіх видів випромінювання від екрану монітора або системного блоку безпечно для людини. Проте вичерпних даних щодо небезпеки дії випромінювання на працюючих з комп’ютерами не існує.

Максимальний рівень рентгенівського випромінювання на робочому місці оператора комп'ютера звичайно не перевищує 10мкбер/год, а інтенсивність ультрафіолетового і інфрачервоного випромінювань від екрану монітора лежить в межах 10…100мВт/м^2. Для зниження дії цих видів випромінювання рекомендується застосовувати якісні монітори та іншу периферійну техніку, а також дотримуватися регламенту періодів праці та відпочинку. Слід обмежити тривалість робочої діяльності біля екрана, не розміщувати дисплеї концентровано у робочій зоні, вимикати монітори, якщо за ним не працюють.

Ергономічні вимоги до робочого місця

Проектування робочих місць належить до важливих проблем ергономічного проектування у області обчислювальної техніки.

Робоче місце і взаємне розташування всіх його елементів повинне відповідати антропометричним, фізичним і психологічним вимогам. Велике значення має також характер роботи. Зокрема, при організації робочого місця співробітника повинні бути дотримані наступні основні умови: оптимальне розміщення обладнання, достатній робочий простір, що дозволяє здійснювати всі необхідні рухи і переміщення.

Ергономічними аспектами проектування робочих місць, зокрема, є: висота робочої поверхні, розміри простору для ніг, вимоги до розташування документів на робочому, характеристики робочого крісла, вимоги до поверхні робочого столу, можливість регулювання елементів робочого місця.

Головними елементами робочого місця співробітника є робочий стіл і крісло. Основним робочим положенням є положення сидячи.

Робоча поза сидячи викликає мінімальне стомлення співробітника(але водночас може привести до скаліозу через постійну, ледве помітну перевтому, що може виникати при порушені режиму праці та відпочинку, та часто ігнорується співробітниками). Раціональна планування робочого місця передбачає чіткий порядок і сталість розміщення предметів, засобів праці і документації. Те, що потрібно для виконання робіт частіше, має бути розташоване у зоні легкої досяжності робочого простору.

Оптимальне розміщення предметів праці і документації в зонах досяжності представлено на рисунку 7.1.

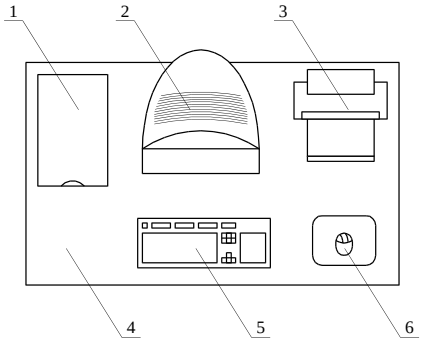

Рисунок 7.1 – Розміщення основних складових робочого місця: 1 – сканер, 2 – монітор, 3 – принтер, 4 – поверхня робочого столу, 5 –клавіатура, 6 – маніпулятор типу «миша».

Для комфортної роботи оператора стіл повинен задовольняти наступним вимогам:

– така висота столу, щоб можна було зручно та вільно сидіти, спиратися на підлокітник якщо потрібно;

– вільне місце для ніг знизу;

– антиблікове покриття робочої поверхні столу;

– не менше трьох висувних ящиків.

Висота від підлоги до робочої поверхні повинна бути близько 680-760 мм. Висота поверхні, на яку встановлюється клавіатура – 650 мм.

Велике значення надається характеристикам робочого крісла. Так, висота сидіння над рівнем підлоги перебуває в межах 420-550 мм. Поверхня сидіння м'яка, передній край закруглений, а кут нахилу спинки – не фіксований, з можливістю регулювання.

Положення екрану визначається:

– відстанню від оператора до матриці (0,6 ... 0,7 м);

– кутом зчитування, напрямком погляду на 20° нижче горизонталі екрану.

Повинна також передбачатися можливість регулювання екрану:

– за висотою +/- 3 см;

– за нахилом від -10° до +20° по вертикалі;

Особливо важливою є поза користувача. При незручній робочій позі можуть з'явитися болі у м'язах, суглобах і сухожиллях, виникнути хронічні проблеми з поставою. Вимоги до робочої пози користувача наступні:

– голова не повинна бути нахилена більш ніж на 20°,

– плечі повинні бути розслаблені,

– лікті – під кутом 80° ... 100°,

– передпліччя і кисті рук – у горизонтальному положенні.

Причина неправильної пози користувачів обумовлена наступними чинниками: недостатній простір для ніг, не правильно відрегулюване кресло, монітор стоїть на неправильній відстані, стол завеликий або навпаки, робоча поверхня розташовано занадто низько.

Вагоме значення для продуктивної і якісної роботи на комп’ютері мають розміри знаків, густина їх розміщення, контраст і співвідношення яскравості символів і фону екрану. Якщо відстань від очей до екрана дисплея становить 60 ... 80 см, то висота знака повинна бути не менше 3 мм, оптимальне співвідношення ширини і висоти знака складає 3:4, а відстань між знаками – 15 ... 20% їх висоти. Співвідношення яскравості фону екрану і символів – від 1:2 до 1:15.

Під час користування комп’ютером медики радять встановлювати монітор на відстані 50-60 см від очей. Фахівці також вважають, що верхня частина дисплея повинна бути на рівні очей або трохи нижче. Коли людина дивиться прямо перед собою, його очі відкриваються ширше, ніж коли він дивиться вниз. За рахунок цього площа огляду значно збільшується, викликаючи обезводнення очей. До того ж, якщо екран встановлений високо, а очі широко відкриті, порушується функція моргання: очі не закриваються повністю, чи не омиваються слізної рідиною, не отримують достатнього зволоження, що приводить до їх швидкої стомлюваності.

Створення сприятливих умов праці і правильне естетичне оформлення робочих місць на виробництві має велике значення як для полегшення праці, так і для підвищення його привабливості, якісно впливає на продуктивність.

7.3 Техніка безпеки при роботі на комп’ютері

Загальні положення:

1) При виконанні робіт на персональних комп’ютерах необхідно дотримуватись вимог загальної та даної інструкцій з охорони праці.

2) До самостійної роботи на комп’ютерах допускаються особи, які пройшли медичний огляд, навчання з професії, ввідний інструктаж з охорони праці і первинний інструктаж з охорони праці на робочому місці. У подальшому вони проходять повторні інструктажі з охорони праці на робочому місці один раз на півріччя.

3) Основним обладнанням робочого місця користувача ПК є: монітор, системний блок, клавіатура, друкуючий пристрій (принтер), зовнішні пристрої пам’яті, сканер, "миша", блок безперервного живлення, робочий стіл, стілець та інше; допоміжним – шафи, полиці, сейф та ін.

Робочі місця повинні бути розташовані на відстані не менше 1,5 м від стіни з вікнами, від інших стін – на відстані 1м, між собою – на відстані не

менше 1,5 м. Відносно вікон робочі місця доцільно розташувати таким чином, щоб природне світло падало на нього збоку, переважно зліва.

5) Монітори слід розташовувати таким чином, щоб запобігти потраплянню в очі прямого світла. Джерела освітлення слід розташовувати з обох боків екрану, паралельно напрямку зору.

Для запобігання світлових відблисків від екрана, клавіатури в напрямку очей користувача від освітлювачів загального призначення або сонячних променів необхідно застосовувати спеціальні фільтри, захисні козирки, на вікнах регульовані жалюзі.

Фільтри із металевої або нейлонової сітки використовувати не рекомендується тому, що сітка спотворює зображення внаслідок інтерференції світла. Найкраща якість зображення забезпечується скляними поляризаційними фільтрами – вони усувають практично усі відблиски і забезпечують чітке та контрастне зображення.

6) При роботі з текстовою інформацією (у режимі введення даних, редагування тексту і читання з екрану) найбільш фізіологічним є зображення чорних знаків на світлому (білому) фоні.

7) Розташовувати монітор на робочому місці слід так, щоб поверхня екрану знаходилась у центрі поля зору на відстані 400-700 мм від очей користувача. Пропонується розташовувати елементи робочого місця таким чином, щоб підтримувалась однакова відстань від очей користувача до екрана, клавіатури, пюпітра.

8) Робота комп’ютера супроводжується електромагнітним випромінюванням різних частот і рівнів, інтенсивність котрого зменшується пропорційно квадрату відстані до екрану. Оптимальною для працюючого за ПК є відстань 50 см від екрану монітору.

9) Раціональною робочою позою працюючого за ПК вважається таке розташування тіла, при якому ступні працівника розташовані горизонтально на підлозі або на підставці для ніг, стегна зорієнтовані в горизонтальній площині, верхні кінцівки рук – вертикально, кут ліктьового суглоба

коливається в межах 70-90 град., зап'ястя зігнуті під кутом, не перевищує 20 град., нахил голови – у межах 15-20 град.

10) Для нейтралізації зарядів статичної електрики в приміщенні, де виконуються роботи на комп'ютерах, в тому числі на лазерних та струменевих принтерах, пропонується збільшувати вологість повітря за допомогою кімнатних зволожувачів. Не бажано носити одяг з синтетичних матеріалів.

Вимоги безпеки перед початком роботи

1) Перевірити надійність встановлення апаратури на робочому столі. Монітор не повинен стояти на краю столу. Повернути монітор треба так, щоб було зручно дивитись на екран – під прямим кутом (а не збоку) і трохи згори вниз, а екран повинен бути нахиленим під невеликим кутом (нижній край ближче до оператора).

2) Перевірити загальний стан апаратури, справність проводки електромережі, з'єднувальних шнурів, штепсельних вилок та розеток, заземлення захисного екрана.

4) Відрегулювати освітленість робочого місця.

5) Відрегулювати і зафіксувати висоту стільця (крісла.

6) Під'єднати до системного блока необхідну апаратуру (принтер, сканер і т.п.). Всі кабелі, що з'єднують системний блок з іншими пристроями слід вставляти і виймати тільки при вимкнутому електричному живленні.

7) Увімкнути апаратуру ПК вмикачами на корпусах в такій послідовності : стабілізатор (або блок безперервного живлення), монітор, системний блок, принтер (якщо передбачається друкування).

8) Відрегулювати яскравість світіння екрану, фокусировку, контрастність. Не слід робити яскравість екрану занадто великою, тому що це втомлює очі.

Рекомендується:

- яскравість світіння екрану – не менше 100 кд/ кв.м (кандел на кв.м);

- відношення яскравості екрану до яскравості оточуючих його поверхонь в робочій зоні – не більше 3:1;
- мінімальний розмір точки світіння – не більше 0,4 мм для монохромного екрану і не більше 0,6 мм для кольорового;
- контрастність зображення знака – не менше 0,8.

9) При появі несправностей комп'ютерної техніки повідомити керівника.

Вимоги безпеки при закінченні роботи на ПК

1) Закінчити і зберегти в пам'яті ПК файли, які знаходились у роботі. Виконати всі дії для коректного завершення роботи в оперативній системі.

2) Вимкнути принтер та інші периферійні пристрої, вимкнути монітор і системний блок. Вимкнути стабілізатор (або блок безперервного живлення при його наявності). Штепсельні вилки витягнути з розеток. Накрити клавіатуру кришкою для попередження потрапляння в неї пилу.

3) Навести порядок на робочому місці.

4) Вимкнути освітлення та електроприлади.

Вимоги безпеки при аварійних ситуаціях

1) При раптовому припиненні подачі електричної енергії вимкнути всі пристрої ПК в такій послідовності: периферійні пристрої, монітор, системний блок, стабілізатор (або блок безперервного живлення). Витягнути вилки з розеток.

2) При наявності ознак горіння (дим, запах горілого) необхідно вимкнути всі пристрої ПК, згідно з попереднім пунктом. Знайти місце загоряння і виконати всі можливі заходи для його ліквідації, попередивши терміново про це керівництво.

3) У випадку виникнення пожежі негайно попередити про це пожежну частину та керівництво, виконати усі можливі заходи щодо евакуації людей з приміщення і розпочати гасіння пожежі первинними засобами пожежогасіння.

8 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

8.1 Вступ

Комплекс екологоорієнтованих засобів щодо захисту навколишнього середовища охоплює заходи, спрямовані на охорону і раціональне використання природних ресурсів, і заходи, які забезпечують нормативні санітарно-гігієнічні параметри середовища міських і сільських поселень. Соціально необхідні охоронні заходи поділяються на організаційні, економічні і містобудівні.

Організаційні заходи забезпечують на законодавчому рівні використання територій, форми власності, правовий захист територій, створення системи адміністративно-господарського управління територіями та спеціальної екологічної служби їх охорони.

Економічні заходи забезпечують впровадження ресурсозберігаючих технологій, введення штрафних санкцій за порушення норм природокористування, визначення платежів і податків за використання територій, надання пільгових кредитів виробникам екологічно чистої продукції тощо.

Містобудівні заходи забезпечують охорону природного середовища за рахунок раціонального функціонального зонування території, створення санітарно-захисних зон, визначення територій природно-заповідного фонду, забезпечення екологічного балансу природно-ландшафтних та урбанізованих територій. Основні принципи екологічного захисту навколишнього середовища такі:

– збереження та раціональне використання цінних природних ресурсів;

– дотримання нормативів гранично допустимих рівнів екологічного навантаження на природне середовище та санітарних нормативів в місцях забудови;

– виділення природно-заповідних, ландшафтних, курортно-рекреаційних, історико-культурних зон з відповідним режимом їх охорони;

– встановлення санітарно-захисних зон для охорони водойм, джерел водопостачання і мінеральних вод, покладів лікувальних грязей, морських пляжів тощо.

Для захисту найбільш цінних елементів території навколишнього середовища вживаються заходи, спрямовані на заборону в їх межах, не властивої для них, містобудівної діяльності (крім будівництва об'єктів, що пов'язані з функціональною експлуатацією цих територій]. Це стосується природних заповідників, заказників, природних національних парків, водоохоронних зон, зелених зон міст, зон санітарної охорони курортів.

Не допускається містобудівна діяльність на площах залягання корисних копалин (до погодження з органами державного гірничого нагляду], в районах розміщення породних відвалів вугільних шахт (ближче 200-500 м залежно від характеристик терикону], на земельних ділянках, забруднених органічними і радіоактивними відходами, у небезпечних зонах зсувів, селевих потоків і снігових лавин, у зонах можливого затоплення, у сейсмічних районах тощо.

Для охорони навколишнього середовища міських і сільських поселень у межах приміських зон на землях лісового фонду формуються «зелені зони» у складі лісопаркової та лісогосподарської частин, місць відпочинку, заповідних об'єктів.

Навколо міських і сільських поселень, які розташовані у безлісих районах, створюються вітрозахисні і берегоукріплювальні лісові смуги завширшки 500 м (для найзначніших і значних міст], 100 м (для великих і середніх міст] і 50 м (для малих міст і сільських поселень].

Історичне середовище з пам'ятками історії та культури зберігається і захищається на засадах створення спеціальних зон, які охоплюють місця концентрації пам'яток, зони регулювання забудови, які прилягають до охоронних зон, зони ландшафту, що охороняється, заповідні зони.

Конкретні заходи щодо захисту навколишнього середовища вживаються відповідно до специфіки окремих джерел забруднення.

8.2 Забруднення навколишнього середовища офісом комп'ютерної фірми

Ніхто з учених так і не придумав можливості сконструювати повністю безпечну для навколишнього середовища техніку. Стара техніка нерідко стає причиною забруднення навколишнього середовища. Відправлена на звалище вийшла з ладу електротехніка цілком здатна дійсно завдати нашій планеті великої шкоди. Електротехніка в наш час не купується на довгі роки, тому що на зміну техніці майже відразу приходять нові моделі техніки. Рідко буває, що обладнання в компаніях в наші дні взагалі ремонтується - зазвичай замість негайно купується нове обладнання. Не секрет, що якщо зламане оснащення міститься в неправильних умовах, то воно загрожує завдати шкоди чистоті довкілля. Без сумніву, правильне поводження з непотрібною технікою – це гарантія того, що екології електронне сміття не завдасть шкоди. Необхідно усвідомлювати, як захистити навколишнє середовище.

Всі, хто застосовують лампи денного світла для освітлення офісів, магазинів, кафе та ін. Приміщень, зазвичай не підозрюють про можливі проблеми, які виникнуть під час перевірок. Люмінесцентні лампи містять небезпечні для здоров'я людини хімічні речовини. У кожної люмінесцентної лампи знаходиться від 20 до 500 мг ртуті (в залежності від типу і технології), тому вони відносяться до першого, тобто найвищого класу небезпеки. Утилізація люмінесцентних ламп, їх зберігання, повинні проводитися відповідно до вимог законодавства, розглянутими нижче.

8.3 Заходи для зменшення забруднення

За допомогою повністю безпечних способів зберегти природу від шкоди людям дає шанс процедура утилізації. Кілька етапів утилізації за правилами повинна пройти утилізованих техніка. Той факт, що необхідність кожного з етапів утилізації величезна, професіонали цієї сфери обов'язково аргументують. Доставка відправленого на утилізацію оснащення – це дуже складне завдання, яку доручають професіоналам. Наступний етап – розбирання привезеного оснащення настає відразу після доставки на підприємство. Зрозуміло, сортувати обладнання теж не просто, цією проблемою в свою чергу займаються професіонали. Ці етапи покликані за всіма правилами відокремити одні елементи техніки від інших. У той час, коли співробітники в процесі сортування знаходять електронний компонент, вони відправляють цей компонент на проведення такої процедури, як афінаж. Під час цієї процедури дорогоцінні метали відділяються від домішок. Треба, до того ж, пам'ятати, що закон передбачає обов'язкове проведення процесу афінажу. Фонд дорогоцінних металів і дорогоцінного каміння – ось куди відправляться оброблені дорогоцінні метали.

Способи утилізації ламп:

1) термічна демеркуризація

Процес по розбору лампи відбувається на спеціальній установці – демеркуризаторі. Лампи подаються по одній в установку, де після подрібнення з склометалевого брухту проходить сублімація парів ртуті. Далі під дією сорбенту пари ртуті уловлюються і осідають в конденсаторі.

2) термовакуумний метод

Відходи, у яких знаходиться ртуть, потрапляють у вакуумну пастку, відбувається конденсація небезпечного пара, після чого пари ртуті піддаються виморожуванню рідким азотом. Потім розморожена зібрана ртуть стікає в приймач.

3) реагентний метод

Спосіб оснований на обробці роздріблених виробів хімічними демеркуризаторами з метою переведення ртуті в важкорозчинні сполуки.

Всі ці процеси небезпечні, оскільки биті лампи – відкрите джерело отруйних парів ртуті. Відомо, що утилізація однієї лампи КЛЛ вагою до 150 грам в процесі переробки може дати до 50 грамів скла (відправляється на повторне використання для виробництва тих же ламп або абразивів) і близько 5 мг ртуті на виготовлення нових ламп, а також до 3 грам люмінофора, що підлягає

ВИСНОВКИ

Метою даної кваліфікаційної роботи є підвищення достовірності оцінювання розміру програмних застосунків на TypeScript. Мета роботи була повністю досягнута.

В кваліфікаційній роботі було удосконалено регресійну модель для оцінювання розміру програмних застосунків на TypeScript та розроблено програмне забезпечення для її реалізації.

Також в роботі розроблена документація:

– технічне завдання (додаток А);
– програма і методика випробувань програмного забезпечення (додаток Б);
– опис програми (додаток В);
– інструкція користувача (див. додаток Г).

Всі задачі, які ставилися в кваліфікаційній роботі, були виконані в повному обсязі.

Виконано тестування та випробування розробленого програмного забезпечення, які показали, що програмне забезпечення повністю відповідає задачам дослідження та вимогам, які висувалися в технічному завданні.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Титов, А.И. Выбор метрики размера проекта в модели оценки трудоемкости разработки программ [Текст] / А.И.Титов // Интеллектуальные технологии на транспорте. –2016. – Вып. 5. – С. 31-38.

2. Malathi, S. et al. Analysis of size metrics and effort performance criterion in software cost estimation [Текст] / S. Malathi et al. // Indian Journal of Computer Science and Engineering. –2012. – Volume 2, Issue 1. – P. 24-31.

3. Приходько, Н. В. Нелінійна регресійна модель для оцінювання розміру програмного забезпечення інформаційних систем на базі VB [Текст] / Н. В. Приходько, С. Б. Приходько // Інформаційні технології та компютерна інженерія. –2018. – № 3. – С. 37-42.

4. Макарова, Л. М. Побудова нелінійної регресійної моделі для оцінювання розміру веб-додатків, реалізованих мовою Java [Текст] / Л. М. Макарова, Н. В. Приходько, О. О. Кудін // Вестник Херсонского национального университета. –2019. – №2(69), ч.1. – С. 145-154.

5. Латанська, Л.О. Побудова довірчого інтервалу та інтервалу передбачення нелінійної регресійної моделі оцінювання розміру програмного забезпечення на TYPESCRIPT / Л.О. Латанська, Д.В. Чебаков // VI Міжнародна науково-технічна конференція «Комп'ютерне моделювання та оптимізація складних систем», Дніпро, ДВНЗ УДХТУ, 4-6 листопада 2020, с.45-46.

6. Klint Finley. Microsoft Previews New JavaScript-Like Programming Language TypeScript TechCrunch (1 октября 2012) [Електронний ресурс]. – Режим доступу: https://techcrunch.com/2012/10/01/microsoft-previews-new-javascript-like-programming-language-typescript/ – Назва з екрану

7. Новичков, А. Метрики кода и их практическая реализация в Subversion и ClearCase. Часть 1 – метрики / А. Новичков, А. Шамрай, А. Черников [Электронный ресурс]. – Режим доступу: http://cmcons.com/articles/ CC_CQ /dev_metrics/mertics_part_1/ – Назва з екрану.

8. Ледовских, И. Метрики сложности кода [Электронный ресурс]. – Режим доступа: http://www.ispras.ru/ru/preprints/docs/prep_25_2013.pdf – Назва з екрану.

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

10. Boehm, B.W. The COCOMO 2.0 Software Cost Estimation Model [Текст] / B.W. Boehm. – American Programmer, 2000. – 586 p.

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

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

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

14. Кожевніков, І.Г. Емпіричні моделі оцінки вартості розробки програмного забезпечення [Текст] / І.Г. Кожевніков // Економічний простір. – 2014. – №83. – С.195-208.

15. Нелінійна регресія. вибір і порівняння нелінійних регресійних моделей [Електронний ресурс]. – Режим доступу: http://dn.khnu.km.ua/dn/k_default.aspx?M=k1314&T=02&lng=1&st=0 – Назва з екрану

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

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

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

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

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

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

22. What accuracy statistics really measure [Електронний ресурс]. – Режим доступу: https://core.ac.uk/download/pdf/334518.pdf – Назва з екрану

23. Вендров А.М., Проектирование программного обеспечения экономических информационных систем. – М.: Финансы и статистика, 2006. – 544 с.

ДОДАТОК А

Технічне завдання на розробку програмного забезпечення «Удосконалення регресійної моделі для оцінювання розміру програмних застосунків на TypeScript»

Вступ

Назва програми – ПЗ оцінювання розміру програмних застосунків на TypeScriptT. Програмне забезпечення являється програмою, яка буде надавати змогу користувачу провести оцінку розміру.

1. Підстави для розробки

Документом, на підставі якого ведеться розробка програмного продукту, є завдання на кваліфікаційну роботу «Удосконалення регресійної моделі для оцінювання розміру програмних застосунків на TypeScript та розробка програми для її реалізації».

2. Призначення розробки

2.1. Функціональне призначення

Даний програмний продукт дозволить автоматизувати та скоротити час відповідних розрахунків.

2.2. Експлуатаційне призначення

Програмний продукт буде використовуватись для оцінювання розміру програмних застосунків на TypeScript та оптимізації коду і побудови інтервала прогнозування та довірчого інтервала на основі десяткового логарифму.

3. Вимоги до програми

3.1. Вимоги до функціональних характеристик

3.1.1. Вимоги до складу функцій, що виконуються

Програма повинна забезпечувати можливість виконання наведених нижче функцій:

1) внесення даних про проекти (кількість класів);

2) виконання нормалізації внесених даних за допомогою перетворення на основі десяткового логарифму;

3) оцінювання розміру коду;

4) побудова довірчого інтервалу;

5) побудова інтервалу передбачення;

6) редагування даних.

3.1.2 Вимоги до організації вхідних та вихідних даних

Введення вхідних даних здійснюється користувачем у відповідні поля діалогового вікна програмного забезпечення.

Вихідні дані (результат оцінювання розміру Java-застосунку, довірчий інтервал, інтервал передбачення) виводиться у діалоговому вікні програмного забезпечення.

3.2 Вимоги до надійності

3.2.1 Вимоги до забезпечення надійності функціонування програми

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

3.2.2 Відмова виконання програмного продукту через некоректні дії користувача

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

3.4 Умови експлуатації

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

3.5 Вимоги до інформаційної та програмної сумісності

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

3.6 Вимоги до захисту інформації та програм

Вимоги до захисту інформації та програм відсутні.

3.7 Вимоги до маркування та упаковки

3.7.1 Вимоги до маркування

Вимоги до маркування відсутні.

3.7.2 Вимоги до упаковки

Вимоги до упаковки відсутні.

3.8 Вимоги до транспортування та зберігання

Вимоги до транспортування програми відсутні.

4 Вимоги до програмної документації

Склад програмної документації:

– технічне завдання;

– текст програми;

– опис програми;

– програма і методика випробувань;

– інструкція користувача.

5 Вимоги до складу та параметрів технічних засобів

– ПК з процесором не нижче : Celeron N2840;

– Обсяг оперативної пам‘яті - не менше 2 Гб;

– Необхідний обсяг на жорсткому диску - не менше 300 Мб.

5 Техніко-економічні показники

Економічна ефективність впровадження програмного забезпечення розрахована у розділі 6. Згідно з отриманими результатами впровадження даного ПЗ є доцільним, а термін його окупності становить приблизно 4 місяці.

6 Стадії та етапи розробки програмного забезпечення

Стадії та етапи розробки програми оцінювання розміру програмних застосунків на TypeScript, зазначені в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки програмного забезпечення

Стадії розробки	Етапи робіт	Вміст робіт	Контрольні точки
Технічне завдання	Обумовлення необхідності розробки	Постановка задачі. Збір початкових матеріалів.	З 08.09.2020 до 12.09.2020
	Розробка та затвердження технічного завдання	Визначення вимог до ПЗ. ВОбговорення та затвердження технічного завдання	З 13.09.2020 до 20.09.2020
Ескізний проект	Розробка ескізного проекту	Попередня розробка структури вхідних та вихідних даних, уточнення методів рішення завдань, загальний опис алгоритму	З 21.09.2020 до 21.10.2020
	Затвердження ескізного проекту	Розробка пояснювальної записки. Узгодження і затвердження ескізного проекту	З 22.10.2020 до 24.10.2020
Технічний проект	Розробка технічного проекту	Уточнення структури вхідних та вихідних даних, розробка алгоритму рішення задачі, розробка структури ПЗ	З 25.10.2020 до 15.11.2020
	Затвердження технічного проекту	Розробка пояснювальної записки. Узгодження і затвердження технічного проекту	З 16.11.2020 до 18.11.2020
Робочий проект	Розробка ПЗ	Програмування та налагодження програми	З 19.11.2020 до 29.11.2020
	Розробка програмної документації	Розробка програмної документації відповідно до вимог ГОСТ 19.101-77	З 30.11.2020 до 10.12.2020
	Випробування програми	Розробка, узгодження та затвердження програми і методики випробувань, коригування програми і програмної документації за результатами випробувань	З 11.12.2020 до 14.12.2020

7 Порядок контролю та приймання

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

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

ДОДАТОК Б

Програма і методика випробувань пз оцінювання розміру програмних застосунків на TypeScript

1 Об’єкт випробувань

Об’єктом випробувань є ПЗ для оцінювання розміру програмних застосунків на TypeScript.

2 Мета випробувань

Перевірка функціональних можливостей програмного забезпечення на відповідність вимогам технічного завдання.

3 Вимоги до програми

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

1) внесення даних про проекти (кількість класів);

2) виконання нормалізації внесених даних за допомогою перетворення на основі десяткового логарифму;

3) оцінювання розміру коду;

4) побудова довірчого інтервалу;

5) побудова інтервалу передбачення;

6) редагування даних.

4 Вимоги до програмної документації

Склад програмної документації, що надається на випробування:

- технічне завдання;
- текст програми;
- опис програми;
- програма і методика випробувань;

– інструкція користувача.

5 Вимоги до складу та параметрів технічних засобів

– ПК з процесором не нижче : Celeron N2840;

– Обсяг оперативної пам‘яті - не менше 2 Гб;

– Необхідний обсяг на жорсткому диску - не менше 300 Мб.

6 Засоби і порядок випробувань

Випробування проводилось на платформі Windows 7 32bit.

Порядок випробувань:

– встановити і запустити програму;

– перевірити можливості додавання даних до програми;

– перевірити можливість редагування введених даних;

– перевірити можливість виконання нормалізації;

– перевірити можливість знаходження прогнозного значення;

– перевірити можливість знаходження інтервалу прогнозування;

– перевірити можливість знаходження довірчого інтервалу.

7 Методи випробувань

Випробування будуть проводитися у наступному порядку:

1) Перевірити запуск програмного забезпечення;

2) Перевірити функцію введення даних;

3) Перевірити функцію корегування.

Перевірка запуску програмного забезпечення

При перевірки запуску програмного забезпечення необхідно здійснити наступні дії:

– запустити програмне забезпечення командою: java –jar ./jcodesize.jar.

В наслідок вказаних дій повинно відкритися головне вікно програми.

Перевірка функції введення даних

При перевірки функції введення даних необхідно:

– у головному меню програми вибрати пункт «Файл-Новий»,

В наслідок вказаних дій повинно відкритися вікно першого кроку формування початкових даних.

– ввести найменування проекту;

– натиснути кнопку «Далі».

В наслідок вказаних дій повинно відкритися вікно другого кроку формування початкових даних.

– ввести початкові дані;

– натиснути кнопку «Далі».

В наслідок вказаних дій повинно відкритися вікно третього кроку формування початкових даних.

– перевірити наведені у вікні початкові дані;

– натиснути кнопку «Так».

В наслідок вказаних дій запис з інформацією про даний проект буде внесена до бази даних і відбувається перехід до функції розрахувати.

ДОДАТОК В

Опис програмного забезпечення для оцінювання розміру програмних застосунків на TypeScript

1 Загальні відомості

Програмне забезпечення для оцінювання розміру програмних застосунків на TypeScript, яке дозволяє автоматизувати та знизити трудомісткість розрахункових робіт.

Програмне забезпечення реалізовано з використанням мови програмування Java на базі jre 1.8. Для функціонування програмного забезпечення необхідний встановлений і налаштований SQL сервер баз даних.

2 Функціональне призначення

Програмне забезпечення призначене для оцінювання розміру програмних застосунків на TypeScript.

3 Опис логічної структури

За своєю логічною структурою програмне забезпечення складається з наступних модулів.

Модуль аналітика – забезпечує користувача такими функціями, як введення початкових даних, редагування даних, а також дозволяє переглянути результати оцінювання розміру програмних застосунків на TypeScript.

Модуль розрахунку – дозволяє виконати функції нормалізації початкових даних, оцінювання розміру програмних застосунків на TypeScript, розрахунку довірчого інтервалу та інтервалу прогнозування.

4 Вимоги до складу та параметрів технічних засобів

– ПК з процесором не нижче : Celeron N2840;

– Обсяг оперативної пам‘яті - не менше 2 Гб;

– Необхідний обсяг на жорсткому диску - не менше 300 Мб.

5 Виклик та завантаження

Для запуска програмного забезпечення необхідна операційна система Windows X/8.1, Linux, jre 1.8, MySQL версії 8.0.17. Завантаження здійснюється командою: java –jar ./jcodesize.jar.

6 Вхідні дані

Вхідними даними є найменування проекту, кількість класів.

7 Вихідні дані

Вихідні дані (результат оцінювання розміру, довірчий інтервал, інтервал передбачення).

ДОДАТОК Г

Керівництво користувача програмного забезпечення для оцінювання розміру програмних застосунків на TypeScript

1 Вступ

Це керівництво призначене для ознайомлення користувача із технічними і функціональними можливостями програмного забезпечення для оцінювання розміру програмних застосунків на TypeScript.

.

Програмне забезпечення призначене для оцінювання розміру програмних застосунків на TypeScript.

Програмне забезпечення має графічний інтерфейс, який дозволяє введення даних, переглянути результати оцінювання розміру програмних застосунків на TypeScript.

2 Загальні відомості про програму

2.1 Позначення і найменування програми

Найменування програми – «Оцінювання розміру програмних застосунків на TypeScript».

2.2 Мови програмування, на яких написана програма

Програма написана на мові програмування Java.

2.3 Призначення програми

Програмне забезпечення призначене для оцінювання розміру програмних застосунків на TypeScript.

2.4 Можливості програми

Програмне забезпечення виконує наступні функції:

- введення початкових даних;
- редагування даних;
- нормалізація даних;
- оцінювання розміру коду;
- розрахунок довірчого інтервалу;
- розрахунок інтервалу прогнозування.

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:

- ПК з процесором не нижче : Celeron N2840;
- Обсяг оперативної пам‘яті - не менше 2 Гб;
- Необхідний обсяг на жорсткому диску - не менше 300 Мб.

3.3 Програмне забезпечення, необхідне для функціонування програми

Операційна система, в якій встановлена версія jre 1.8.

4 Загальний опис роботи програми

Робота користувача починається з вибору відповідного пункту меню.

В меню Файл користувач може виконати наступні дії:

- відкрити новий проект;
- відкрити раніше збережений проект;
- закінчити роботу із програмою.

ДОДАТОК Д

Текст програмного забезпечення

Interval.java

```
package com.jcs;

import com.jcs.exceptions.*;

/**
 *Базовый класс для хранения доверительного интервала
 * и интервала прогнозирования.
 * @author User
 */
public final class Interval {
    /**
     * Хранит значение нижнюю границу интервала.
     */
    private double low;

    /**
     * Хранит значение верхней границы интервала.
     */
    private double up;

    /**
     * Возвращает нижнюю границу интервала.
     * @return нижnz границf интервала.
     */
    public double getLow() {
        return low;
    }


    /**
     * Возвращает значение верхней границы интервала.
     * @return верхняя граница интервала.
     */
    public double getUp() {
        return up;
    }


    /**
     * Конструктор принимающий границы интервала.
     * @param low   нижняя граница.
```

```
     * @param up    верхняя граница.
     * @throws LessThanZeroException    если параметр меньше нуля.
     * @throws EqualsZeroException      если параметр равен нулю.
     * @throws IllegalArgumentException если нижняя граница больше
     * или равна правой.
     */
    public Interval(double low, double up) throws LessThanZeroException,
EqualsZeroException, IllegalArgumentException{
        if(low<0 || up<0)
            throw new LessThanZeroException();
        if(low==0 || up==0)
            throw new EqualsZeroException();
        if(low >= up)
            throw new IllegalArgumentException();

        this.low = low;
        this.up = up;
    }


}
```

XData.java

```
package com.jcs;

import com.jcs.exceptions.*;

/**
 * Класс для инкапсуляции начальных данных для оценки размера кода.
 * @author User
 */
public final class XData {

    /**
     * Хранит количество классов.
     */
    private double x;

    /**
     * Возвращает значение количество классов в проекте.
     * @return количество классов в проекте.
     */
    public double getX() {
        return x;
    }

    /**
     * Конструктор принимает значения факторов уровнения.
```

```
     * @param x    значение количество классов в проекте.
     * @throws LessThanZeroException    если параметр меньше нуля.
     * @throws EqualsZeroException      если параметр равен нулю.
     */
    public XData(double x) throws LessThanZeroException, EqualsZeroException{
        if(x<0.0)
            throw new LessThanZeroException();
        if(x==0.0)
            throw new EqualsZeroException();

        this.x = x;
    }

    /**
     * Принимает значения факторов уровнения и возвращает объект XData,
     *  содержащий исходные данные.
     * @param x    значение кол-ва классов.
     * @return      объект XData, содержащий исходные данные.
     * @throws LessThanZeroException    если параметр меньше нуля.
     * @throws EqualsZeroException      если параметр равен нулю.
     */
    public static XData toXData(double x) throws LessThanZeroException,
EqualsZeroException {

        return new XData(x);
    }

    /**
     * Возвращает массив нормализированых значений исходных
     * данных для уровнения регрессии.
     * @return массив нормализированных значений сходных данных.
     */
    public double[]     getNormal(){
        double m[] = new double[1];

        m[0]    =   Math.log(x);

        return m;
    }

}
```

DljaRaschotov.java

```
package com.jcs.mat;

import com.jcs.exceptions.*;
import com.jcs.XData;
```

```
public interface DljaRaschotov {

   default double getTStjudent(){
      return 2.3534;
   }

   default double getSS(){
      return 0.2226;
   }

   default int     getN(){
      return  30;
   }
```

JCSException.java

```
package com.jcs.exceptions;

/**
 * Базовый класс для исключений программы
 * @author User
 */
public class JCSException extends Exception{

   public JCSException() {
   }

   public JCSException(String arg0) {
      super(arg0);
   }

}
```

EmptyLineException.java

```
package com.jcs.exceptions;

public class EmptyLineException extends JCSException{


}
```

EqualsZeroException.java

```
package com.jcs.exceptions;

public class EqualsZeroException extends JCSException{


}
```

LessThanZeroException.java

```
package com.jcs.exceptions;

public class LessThanZeroException extends JCSException{


}
```

NoIntervalException.java

```
package com.jcs.exceptions;

public class NoIntervalException extends JCSException{


}
```

NoSourceDataException.java

```
package com.jcs.exceptions;

public class NoSourceDataException extends JCSException{


}
```

About.java

```
package com.jcs.gui;

public class About extends javax.swing.JPanel {

    public About() {
        initComponents();
    }


    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JSeparator jSeparator1;
    // End of variables declaration//GEN-END:variables
}
```

InternalRezultat.java

```
package com.jcs.gui;
```

```
public class InternalRezultat extends javax.swing.JInternalFrame {

    /**
     * Creates new form InternalRezultat
     */
    public InternalRezultat() {
        initComponents();
    }

    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jLabel7 = new javax.swing.JLabel();
        jSeparator1 = new javax.swing.JSeparator();
        jPanel2 = new javax.swing.JPanel();
        jLabel8 = new javax.swing.JLabel();
        jLabel9 = new javax.swing.JLabel();
        jSeparator2 = new javax.swing.JSeparator();
        jLabel10 = new javax.swing.JLabel();
        jLabel11 = new javax.swing.JLabel();
        jLabel12 = new javax.swing.JLabel();
        jLabel13 = new javax.swing.JLabel();
        jLabel14 = new javax.swing.JLabel();
        jLabel15 = new javax.swing.JLabel();
        jLabel16 = new javax.swing.JLabel();
        jLabel17 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();

        setTitle("Результат оцінки");

        jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder("Проект"));

        jLabel1.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
        jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

        jLabel2.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
        jLabel2.setText("X");

        jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

        jLabel4.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
        jLabel7.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
```

```
        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(114)
                .addComponent(jLabel1,  javax.swing.GroupLayout.PREFERRED_SIZE,  121,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TR
AILING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(90)
                        .addComponent(jLabel3)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(jLabel6))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addComponent(jLabel2)
                        .addGap(46)
                        .addComponent(jLabel4)))
                .addGap(45)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                    .addComponent(jLabel5, javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jLabel7, javax.swing.GroupLayout.Alignment.TRAILING))
                .addGap(114))
        );
        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel1)
                .addGap(18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BA
SELINE)
```

```
                .addComponent(jLabel2)
                .addComponent(jLabel4)
                .addComponent(jLabel5))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BA
SELINE)
                .addComponent(jLabel3)
                .addComponent(jLabel6)
                .addComponent(jLabel7))
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );

        jPanel2.setBorder(javax.swing.BorderFactory.createTitledBorder("Оцінка розміру
коду"));

        jLabel8.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
        jLabel8.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
        jLabel8.setText("Розмір:");

        jLabel9.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N

        jLabel10.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N
        jLabel10.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel10.setText("Ніжня межа");

        jLabel11.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N
        jLabel11.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel11.setText("Верхня межа");

        jLabel12.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
        jLabel12.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
        jLabel12.setText("Довірчий інтервал:");

        jLabel13.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
        jLabel13.setText("Інтервал прог-ня:");

        jLabel14.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N

        jLabel15.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N

        jLabel16.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N

        jLabel17.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N

        javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
```

```
        jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel2Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jSeparator2)
                .addContainerGap())
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel2Layout.createSequentialGroup()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                    .addGroup(jPanel2Layout.createSequentialGroup()
                        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addComponent(jLabel10)
                        .addGap(36, 36, 36))
                    .addGroup(jPanel2Layout.createSequentialGroup()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                            .addGroup(jPanel2Layout.createSequentialGroup()
                                .addGap(121, 121, 121)
                                .addComponent(jLabel8)
                                .addGap(18, 18, 18)
                                .addComponent(jLabel9))
                            .addGroup(jPanel2Layout.createSequentialGroup()
                                .addContainerGap()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                                    .addComponent(jLabel12)
                                    .addComponent(jLabel13))
                                .addGap(32, 32, 32)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                                    .addComponent(jLabel16)
                                    .addComponent(jLabel14))))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                    .addComponent(jLabel15)
                    .addComponent(jLabel11)
```

```
                .addComponent(jLabel17))
            .addGap(38, 38, 38))
        );
        jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BA
SELINE)
                    .addComponent(jLabel8)
                    .addComponent(jLabel9))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jSeparator2,     javax.swing.GroupLayout.PREFERRED_SIZE,
10, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BA
SELINE)
                    .addComponent(jLabel10)
                    .addComponent(jLabel11))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BA
SELINE)
                    .addComponent(jLabel12)
                    .addComponent(jLabel14)
                    .addComponent(jLabel15))
                .addGap(18, 18, 18)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BA
SELINE)
                    .addComponent(jLabel13)
                    .addComponent(jLabel16)
                    .addComponent(jLabel17))
                .addContainerGap(19, Short.MAX_VALUE))
        );

        jButton1.setText("Так");

        javax.swing.GroupLayout                layout                =                new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
```

```
            .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                .addComponent(jPanel2,          javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jSeparator1)
                .addComponent(jPanel1,          javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addContainerGap())
          .addGroup(layout.createSequentialGroup()
            .addGap(169, 169, 169)
            .addComponent(jButton1)
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
      );
      layout.setVerticalGroup(
         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
         .addGroup(layout.createSequentialGroup()
            .addComponent(jPanel1,          javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jPanel2,          javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jSeparator1,    javax.swing.GroupLayout.PREFERRED_SIZE,
12, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jButton1)
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
      );

      pack();
   }
```