

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

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

«Допущений до захисту»
Завідувач кафедри

«__» _____ 2022 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «магістр»

на тему: Нелінійна регресійна модель для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій та розробка програми для її реалізації

Виконала: студентка 6151м групи

Павлюк К.А.

(підпис, ПІБ)

Керівник роботи:

доцент кафедри ПЗАС, к.ф.-м.н., доцент

(посада, науковий ступень вчене звання)

Латанська Л.О.

(підпис, ПІБ)

Миколаїв – 2022 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

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

«ЗАТВЕРДЖУЮ»

Гарант освітньої програми

_____ проф. С.Б.Приходько
 (підпис)

«10» жовтня 2022 р.

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

Студенту Павлюк Ксенії Андріївні _____

(Прізвище, ім'я, по батькові)

1. Тема роботи: «Нелінійна регресійна модель для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій та розробка програми для її реалізації»

Керівник роботи Латанська Л. О.

Затверджені наказом ректора № 798 уч від «28» _____ 09 _____ 2022 р.

2. Термін подання роботи: 14.11.2022 р. _____

3. Вихідні дані по роботі: _____

4. Перелік питань, що належать до розробки (найменування розділів) _____

Вступ (Актуальність теми. Зв'язок роботи з науковими програмами, планами, темами. Мета і завдання дослідження. Об'єкт дослідження. Предмет дослідження. Методи дослідження. Наукова новизна одержаних результатів. Практичне значення одержаних результатів. Особистий внесок здобувача. Апробація результатів досліджень. Публікації.); Огляд літератури та обґрунтування необхідності проведення досліджень за обраною темою; Викладення результатів власних досліджень з висвітленням того нового, що пропонується; Проект програмного забезпечення; Організаційно-економічний розділ; Розділи з охорони праці та охорони навколишнього середовища; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма і методика випробувань програмного забезпечення)

5. Перелік презентаційних матеріалів: 1 Актуальність теми, 2 Мета, об'єкт і предмет дослідження, 3 Методи дослідження, 4 Наукова новизна одержаних результатів, 5 Практичне значення одержаних результатів, 6 Апробація результатів досліджень та публікації, 7 Емпіричні дані, 8 Нелінійна регресійна модель для оцінювання розміру програмних проектів для криптовалютних операцій, написаних мовою TypeScript, 9 Лінійна регресійна модель для оцінювання розміру програмних проектів для криптовалютних операцій, написаних мовою TypeScript, 10 Метрики якості, 11 Контекстна діаграма ПЗ для оцінювання розміру програмних проектів, 12 Концептуальна модель даних ПЗ для оцінювання розміру програмних проектів, 13 Діаграма переходів станів ПЗ для оцінювання розміру програмних проектів, 14 Декомпозиція контекстного процесу ПЗ для оцінювання розміру програмних проектів, 15 Логічна модель даних ПЗ, 16 Інтерфейс, 17 Висновки _____

6. Консультанти розділів роботи

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

7. Дата видачі завдання 10.10.2022 р.

КАЛЕНДАРНИЙ ПЛАН

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу Вступ	11.10.2022	НС*
2.	Підготовка розділу з огляду літератури та обґрунтування необхідності проведення досліджень за обраною темою	12.10.2022	НС*
3.	Підготовка розділу (ів) з результатів власних досліджень	13.10.2022	НС*
4.	Підготовка розділу з проекту програмного забезпечення	03.11.2022	
5.	Підготовка організаційно-економічного розділу	06.11.2022	
6.	Підготовка розділу з охорони праці	09.11.2022	
7.	Підготовка розділу з охорони навколишнього середовища	11.11.2022	
8.	Підготовка розділу Висновки	12.11.2022	
9.	Оформлення списку використаних джерел та додатків	13.11.2022	
10.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версії роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	14.11.2022	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку)
11.	Підготовка презентації та доповіді	19.11.2022	
12.	Попередній захист роботи на засіданні кафедри ПЗАС	20.11.2022	
13.	Подання на кафедру ПЗАС електронних версії наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді	28.11.2022	

* - за результатами наукового стажування (НС), яке було з 01.09.2022 по 09.10.2022 р.

Студент

(підпис)

Павлюк К.А.

(ПІБ)

Керівник роботи

(підпис)

Латанська Л. О.

(ПІБ)

РЕФЕРАТ

Павлюк Ксенія Андріївна

Нелінійна регресійна модель для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій 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, 2022 р.

Обсяг роботи: 110 стор., 15 табл., 9 рис., 21 використане джерело, 5 додатків.

Актуальність теми. Ефективність оцінювання розміру програм є основою для інших оцінювань та не існує ефективних універсальних інструментів прогнозування розміру майбутніх продуктів, які є оптимальними для всіх типів проектів, мов програмування, фреймворків, технологій і середовищ розробки. Тому удосконалення двохфакторної регресійної моделі для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій з використанням нормалізуючого перетворення на основі десяткового логарифму є актуальною задачею інженерії програмного забезпечення.

Мета дослідження. Підвищення достовірності оцінювання розміру програмних проектів для криптовалютних операцій, написаних мовою TypeScript, за допомогою нелінійної регресійної моделі являється метою дослідження.

Об'єкт дослідження. В якості об'єкту дослідження обрано процес оцінювання розміру програмних проектів для криптовалютних операцій, що створюються мовою TypeScript.

Предмет дослідження. В якості предмету дослідження розглядаються нелінійні регресійні моделі для оцінювання розміру програмних проектів для криптовалютних операцій, написаних мовою TypeScript.

Методи дослідження. При дослідженні використовувалися методи математичної статистики, теорії ймовірностей, лінійного та нелінійного регресійного аналізу.

Наукова новизна одержаних результатів. Удосконалено регресійну модель для оцінювання розміру програмних проектів для криптовалютних операцій, написаних мовою TypeScript, за рахунок використання нормалізуючого перетворення на основі десяткового логарифму і багатовимірних викидів. Отримана двохфакторна нелінійна регресійна модель в порівнянні з двохфакторною лінійною регресійною моделлю має кращі показники якості R^2 , $MMRE$ та $PRED(0,25)$, що підвищує достовірність даного оцінювання в порівнянні з лінійною двохфакторною регресійною моделлю.

Практичне значення отриманих результатів. В рамках магістерської роботи розроблене ПЗ для оцінювання розміру програмних проектів для криптовалютних операцій, написаних мовою TypeScript, що дозволило

автоматизувати, скоротити час та підвищити достовірність оцінювання на ранніх етапах розробки.

Апробація результатів досліджень. Результати досліджень, викладених у роботі, були оприлюднені на V Всеукраїнській науково-практичній інтернет-конференції студентів, аспірантів та молодих вчених за тематикою: «Сучасні комп'ютерні системи та мережі в управлінні» (м. Хмельницький, 30.11.2022) .

Публікації. Результати досліджень викладені в 1 роботі.

Ключові слова: програмні застосунки для криптовалютних операцій, TYPEScript, оцінювання розміру, нелінійна регресійна модель, нормалізуюче перетворення.

ABSTRACT

Pavliuk Kseniia Andriivna

A nonlinear regression model for estimating the size of TypeScript software applications for cryptocurrency transactions, and the development of a program for its implementation

Qualification work for obtaining a master's degree in Specialty 121 – Software Engineering. Admiral Makarov National University of Shipbuilding. Mykolaiv, 2022.

The work contains: 110 pages, 15 tables, 9 figures, 21 references, 5 appendices.

Relevance of the theme. The performance of program size estimation is the basis for other estimates and there are no effective universal tools for predicting the size of future products that are optimal for all types of projects, programming languages, frameworks, technologies and development environments. Therefore, the improvement of a two-factor regression model for estimating the size of TypeScript software applications for cryptocurrency operations using a normalizing transformation based on a decimal logarithm is an urgent task of software engineering.

Purpose of the research. Increasing the reliability of estimating the size of software projects for cryptocurrency operations written in the TypeScript language using a nonlinear regression model is the goal of the study.

The object of the study. The process of evaluating the size of software projects for cryptocurrency operations created in the TypeScript language was chosen as the object of the research.

The subject of the study. Non-linear regression models for estimating the size of software projects for cryptocurrency transactions written in TypeScript language are considered as the subject of research.

Research methods. The research used methods of mathematical statistics, probability theory, linear and non-linear regression analysis.

Scientific novelty of the obtained results. Improved the regression model for estimating the size of software projects for cryptocurrency transactions written in the TypeScript language by using a normalizing transformation based on the decimal logarithm and multivariate outliers. The obtained two-factor nonlinear regression model compared to the two-factor linear regression model has better quality indicators R^2 , MMRE and PRED(0.25), which increases the reliability of this assessment compared to the linear two-factor regression model.

The practical significance of the obtained results. As part of the master's work, software was developed for estimating the size of software projects for cryptocurrency operations written in the TypeScript language, which made it possible to automate, reduce time, and increase the reliability of the assessment at the early stages of development.

Approbation of research results. The results of the research presented in the paper were made public at the 5th All-Ukrainian Scientific and Practical Internet Conference of

students, postgraduates and young scientists on the topic: "Modern computer systems and networks in management" (Khmelnyskyi, November 30, 2022).

Publications. The research results are presented in 1 paper.

Keywords: software applications for cryptocurrency operations, TypeScript, size estimation, nonlinear regression model, normalizing transformation.

ЗМІСТ

ВСТУП.....	10
1 АНАЛІЗ МЕТОДІВ ТА МОДЕЛЕЙ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНИХ ЗАСТОСУНКІВ.....	14
1.1 Основні метрики для оцінювання розміру програмних застосунків.....	14
1.2 Основні методи та моделі для оцінювання розміру програмних застосунків.....	20
1.3 Обґрунтування необхідності проведення досліджень	24
2 НЕЛІНІЙНА РЕГРЕСІЙНА МОДЕЛЬ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНИХ ЗАСТОСУНКІВ НА TYPESCRIPT ДЛЯ КРИПТОВАЛЮТНИХ ОПЕРАЦІЙ.....	26
2.1 Виявлення мультиколінеарності та викидів в емпіричних даних	26
2.2 Розробка двохфакторної нелінійної регресійної моделі оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій.....	30
3 ПРОЕКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНИХ ЗАСТОСУНКІВ НА TYPESCRIPT ДЛЯ КРИПТОВАЛЮТНИХ ОПЕРАЦІЙ.....	38
3.1 Постановка задачі на розробку ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій	38
3.2 Розробка ескізного проекту ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій.....	39
3.2.1 Розробка контекстної діаграми ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій.....	39
3.2.2 Розробка концептуальної моделі даних ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій.....	41
3.2.3 Розробка моделі переходів станів ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій.....	43
3.2.4 Розробка інтерфейсу користувача ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій.....	45
3.3 Розробка технічного проекту програмного забезпечення для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій.....	47

3.3.1 Розробка ієрархії діаграм потоків даних ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій.....	48
3.3.2 Структура представлення даних ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій.....	53
3.4 Розробка робочого проекту ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій.....	54
3.4.1 Вибір мови програмування та засобів розробки ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій.....	54
3.4.2 Кодування та тестування ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій.....	56
3.4.3 Випробування ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій.....	58
4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВІД РОЗРОБКИ ТА ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНИХ ЗАСТОСУНКІВ НА TYPESCRIPT ДЛЯ КРИПТОВАЛЮТНИХ ОПЕРАЦІЙ	61
4.1 Вступ.....	61
4.2 Розрахунок витрат на створення й експлуатацію програми для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій....	61
4.3 Розрахунок економічної ефективності розробки і впровадження ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій.....	65
4.4 Висновки за розділом 4.....	67
5 ОХОРОНА ПРАЦІ	68
5.1 Загальні визначення та актуальність питання охорони праці.....	68
5.2 Аналіз небезпечних і шкідливих факторів при роботі з ПК.....	71
5.3 Розрахунок системи загального освітлення приміщення з ПК.....	76
5.4 Розробка заходів щодо зменшення впливу шкідливих та небезпечних факторів при роботі з персональним комп'ютером.....	78
6 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА	84
6.1 Вступ	84
6.2 Види забруднення навколишнього природного середовища.....	85

6.3 Природоохоронна діяльність підприємств.....	85
6.4 Розробка заходів щодо зменшення забруднення	86
ВИСНОВКИ	89
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	90
ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ПЗ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНИХ ЗАСТОСУНКІВ НА TYPESCRIPT ДЛЯ КРИПТОВАЛЮТНИХ ОПЕРАЦІЙ	93
ДОДАТОК Б – ОПИС ПЗ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНИХ ЗАСТОСУНКІВ НА TYPESCRIPT ДЛЯ КРИПТОВАЛЮТНИХ ОПЕРАЦІЙ	97
ДОДАТОК В – ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ ПЗ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНИХ ЗАСТОСУНКІВ НА TYPESCRIPT ДЛЯ КРИПТОВАЛЮТНИХ ОПЕРАЦІЙ	99
ДОДАТОК Г – ІНСТРУКЦІЯ КОРИСТУВАЧА ПЗ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНИХ ЗАСТОСУНКІВ НА TYPESCRIPT ДЛЯ КРИПТОВАЛЮТНИХ ОПЕРАЦІЙ.....	102
ДОДАТОК Д – ТЕКСТ ПЗ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНИХ ЗАСТОСУНКІВ НА TYPESCRIPT ДЛЯ КРИПТОВАЛЮТНИХ ОПЕРАЦІЙ	105

ВСТУП

Актуальність теми. Розробка програмних систем, в тому числі і програмних застосунків на TypeScript для криптовалютних операцій, – процес трудомісткий, що вимагає комплексного вирішення проблем часу та бюджету систем, що розробляються. Найчастіше розробки завершуються не в строк, їхній бюджет перевищує наперед заданий і т. д. Головними причинами таких негативних результатів є неякісне оцінювання проекту; складові бюджету, які не оговорені на попередніх етапах планування проекту. Домогтися правильного оцінювання на ранніх стадіях розробки не просто. Неправильна оцінка, особливо на попередніх етапах, тягне за собою серйозні проблеми на наступних стадіях розробки [1].

Виділяють чотири основні етапи оцінювання програмних проектів:

- оцінювання розміру продукту розробки;
- оцінювання зусилля у людино-місяцях або людино-годинах;
- оцінювання графіку у календарних місяцях;
- оцінювання вартості проекту у погодженій валюті.

Отримання ефективної системи оцінювання розміру програмних проектів на ранньому етапі розробки є важливим завданням, що вимагає удосконалення існуючих методів. Адже саме ефективність оцінювання розміру програм є основою для інших оцінювань та може стати відправною точкою для успіху або невдачі проекту.

В роботі досліджується оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій в строках коду.

Серед моделей, які використовуються для оцінювання розміру програмного забезпечення в строках коду, можна виділити регресійні, які досить активно розробляються та вдосконалюються останнім часом. В залежності від виду функції, яка використовується для моделювання зв'язків

між залежною змінною та предиктором (предикторами) розрізняють лінійні [2] та нелінійні регресійні моделі [3-7].

Існує ряд припущень, при яких можливо використовувати лінійну регресійну теорію. Одним з них є нормальний розподіл залишків з нульовим середнім значенням. Це припущення виконується тільки в окремих випадках. Тому зазвичай будують нелінійні регресійні моделі, застосовуючи лінеарізуючі, нормалізуючі перетворення або метод простого перебору. На практиці досить вживаними є нормалізуючі перетворення, серед яких можна виділити десятковий логарифм.

За кількістю предикторів (факторів) регресійні моделі поділяються на однофакторні [3, 4] та багатфакторні [2, 5-7]. Багатфакторні моделі зазвичай мають кращу якість в порівнянні з однофакторними.

В даному дослідженні буде удосконалюватися двохфакторна регресійна модель для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій з використанням нормалізуючого перетворення на основі десятичного логарифму.

Незважаючи на постійні зусилля дослідників в напрямку оцінювання розміру програмних проектів, не існує ефективних універсальних інструментів прогнозування розміру майбутніх продуктів, які є оптимальними для всіх типів проектів, мов програмування, фреймворків і середовищ розробки. Тому удосконалення двохфакторної регресійної моделі для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій з використанням нормалізуючого перетворення на основі десятичного логарифму є актуальною задачею інженерії програмного забезпечення.

Мета і завдання дослідження. Підвищення достовірності оцінювання розміру програмних проектів для криптовалютних операцій, написаних мовою TypeScript, за допомогою нелінійної регресійної моделі являється метою дослідження.

Досягнення мети забезпечується вирішенням наступних завдань:

- аналізом методів та моделей оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій;
- обґрунтуванням необхідності проведення досліджень;
- побудовою нелінійної двохфакторної регресійної моделі оцінювання розміру програмних проектів для криптовалютних операцій, написаних мовою TypeScript;
- побудовою лінійної двохфакторної регресійної моделі оцінювання розміру програмних проектів для криптовалютних операцій, написаних мовою TypeScript у припущенні про нормальність емпіричних даних;
- побудовою для нелінійної та лінійної двохфакторних регресійних моделей довірчих інтервалів та інтервалів передбачення;
- порівнянням якості побудованих нелінійної та лінійної двохфакторних регресійних моделей;
- порівнянням довірчих інтервалів та інтервалів передбачення нелінійної та лінійної двохфакторних регресійних моделей;
- розробкою програми для реалізації побудованої нелінійної двохфакторної регресійної моделі оцінювання розміру програмних проектів для криптовалютних операцій, написаних мовою TypeScript.

Об’єкт дослідження. В якості об’єкту дослідження обрано процес оцінювання розміру програмних проектів для криптовалютних операцій, що створюються мовою TypeScript.

Предмет дослідження. В якості предмету дослідження розглядаються нелінійні регресійні моделі для оцінювання розміру програмних проектів для криптовалютних операцій, написаних мовою TypeScript.

Методи дослідження. При дослідженні використовувалися методи математичної статистики, теорії ймовірностей, лінійного та нелінійного регресійного аналізу.

Наукова новизна одержаних результатів. Удосконалено регресійну модель для оцінювання розміру програмних проектів для криптовалютних

операцій, написаних мовою TypeScript, за рахунок використання нормалізуючого перетворення на основі десяткового логарифму і багатовимірних викидів. Отримана двохфакторна нелінійна регресійна модель в порівнянні з двохфакторною лінійною регресійною моделлю має кращі показники якості R^2 , $MMRE$ та $PRED(0,25)$, що підвищує достовірність даного оцінювання в порівнянні з лінійною двохфакторною регресійною моделлю.

Практичне значення отриманих результатів. В рамках магістерської роботи розроблене ПЗ для оцінювання розміру програмних проектів для криптовалютних операцій, написаних мовою TypeScript, що дозволило автоматизувати, скоротити час та підвищити достовірність оцінювання на ранніх етапах розробки.

Особистий внесок здобувача. Основні положення кваліфікаційної роботи, що представлені до захисту, отримані здобувачем самостійно. У тезах конференції [8], які опубліковані у співавторстві, здобувачу належить розробка двохфакторної нелінійної регресійної моделі для оцінювання розміру програмних проектів для криптовалютних операцій, написаних мовою TypeScript.

Апробація результатів роботи. Результати досліджень, викладених у роботі, були оприлюднені на V Всеукраїнській науково-практичній інтернет-конференції студентів, аспірантів та молодих вчених за тематикою: «Сучасні комп'ютерні системи та мережі в управлінні» (м. Хмельницький, 30.11.2022) .

Публікації. Результати досліджень викладені в 1 роботі.

1 АНАЛІЗ МЕТОДІВ ТА МОДЕЛЕЙ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНИХ ЗАСТОСУНКІВ

1.1 Основні метрики для оцінювання розміру програмних застосунків

Для визначення реалістичності цілей проекту та для підвищення точності планування, виконується оцінювання трудомісткості розробки програмного забезпечення. Серед чинників, які впливають оцінку трудомісткості, найважливішим показником є розмір проекту. Хоча оцінки розміру недостатньо для розуміння загальної складності продукту, що розробляється, існує явна залежність між розміром проекту і його трудомісткістю [9].

Розглянемо основні групи метрик для оцінювання розміру програмного забезпечення (ПЗ) [10-11].

– Кількість рядків коду

Кількість рядків коду – це розмірно-орієнтована метрика ПЗ, в якій об'єм проекту розраховується виходячи з кількості рядків у тексті вихідного коду. Виникнення цієї методики оцінювання відноситься до 50-х років XX століття. Серед існуючих метрик обліку числа рядків можна виділити наступні:

- кількість фізичних рядків (LOC) – загальна кількість рядків вихідного коду, включаючи коментарі та порожні рядки;

- кількість логічних рядків коду (LLOC) – загальна кількість команд (залежить від мови програмування). Якщо мова підтримує розміщення декількох команд в одному рядку, то один фізичний рядок повинен бути врахований як кілька логічних, якщо він містить більше однієї команди мови.

Також існують похідні від основних метрик:

- кількість порожніх рядків;
- кількість рядків, що містять коментарі;

- відсоток коментарів (відношення рядків коду до рядків коментаря);
- середня кількість рядків для функцій (класів, файлів);
- середня кількість рядків, що містять вихідний код для функцій (класів, файлів);
- середня кількість рядків для модулів.

– Метрики Холстеда

Метрики, які базуються на аналізі числа рядків та синтаксичних елементів вихідного коду програми, були запропоновані М. Холстедом у 1977 році. Метрики Холстеда частково дозволяють врахувати можливість реалізації однієї й тієї ж функціональності різним числом рядків і операторів. Частіше всього ці метрики використовуються для оцінювання складності проміжних продуктів розробки, проте набір метрик також містить оцінки розміру. Метрики Холстеда базуються на чотирьох вимірюваних характеристиках програми:

- кількість унікальних операторів програми, включаючи символи-розділювачі, імена процедур та знаки операцій (словник операторів);
- кількість унікальних операндів програми (словник операндів);
- загальна кількість операторів у програмі;
- загальна кількість операндів у програмі.

На основі цих характеристик обчислюються різні метрики розміру, складності та якості програм, такі як:

- словник програми;
- довжина програми;
- обсяг програми;
- складність програми;
- зусилля на розробку програми.

Хоча метрики Холстеда дозволяють використовувати додаткові можливості з аналізу вихідного коду, які відсутні в метриці числа рядків коду, з точки зору оцінювання розміру проекту більша частина завдань залишається не вирішеною. Оцінювання загальної кількості операторів та їх операндів до

завершення етапу розробки програми може виявитися ще складнішим завданням, ніж оцінювання числа рядків коду.

– Функціональні точки

Аналіз функціональних точок – це метод вимірювання розміру ПЗ з погляду користувачів системи. Метод був розроблений Аланом Альбрехтом у середині 70-х. Даний метод призначений для оцінювання обсягу програмного продукту за функціональною моделлю, тобто оцінюється обсяг функцій системи, що розробляється. Основна мета цього методу в декомпозиції системи таким чином, щоб забезпечити прийнятну складність аналізу. Базовою одиницею виміру, на якій ґрунтується даний метод, є функціональна точка. Оцінювання розміру у функціональних точках базується на кількості та складності наступних елементів:

- зовнішніх вхідних елементів;
- зовнішніх вихідних елементів;
- зовнішніх запитів;
- внутрішніх логічних файлів;
- зовнішніх файлів інтерфейсу.

Вимірювання розміру проекту функціональними точками є актуальною метрикою. Основною перевагою цього підходу є те, що оцінка основана на вимогах до продукту, що дозволяє отримати оцінку трудомісткості на самих ранніх етапах роботи над проектом, відразу після виявлення необхідних вимог. На наступних етапах роботи оцінка може бути уточнена, тому використання цього методу можливе при використанні гнучких методологій розробки ПЗ. Проте, використання функціональних точок як одиниць виміру має ряд істотних недоліків. Для обчислення функціональних точок необхідно детально вивчити специфікацію вимог і підрахувати всі вихідні та результуючі елементи, файли, транзакції, що може бути дуже трудомістко. При цьому отримана оцінка може бути дуже суб'єктивна, оскільки деякі бізнес-процеси можуть мати високу алгоритмічну складність, але мати досить прості зовнішні введення і виведення.

– Точки властивостей

Метод точок властивостей є розширенням методу функціональних точок, яке враховує вимоги до системи і особливості її реалізації. Головна відмінність від методу функціональних точок полягає в тому, що одержувана оцінка коригується з урахуванням алгоритмічної складності реалізації. До вихідних типів елементів додається ще один – алгоритми. У методі точок властивостей алгоритми розглядаються як набір правил, що визначають будь-яку істотну обчислювальну задачу. Завдяки врахуванню алгоритмічної складності точки властивостей краще адаптовані до оцінювання систем з високою складністю алгоритмів, наприклад:

- системного ПЗ, операційних систем або компіляторів;
 - систем, що функціонують в реальному часі;
 - інженерних програм, таких як системи автоматизованого проектування (САПР) чи математичне ПЗ;
 - систем штучного інтелекту;
 - систем підтримки телекомунікацій.
- Об'єктні точки

У сучасних методологіях розробки, об'єктно-орієнтоване програмування (ООП) є найбільш поширеною методологією. Оскільки у початковому варіанті методу функціональних точок не було передбачено застосування об'єктно-орієнтованого підходу, був розроблений адаптований варіант, що оперує термінами ООП. Важливою його відмінністю від інших різновидностей методу функціональних точок є те, що він не розширяє стандартний набір типів елементів, а використовується інші:

- форми;
- звіти;
- модулі.

Фактично, в даній метриці кожному унікальному класу або об'єкту призначається одна об'єктна точка. Оцінювання проводиться приблизно за тими ж етапами, що і в функціональних точках:

- підрахунок кількості форм, звітів та компонентів;
- класифікація кожного екземпляра об'єкта за рівнем складності;
- визначення ваги для об'єктів;
- підсумовування зважених об'єктів;
- визначення відсотка повторного використання коду;
- визначення рівня продуктивності;
- підрахунок значення тривалості роботи у людино-місяцях.

Дана метрика зручна для оцінювання проектів, які розробляються за об'єктно-орієнтованою методологією та мають велику кількість компонентів з візуальною складовою.

– Метод UCP

Метод UCP (Use case points) є методом оцінюванням розміру проектів на основі діаграм UML та методології RUP. Як і багато інших сучасних методів оцінювання, UCP базується приблизно на тих же принципах, що і метод функціональних точок. Головна відмінність полягає в заміні одиниць вимірювання з функціональних точок на варіанти використання.

Оцінювання методом UCP складається з наступних етапів:

- Оцінювання акторів. На цьому кроці визначаються всі актори системи та кожному з них ставиться у відповідність вага відповідно до його оцінки.
- Нескориговане оцінювання варіантів використання. Оцінка варіантів використання розраховується виходячи з кількості транзакцій.
- Оцінювання технічних чинників. Оцінка технічних факторів використовується для визначення складності архітектури програми. Кожен фактор оцінюється за шкалою від 0 (фактор не значущий) до 5 (фактор надає істотний вплив), після чого множиться на вагу фактора.
- Оцінювання зовнішніх факторів. Даний вид оцінки використовується для визначення коефіцієнта впливу організаційних ризиків на розробку. Обчислення проводяться за аналогією з технічними факторами.
- Остаточний підрахунок. На останньому етапі обчислень оцінюється загальна кількість варіантів з урахуванням інших факторів.

При використанні методу UCP як метрики розміру проекту, обчислення на цьому закінчуються, при використанні як моделі оцінювання трудомісткості потрібно перевести отриманий результат з кількості варіантів використання у трудовитрати у людино-місяцях. На відміну від обчислення кількості елементів, переведення оцінки в людино-місяці може бути досить трудомістким і варіативним завданням, оскільки рекомендовані коефіцієнти переводу (20-28 годин на один елемент) не завжди можуть відповідати реальним трудовитратам. Вибір кількості годин на один елемент повинен залежати від ступеня абстракції при створенні діаграм та досвіду розробки подібних модулів.

В представленій роботі, присвяченій оцінюванню розміру програмних застосунків, в якості метрики розміру обрано кількість рядків коду. Ця метрика являється простим та поширеним способом оцінювання обсягу робіт за проектом, а також є ключовим параметром відомих моделей оцінювання при розробці програмного проекту.

1.2 Основні методи та моделі для оцінювання розміру програмних застосунків

Розмір ПЗ є ключовим параметром будь-якої моделі оцінювання при розробці програмного проекту. Метрика включає кількість рядків програмного коду, функціональних точок та інші [9-11]. Існує безліч різних методів та моделей оцінювання розміру ПЗ [12-14]. Дослідники активно працюють над підвищенням достовірності оцінювання розміру, вдосконалюючи вже існуючі методи та моделі та створюючи нові.

У результаті склалися цілі напрямки оцінювання розміру програмного забезпечення, основні з яких представлені у таблиці 1.1.

Таблиця 1.1 – Основні напрямки оцінювання розміру ПЗ

№ п/п	Назва напрямку	Опис напрямку
1	Експертне оцінювання	Базується на думках експертів, які мають значний досвід у розробці програмного забезпечення у заданій сфері
2	Оцінювання за аналогією	Базується на експертному оцінюванні нового проекту по проектам-аналогам
3	Модельне оцінювання	Базується на математичній моделі та вихідних параметрах
4	Нейронні мережі	Базуються на штучному інтелекті, що самонавчається по проектам-аналогам
5	Регресійний аналіз	Базується на модельному оцінюванні та різновидах регресії
6	Композитне оцінювання	Базується на поєднанні декількох напрямків оцінювання

Серед моделей, які використовуються для оцінювання розміру програмного забезпечення в строках коду, можна виділити регресійні, які досить активно розробляються та вдосконалюються останнім часом. В залежності від виду функції, яка використовується для моделювання зв'язків між залежною змінною та предиктором (предикторами) розрізняють лінійні [2] та нелінійні регресійні моделі [3-7]. Існує ряд припущень, при яких

можливо використовувати лінійну регресійну теорію. Одним з них є нормальний розподіл залежної змінної або залишків з нульовим середнім значенням. Це припущення виконується тільки в окремих випадках. Тому зазвичай будують нелінійні регресійні моделі, застосовуючи лінеарізуючі, нормалізуючі перетворення або метод простого перебору. На практиці досить вживаними є нормалізуючі перетворення, серед яких можна виділити десятковий логарифм, який використовувався при розробці ряду відомих регресійних моделей.

За кількістю предикторів (факторів) регресійні моделі поділяються на однофакторні [3, 4] та багатфакторні [2,5-7]. Багатфакторні моделі зазвичай мають кращу якість в порівнянні з однофакторними.

При аналізі публікацій за темою дослідження не було виявлено регресійної моделі для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій, проте відомо, що розроблені оригінальні моделі мають кращу якість прогнозування.

Все вище перераховане обґрунтовує необхідність побудови багатфакторної нелінійної регресійної моделі оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій з використанням в якості нормалізуючого перетворення десятичного логарифму. В даній роботі буде побудована двохфакторна регресійна модель. В якості предикторів двохфакторної регресійної залежності будемо використовувати кількість класів та середню кількість методів, в якості залежної змінної – розмір застосунку в тисячах строк коду.

Якщо випадкові величини $Y, X_1, X_2, \dots, X_k$ нормально розподілені та $X_1, X_2, \dots, X_k$ – незалежні змінні, то модель лінійної регресії має вигляд [15]:

$$Y = \hat{Y} + \varepsilon = b_0 + b_1X_1 + b_2X_2 + \dots + b_kX_k + \varepsilon, \quad (1.1)$$

де $\hat{Y}$ – результат прогнозування за рівнянням лінійної регресії,

b_i ($i = \overline{0, k}$) – параметри лінійного рівняння регресії,

X_i ($i = \overline{1, k}$) – незалежні змінні,

ε – залишки,

k – кількість предикторів.

Якщо залишки ε мають нормальний розподіл, то довірчий інтервал лінійної регресії визначається за формулою [5, 15]:

$$\hat{Y} \pm t_{\alpha/2, v} S_Y \left\{ \frac{1}{N} + (\mathbf{x}^+)^T [(\mathbf{X}^+)^T \mathbf{X}^+]^{-1} (\mathbf{x}^+) \right\}^{1/2}, \quad (1.2)$$

а інтервал передбачення за формулою [5, 15]:

$$\hat{Y} \pm t_{\alpha/2, v} S_Y \left\{ 1 + \frac{1}{N} + (\mathbf{x}^+)^T [(\mathbf{X}^+)^T \mathbf{X}^+]^{-1} (\mathbf{x}^+) \right\}^{1/2}, \quad (1.3)$$

де $S_Y^2 = \frac{1}{v} \sum_{i=1}^N (Y_i - \hat{Y}_i)^2$,

$v = N - k - 1$,

N – кількість точок багатовимірних даних,

$t_{\alpha/2, v}$ – квантіль t -розподілу Стюдента з $\alpha/2$ рівнем значимості та v степенями вільності,

$(\mathbf{X}^+)^T \mathbf{X}^+ - k \times k$ матриця

$$(\mathbf{X}^+)^T \mathbf{X}^+ = \begin{pmatrix} S_{X_1 X_1} & S_{X_1 X_2} & \dots & S_{X_1 X_k} \\ S_{X_1 X_2} & S_{X_2 X_2} & \dots & S_{X_2 X_k} \\ \dots & \dots & \dots & \dots \\ S_{X_1 X_k} & S_{X_2 X_k} & \dots & S_{X_k X_k} \end{pmatrix}, \quad (1.4)$$

де $S_{X_q X_r} = \sum_{i=1}^N [X_{qi} - \bar{X}_q][X_{ri} - \bar{X}_r]$, $q, r = \overline{1, k}$.

Якщо випадкові величин $Y, X_1, X_2, \dots, X_k$ не підпорядковуються нормальному розподілу, то необхідно будувати нелінійну регресійну модель.

Найбільш вживаним методом побудови нелінійної регресійної моделі є метод нормалізуючих перетворень. Розглянемо основні етапи цього методу [5]:

- вибір нормалізуючого перетворення;
- нормалізація даних;
- вилучення викидів;
- побудова лінійної регресійної моделі для нормалізованих даних;
- побудова довірчого інтервалу та інтервалу передбачення для лінійної регресії;
- перехід до нелінійної регресії через зворотнє перетворення;
- перехід до довірчого інтервалу та інтервалу передбачення для нелінійної регресії через зворотнє перетворення.

Позначимо нормалізуюче перетворення випадкових величин $Y, X_1, X_2, \dots, X_k$, що не підпорядковуються нормальному закону, у нормалізовані дані $Z_Y, Z_1, Z_2, \dots, Z_k$ через ψ . Тоді

$$T = \psi(P), \quad (1.5)$$

де $P = \{Y, X_1, X_2, \dots, X_k\}^T$ – негаусівський випадковий вектор,

$T = \{Z_Y, Z_1, Z_2, \dots, Z_k\}^T$ – гаусівський випадковий вектор,

$\psi = \{\psi_Y, \psi_1, \psi_2, \dots, \psi_k\}^T$ – нормалізуюче перетворення.

Тоді з (1.5) через зворотнє перетворення ψ^{-1} отримаємо:

$$P = \psi^{-1}(T), \quad (1.6)$$

Для $k+1$ нормалізованих величин $Z_Y, Z_1, Z_2, \dots, Z_k$ лінійна регресійна модель має вигляд

$$Z_Y = \hat{Z}_Y + \varepsilon = b_0 + b_1 Z_1 + b_2 Z_2 + \dots + b_k Z_k + \varepsilon, \quad (1.7)$$

довірчий інтервал обчислюється за формулою

$$\hat{Z}_Y \pm t_{\alpha, \frac{v}{2}} S_{Z_Y} \left\{ \frac{1}{N} + (Z_X^+)^T [(Z_X^+)^T Z_X^+]^{-1} (Z_X^+) \right\}^{1/2}, \quad (1.8)$$

а інтервал передбачення представляється співвідношенням

$$\hat{Z}_Y \pm t_{\alpha, \frac{v}{2}} S_{Z_Y} \left\{ 1 + \frac{1}{N} + (Z_X^+)^T [(Z_X^+)^T Z_X^+]^{-1} (Z_X^+) \right\}^{1/2}, \quad (1.9)$$

$$\text{де } S_{Z_Y}^2 = \frac{1}{v} \sum_{i=1}^N (Z_{Y_i} - \hat{Z}_{Y_i})^2.$$

$$v = N - k - 1,$$

$(Z_X^+)^T Z_X^+ - k \times k$ матриця

$$(Z_X^+)^T Z_X^+ = \begin{pmatrix} S_{Z_1 Z_1} & S_{Z_1 Z_2} & \dots & S_{Z_1 Z_k} \\ S_{Z_2 Z_1} & S_{Z_2 Z_2} & \dots & S_{Z_2 Z_k} \\ \dots & \dots & \dots & \dots \\ S_{Z_k Z_1} & S_{Z_k Z_2} & \dots & S_{Z_k Z_k} \end{pmatrix}, \quad (1.10)$$

$$\text{де } S_{Z_q Z_r} = \sum_{i=1}^N [Z_{q_i} - \bar{Z}_q][Z_{r_i} - \bar{Z}_r], \quad q, r = \overline{1, k}.$$

Через зворотнє перетворення з (1.7), (1.8), (1.9) отримуємо нелінійну регресійну модель, її довірчий інтервал та інтервал прогнозування.

В роботі використовується нормалізуюче перетворення десятковий логарифм.

1.3 Обґрунтування необхідності проведення досліджень

У попередньому підрозділі було підкреслено, що серед моделей, які використовуються для оцінювання розміру програмного забезпечення в строках коду, можна виділити регресійні, які досить активно розробляються та вдосконалюються останнім часом. В залежності від виду функції, яка

використовується для моделювання зв'язків між залежною змінною та предиктором (предикторами) розрізняють лінійні [2] та нелінійні регресійні моделі [3-7]. Проте існує ряд припущень, при яких можливо використовувати лінійну регресійну теорію. Одним з них є нормальний розподіл залежної змінної або залишків з нульовим середнім значенням. Однак це припущення виконується тільки в окремих випадках. Тому зазвичай будують нелінійні регресійні моделі, застосовуючи лінеаризуючі, нормалізуючі перетворення або метод простого перебору. На практиці досить вживаними є нормалізуючі перетворення, серед яких можна виділити нормалізуюче перетворення на основі десяткового логарифма, яке використовувалося при розробці ряду відомих регресійних моделей.

Також було визначено, що за кількістю предикторів (факторів) регресійні моделі поділяються на однофакторні [3, 4] та багатофакторні [2,5-7] і що багатофакторні моделі зазвичай мають кращу якість в порівнянні з однофакторними.

При аналізі публікацій за темою дослідження не було виявлено регресійної моделі для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій, проте відомо, що розроблені оригінальні моделі мають кращу якість прогнозування.

Все вище перераховане обґрунтовує необхідність побудови багатофакторної нелінійної регресійної моделі оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій з використанням, в якості нормалізуючого перетворення, перетворення на основі десяткового логарифму. В якості предикторів двохфакторної регресійної залежності будемо використовувати кількість класів та середню кількість методів, в якості залежної змінної – розмір застосунку в тисячах строк коду.

2 НЕЛІНІЙНА РЕГРЕСІЙНА МОДЕЛЬ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНИХ ЗАСТОСУНКІВ НА TYPESCRIPT ДЛЯ КРИПТОВАЛЮТНИХ ОПЕРАЦІЙ

2.1 Виявлення мультиколінеарності та викидів в емпіричних даних

У роботі буде побудовано двохфакторну нелінійну регресійну модель для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій. Для побудови моделі обрано 44 відповідні проекти, з яких зібрано наступні метрики:

- кількість строк коду в тисячах;
- кількість класів;
- середня кількість методів.

Фрагмент зібраних даних представлений в таблиці 2.1.

Таблиця 2.1– Фрагмент емпіричних даних для побудови двохфакторної нелінійної регресійної моделі для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій та квадрат відстані Махаланобіса

№ застосунку	Y (кількість строк коду в тисячах)	X1 (кількість класів)	X2 (середня кількість методів)	SMD (квадрат відстані Махаланобіса)
1	2	3	4	5
1	89,4540	1259	2,7014	1,2664
2	33,1220	390	4,9667	1,1790
3	120,3870	2996	1,2894	14,4173
4	67,9130	480	4,7313	2,9052
5	162,3830	3522	1,6371	17,5986
6	73,4610	612	1,4886	4,6017
7	86,1710	918	2,0871	3,1833
8	75,5870	821	1,7734	2,5014
9	47,6630	653	1,4809	1,8899

Продовження табл. 2.1

1	2	3	4	5
10	76,4680	709	1,7997	3,6202
11	109,5190	1643	2,9440	2,9522
12	86,3830	1117	2,8380	1,3515
13	47,1380	479	5,3466	1,3711
14	95,2260	1211	3,9769	2,5804
15	29,6450	237	4,2236	0,7491
16	80,9290	971	3,1009	1,2332
17	21,8660	118	9,4831	14,1101
18	34,4610	611	2,5843	1,3762
19	46,1230	871	2,3318	1,2303
20	73,7710	2013	1,5713	8,3794

В якості предикторів регресійної залежності будемо використовувати кількість класів та середню кількість методів, а в якості залежної змінної – розмір застосунку в тисячах строк коду.

Також в таблиці представлений квадрат відстані Махаланобіса для наведених емпіричних даних, який розраховується для кожної багатовимірної точки i ($i = \overline{1, N}$) за формулою [16]:

$$d_i^2 = (X_i - \bar{X})^T S_N^{-1} (X_i - \bar{X}), \quad (2.1)$$

де X_i – i -а багатовимірна точка, $X_i = \{X_{1i}, X_{2i}, \dots, X_{mi}\}^T$, m – кількість координат i – ї точки (в випадку двохфакторної моделі X_i має 3 координати),

N – кількість багатовимірних точок,

$\bar{X}$ – середній вектор вибірки,

S_N – коваріаційна матриця:

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

Емпіричні дані не розподілені за нормальним законом. Це підтверджується значеннями квадрату відстані Махаланобіса для 3, 5, 17 та 20 застосунків із наведеного фрагменту таблиці 2.1, які є більшими за квантіль розподілу χ^2 з рівнем значимості 0,05 (7,8147).

Перед побудовою двофакторної регресійної моделі майбутні предиктори необхідно перевірити на наявність мультиколінеарності, так як її відсутність є однією з передумов лінійної множинної регресії. Мультиколінеарність в регресійному аналізі – це наявність лінійної залежності між пояснювальними змінними (факторами). Негативний вплив мультиколінеарності полягає в наступному:

- ускладнюється процедура вибору основних факторів;
- спотворюється зміст коефіцієнта множинної кореляції (він передбачає незалежність факторів);
- ускладнюються обчислення при побудові самої моделі;
- знижується точність оцінки параметрів регресії, спотворюється оцінка дисперсії.

Наслідком зниження точності є ненадійність коефіцієнтів регресії і частково неприйнятність їх використання для інтерпретації як міри впливу пояснювальної змінної на залежну змінну. Оцінки коефіцієнта стають дуже чутливими до вибірових спостережень. Невелике збільшення обсягу вибірки може призвести до дуже сильних змін значення оцінок.

Один із способів визначення мультиколінеарності – це аналіз коефіцієнтів впливу дисперсії (VIFs) проміж незалежних змінних в лінійній моделі множинної регресії. В даному випадку VIFs будуть діагональними елементами оберненої коваріац. матриці 2×2 2-предикторів (1,2221; 1,2221).

VIFs для предикторів кількість класів та середня кількість методів менші ніж 5, це свідчить про відсутність мультиколінеарності [15].

Так як вихідні дані не підпорядковуються нормальному закону розподілу, то необхідно виконати їх нормалізацію та вилучити викиди. В перетворенні

$$(1.5) \quad \psi = \{\psi_Y, \psi_1, \psi_2, \dots, \psi_k\}^T$$

набуде вигляду

$\psi = \{\lg Y, \lg X_1, \lg X_2, \dots, \lg X_k\}^T$. Для вилучення викидів з нормалізованих даних будемо розраховувати SMD для кожної багатовимірної точки i ($i = \overline{1, N}$) за формулою [16]:

$$d_i^2 = (Z_i - \bar{Z})^T S_N^{-1} (Z_i - \bar{Z}), \quad (2.2)$$

де Z_i – i -а багатовимірна точка нормалізованих даних, яка має три координати в двохфакторній моделі

N – кількість багатовимірних точок,

$\bar{Z}$ – середній вектор нормалізованих даних,

S_N – коваріаційна матриця:

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

При підпорядкуванні даних закону багатовимірного нормального розподілу квадрат відстані Махаланобіса не повинен перевищувати $\chi_{m,\alpha}^2$ (квантіль розподілу χ^2 з рівнем значимості α). Ті емпіричні дані, для яких дана умова не виконується, вважаються викидами.

Для кожного d_i^2 розрахуємо також статистику T_{S_i} :

$$T_{S_i} = (N - m) N d_i^2 / ((N^2 - 1) m). \quad (2.3)$$

Якщо дані мають нормальний розподіл, то ті спостереження, для яких статистика T_{S_i} більша, ніж $F_{m, N-m, \alpha}$ (квантіль F -розподілу з рівнем значимості α), вважаються викидами.

При порівнянні статистики (2.3) з квантілем F -розподілу з рівнем значимості 0,05, який в нашому випадку для 44 емпіричних даних дорівнює 2,8327, на першій ітерації було знайдено та вилучено 1 викид. На наступній ітерації, після перерахунку d_i^2 та T_{S_i} , отримані значення T_{S_i} не перевищують квантіль F -розподілу з рівнем значимості 0,05, тобто викиди відсутні.

Фрагмент нормалізованих даних без викидів для побудови двохфакторної нелінійної регресійної моделі для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій та статистика T_{S_i} наведені в таблиці 2.2.

2.2 Розробка двохфакторної нелінійної регресійної моделі оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій

Вихідні дані, які використовуються для побудови двохфакторної нелінійної регресійної моделі для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій, мали негаусівський розподіл. Для побудови нелінійної регресійної моделі з використанням нормалізуючих перетворень, виконано нормалізацію емпіричних даних з допомогою нормалізуючого перетворення на основі десяткового логарифму та вилючено викиди з нормалізованих даних. Фрагмент нормалізованих даних без викидів наведено в таблиці 2.2.

Таблиця 2.2 – Фрагмент нормалізованих даних без викидів для побудови двохфакторної нелінійної регресійної моделі для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій та статистика T_{S_i}

№ застосунку	Z_Y	Z_{X1}	Z_{X2}
1	2	3	4
1	1,9516	3,1000	0,4316
2	1,5201	2,5911	0,6961
3	2,0806	3,4765	0,1104
4	1,8320	2,6812	0,6750
5	2,2105	3,5468	0,2141
6	1,8661	2,7868	0,1728

Продовження табл. 2.2

1	2	3	4
7	1,9354	2,9628	0,3196
8	1,8784	2,9143	0,2488
9	1,6782	2,8149	0,1705
10	1,8835	2,8506	0,2552
11	2,0395	3,2156	0,4689
12	1,9364	3,0481	0,4530
13	1,6734	2,6803	0,7281
14	1,9788	3,0831	0,5995
15	1,4720	2,3747	0,6257
16	1,9081	2,9872	0,4915
17	1,3398	2,0719	0,9769
18	1,5373	2,7860	0,4123
19	1,6639	2,9400	0,3677
20	1,8679	3,3038	0,1963

Лінійна регресійна модель оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій для нормалізованих з допомогою десяткового логарифму даних будується за формулою (1.7), яка у випадку двох факторів має вигляд:

$$Z_Y = \hat{Z}_Y + \varepsilon = b_0 + b_1 Z_1 + b_2 Z_2 + \varepsilon, \quad (2.4)$$

де $Z_Y = \lg Y$; $Z_1 = \lg X_1$; $Z_2 = \lg X_2$; ε – випадкова похибка.

Параметри b_0, b_1, b_2 були знайдені методом найменших квадратів. Вони мають наступні значення: $b_0 = -0,9374$, $b_1 = 0,8751$ та $b_2 = 0,3723$. Після підстановки параметрів (2.4) прийме вигляд:

$$Z_Y = \hat{Z}_Y + \varepsilon = -0,9374 + 0,8751 Z_1 + 0,3723 Z_2 + \varepsilon, \quad (2.5)$$

Тобто, лінійна регресійна модель оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій для нормалізованих даних має вигляд (2.5),

Для теоретичного обґрунтування використання лінійної регресійної моделі необхідно перевірити гіпотезу H_0 , яка полягає в тому, що випадкова величина ε із (2.5) розподілена за нормальним законом.

Перевірка початкової гіпотези про передбачуваний закон розподілу здійснюється на основі критерію згоди. Існує кілька критеріїв згоди: критерій Смирнова, критерій Колмогорова, критерій χ^2 Пірсона та інші. Критерії згоди дозволяють відкинути або підтвердити правильність висунутої гіпотези про характер розподілу емпіричних даних.

Для перевірки гіпотези H_0 будемо використовувати критерій χ^2 Пірсона, який застосовують, якщо об'єм вибірки достатньо великий ($n > 30$).

Згідно відомої схеми використання критерію χ^2 , після висування нульової гіпотези про нормальний розподіл випадкової величини ε необхідно обчислити випадкову величину χ^2 за наступною формулою [17]:

$$\chi^2 = \sum_{j=1}^m \frac{(n_j - np_j)^2}{np_j}, \quad (2.6)$$

де m – кількість підінтервалів розбиття інтервалу випадкової величини,

p_j – ймовірність попадання випадкової величини в підінтервал j ,

n_j – емпірична частота в підінтервалі j .

Кількість підінтервалів m визначається зазвичай або за формулою Старджеса: $m = 5 \log_2 n + 1 = 3,3 \lg n + 1$, або Брукса і Карузера: $m = 5 \lg n$. В роботі за формулою Старджеса розбили інтервал випадкової величини на 5 підінтервалів.

Ймовірність попадання випадкової величини в підінтервал j визначається за формулою:

$$p_j = \int_{x_{j-1}}^{x_j} f(x) dx,$$

де $f(x)$ – функція щільності ймовірності нормального розподілу,

x_{j-1}, x_j – границі підінтервалу j .

Формула для обчислення функції щільності ймовірності має вигляд:

$$f(x) = \frac{1}{\sigma_x \sqrt{2\pi}} e^{-\frac{(x-m_x)^2}{2\sigma_x^2}},$$

де σ_x – вибіркове середньоквадратичне відхилення,

m_x – математичне сподівання.

На наступному кроці по таблиці χ^2 Пірсона знаходимо критичне значення $\chi_{кр}^2$, яке залежить від двох параметрів α та ν , де α – заданий рівень значимості (у нас 0,05, тобто з ймовірністю 0,95 можна гарантувати прийняття чи спростування гіпотези), ν – число степеней вільності. ν знаходиться по формулі $\nu = m - k - 1$, де k – число параметрів розподілу (для нормального розподілу $k = 2$).

Після знаходження $\chi_{кр}^2$, необхідно порівняти χ^2 та $\chi_{кр}^2$. Якщо χ^2 менше $\chi_{кр}^2$, то нульова гіпотеза приймається, в протилежному випадку – спростовується. Отримали $\chi^2 = 4,2853$, а $\chi_{кр}^2 = 5,9915$, тобто приймається нульова гіпотеза про нормальний розподіл випадкової величин ε . Це теоретично обґрунтовує використання лінійної регресійної моделі.

Так як випадкова величина ε має нормальний розподіл, то довірчий інтервал та інтервал передбачення лінійної регресії (2.5) обчислюємо загальноприйнятим способом, скориставшись співвідношеннями (1.8) та (1.9) відповідно.

Застосувавши до (2.5) зворотне нормалізуюче перетворення, отримаємо для емпіричних даних двохфакторну нелінійну регресійну модель оцінювання розміру програмних застосунків на TYPEScript для криптовалютних операцій:

$$Y = 10^{\varepsilon - 0,9374} X_1^{0,8751} X_2^{0,3723}. \quad (2.6)$$

Також через зворотне нормалізуюче перетворення знаходимо границі довірчого інтервалу та інтервалу передбачення для побудованої двохфакторної нелінійної регресії (2.6).

При аналізі публікацій за темою дослідження не були виявлені ні двохфакторні регресійні методи чи моделі для оцінювання розміру програмного забезпечення, ні регресійні моделі для оцінювання розміру програмних застосунків на TYPEScript для криптовалютних операцій. Тому для порівняльного аналізу за формулою (1.1) побудовано двохфакторну лінійну регресійну модель для емпіричних даних у припущенні про їх гаусівський розподіл:

$$Y = \hat{Y} + \varepsilon = b_0 + b_1 X_1 + b_2 X_2 + \varepsilon. \quad (2.7)$$

Параметри b_0 , b_1 , b_2 для моделі (2.7) визначені методом найменших квадратів та мають значення: $b_0 = 18,5817$, $b_1 = 0,0429$ та $b_2 = 1,1471$. Після підстановки параметрів, (2.7) набуває вигляду:

$$Y = \hat{Y} + \varepsilon = 18,5817 + 0,0429 X_1 + 1,1471 X_2 + \varepsilon.$$

Скориставшись співвідношеннями (1.2) та (1.3) побудовано довірчий інтервал та інтервал передбачення двохфакторної лінійної регресійної моделі оцінювання розміру програмних застосунків на TYPEScript для криптовалютних операцій.

Фрагменти границь довірчих інтервалів та їх довжин для лінійної і нелінійної двохфакторних регресійних моделей наведені в таблиці 2.3.

Таблиця 2.3 – Границі довірчих інтервалів та їх довжини для лінійної і нелінійної двохфакторних регресійних моделей

№ п/п	Границі та довжини довірчих інтервалів					
	Двохфакторна лінійна регресія (емпіричні дані)			Двохфакторна нелінійна регресія (нормалізовані дані)		
	Нижня	Верхня	Довжина	Нижня	Верхня	Довжина
1	69,8155	81,7578	11,9422	78,7845	95,7694	16,9848
2	34,4596	47,6198	13,1603	35,4253	43,1031	7,6779
3	132,3306	165,2998	32,9692	118,2393	167,7695	49,5303
4	38,5391	50,7357	12,1966	41,8489	50,6131	8,7642
5	151,4678	192,1706	40,7028	148,9845	211,1070	62,1224
6	54,1351	66,7198	12,5846	54,5580	65,6583	11,1003
7	48,7658	63,0322	14,2663	45,6253	57,1987	11,5734
8	40,0045	56,6825	16,6780	33,7009	45,3542	11,6533
9	43,7595	58,4723	14,7128	40,2827	50,6693	10,3866
10	84,6477	100,4874	15,8397	100,0220	128,1550	28,1330
11	64,3818	75,2999	10,9181	72,7528	87,2588	14,5060
12	37,9945	52,6061	14,6116	42,9977	53,7595	10,7619
13	68,6633	81,7108	13,0475	85,3716	110,1231	24,7515
14	27,4365	39,7877	12,3512	21,4378	26,4019	4,9640
15	58,8320	68,9042	10,0723	66,8582	79,3760	12,5178
16	16,1103	52,9524	36,8421	14,4001	21,1803	6,7802
17	41,8258	53,7829	11,9571	42,1040	48,9190	6,8149
18	52,8156	64,5610	11,7454	54,8499	64,6949	9,8449
19	96,6926	117,0949	20,4022	92,9787	123,2472	30,2685
20	17,4321	46,9908	29,5586	11,5703	16,8596	5,2893

Фрагменти границь інтервалів передбачення та їх довжин для лінійної і нелінійної двохфакторних регресійних моделей наведені в таблиці 2.4.

Таблиця 2.4 – Границі інтервалів передбачення та їх довжини для лінійної і нелінійної двохфакторних регресійних моделей

№ п/п	Границі та довжини інтервалів передбачення					
	Двохфакторна лінійна регресія (емпіричні дані)			Двохфакторна нелінійна регресія (нормалізовані дані)		
	Нижня	Верхня	Довжина	Нижня	Верхня	Довжина
1	43,6582	107,9151	64,2568	55,2856	136,4757	81,1901
2	8,7925	73,2869	64,4943	24,8682	61,4012	36,5330
3	113,2017	184,4287	71,2270	87,6265	226,3807	138,7542
4	12,4851	76,7897	64,3046	29,3081	72,2702	42,9622
5	134,2591	209,3793	75,1202	110,3642	284,9806	174,6164
6	28,2378	92,6171	64,3793	38,1339	93,9370	55,8031
7	23,5344	88,2635	64,7291	32,3981	80,5513	48,1532
8	15,6920	80,9950	65,3030	24,5460	62,2700	37,7241
9	18,7014	83,5303	64,8289	28,6401	71,2671	42,6270
10	60,0206	125,1145	65,0939	71,5999	179,0271	107,4272
11	37,8037	101,8781	64,0744	50,7828	125,0093	74,2265
12	12,8973	77,7033	64,8060	30,5013	75,7849	45,2836
13	42,9514	107,4228	64,4714	61,2621	153,4616	92,1995
14	1,4450	65,7791	64,3341	15,1201	37,4335	22,3135
15	31,9003	95,8360	63,9357	46,4776	114,1827	67,7051
16	-2,0188	71,0816	73,1004	10,7905	28,2654	17,4749
17	15,6745	79,9341	64,2596	29,0110	70,9966	41,9856
18	26,5780	90,7985	64,2205	38,0286	93,3115	55,2828
19	73,7178	140,0697	66,3519	67,3684	170,0999	102,7315
20	43,6582	107,9151	69,7140	8,6456	22,5629	13,9174

Аналіз ширин довірчих інтервалів та інтервалів передбачення виявив звуження відповідних інтервалів в більшості випадків для нелінійної двохфакторної регресії в порівнянні з лінійною.

Для побудованих двохфакторних нелінійної та лінійної регресійних моделей визначено R^2 (Coefficient of determination), $MMRE$ (Mean magnitude

of relative error) та $PRED(0,25)$ (Prediction Level) [16-19]. Результати обчислень наведені в таблиці 2.5.

Таблиця 2.5 – R^2 (Coefficient of determination), $MMRE$ (Mean magnitude of relative error) та $PRED(0,25)$ (Prediction Level) двохфакторних нелінійної та лінійної регресійних моделей оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій

Модель	R^2 (Coefficient of determination)	$MMRE$ (Mean magnitude of relative error)	$PRED(0,25)$ (Prediction Level)
Лінійна	0,7805	0,3378	0,5581
Нелінійна	0,8698	0,1652	0,7209

Аналіз R^2 , $MMRE$ та $PRED(0,25)$ з таблиці 2.4 свідчить про кращу якість нелінійної моделі.

З ПРОЕКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНИХ ЗАСТОСУНКІВ НА TYPESCRIPT ДЛЯ КРИПТОВАЛЮТНИХ ОПЕРАЦІЙ

3.1 Постановка задачі на розробку ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій

Провівши дослідження існуючих методів та моделей для оцінювання розміру програмного забезпечення та побудувавши двохфакторну нелінійну регресійну модель для прогнозування розміру програмних застосунків на TypeScript для криптовалютних операцій, необхідно перейти до розробки програмного забезпечення, яке автоматизує процес оцінювання розміру згідно розробленої моделі. У якості факторів в моделі використовуються: кількість класів та середня кількість методів, а в якості залежної змінної – кількість строк коду в тисячах

У розроблюваному програмному забезпеченні повинні бути реалізовані наступні функції:

- Занесення даних.
- Зміна даних.
- Прогнозування кількості сторок коду.
- Розрахунок верхньої та нижньої меж довірчого інтервалу.
- Розрахунок верхньої та нижньої меж інтервалу прогнозування.

Розгорнута постановка задачі сформульована в технічному завданні, яке наведене в додатку А.

3.2 Розробка ескізного проекту ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій

Ескізний проект повинен містити попередні загальносистемні рішення щодо обраного на етапі концептуального проектування варіанту програмної системи та окремих видів її забезпечення, достатні для розробки технічного проекту [20].

Основою для ескізного проектування є документи, розроблені на передпроектній стадії: звіт про передпроектне обстеження, технічне завдання.

Зміст робіт на етапі ескізного проектування: побудова контекстної діаграми ПЗ; попередня розробка структури вхідних та вихідних даних; розробка проекту інтерфейсу користувача, розробка моделі переходів станів.

3.2.1 Розробка контекстної діаграми ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій

Контекстна діаграма – це діаграма, розташована на вершині деревоподібної структури діаграм FD, що є найзагальнішим описом системи та її взаємодії із зовнішнім середовищем (як правило, тут описується основне призначення модельованого об'єкта). Контекстна діаграма містить один блок, що описує функцію верхнього рівня, її входи, виходи та зовнішні по відношенню до системи сутності.

Контекстна діаграма ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій представлена на рисунку 3.1.

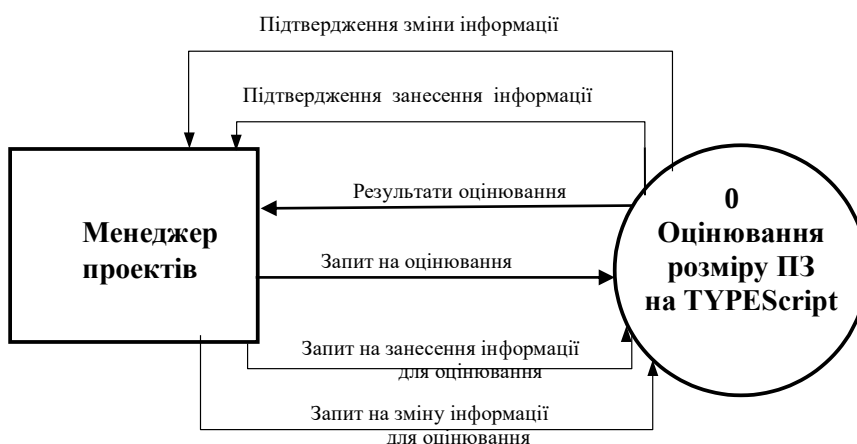


Рисунок 3.1 – Контекстна діаграма ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій

Працювати з даним програмним забезпеченням буде менеджер проектів. Він буде мати можливість послати в контекстний процес запити на оцінювання розміру ПЗ на TypeScript для криптовалютних операцій, на занесення чи зміну інформації для оцінювання. У відповідь менеджер буде отримувати результати оцінювання, підтвердження занесення інформації, підтвердження зміни інформації.

Для деталізації потоків даних контекстної діаграми ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій розроблений словник даних, який представлений в таблиці 3.1.

Таблиця 3.1 – Словник даних контекстної діаграми ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій

Ім'я потоку	Тип потоку	Атрибути потоку
1	2	3
Запит на оцінювання розміру ПЗ на TypeScript для криптовалютних операцій	Складовий	Запит на отримання прогнозованої кількості строк коду Запит на довірчий інтервал Запит на інтервал передбачення

Продовження табл. 3.1

1	2	3
Запит на занесення інформації для оцінювання	Простий	Назва програмного застосунку, кількість класів та середня кількість методів
Запит на зміну інформації для оцінювання	Простий	Назва програмного застосунку, кількість класів та середня кількість методів
Результат оцінювання розміру ПЗ на TypeScript для криптовалютних операцій	Складовий	Прогнозна кількість строк коду Довірчий інтервал Інтервал передбачення
Підтвердження занесення інформації для оцінювання	Простий	Назва програмного застосунку, кількість класів та середня кількість методів
Підтвердження зміни інформації для оцінювання	Простий	Назва програмного застосунку, кількість класів та середня кількість методів

Наступним етапом ескізного проектування є розробка концептуальної моделі даних ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій.

3.2.2 Розробка концептуальної моделі даних ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій

Побудуємо загальну модель інформаційної системи – концептуальну модель. Концептуальна модель представляє об'єкти і їхні взаємозв'язки без вказування способів їхнього фізичного збереження.

Концептуальну модель даних можна представити з допомогою діаграми «сутність-зв'язок» (ER-діаграми). Для її побудови необхідно визначити об'єкти (сутності), їхні властивості (атрибути) і відношення між об'єктами

(зв'язки), які являються важливими для даної предметної галузі. ER-діаграма використовується для проектування реляційних БД.

Перший крок побудови моделі – визначення об'єктів.

Об'єктами в даній моделі будуть:

- «Програмні застосунки»;
- «Метрики»;
- «Результати оцінювання».

На другому кроці побудови моделі визначимо відношення між об'єктами.

Опис зв'язків між об'єктами представлений нижче:

Зв'язок між об'єктами «Програмні застосунки» та «Метрики» – 1:1.

Зв'язок між об'єктами «Результати оцінювання» та «Метрики» – 1:1.

На третьому кроці побудови моделі визначимо атрибути об'єктів:

- «Програмні застосунки»: код застосунку, назва застосунку;
- «Метрики»: код метрик застосунку, кількість класів, середня кількість методів.
- «Результати оцінювання»: код результатів оцінювання, розмір застосунку, min знач. інтервалу прогнозування, max знач. інтервалу прогнозування, min знач. довірчого інтервалу, max знач. довірчого інтервалу.

Концептуальна модель даних ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій представлена на рисунку 3.2.

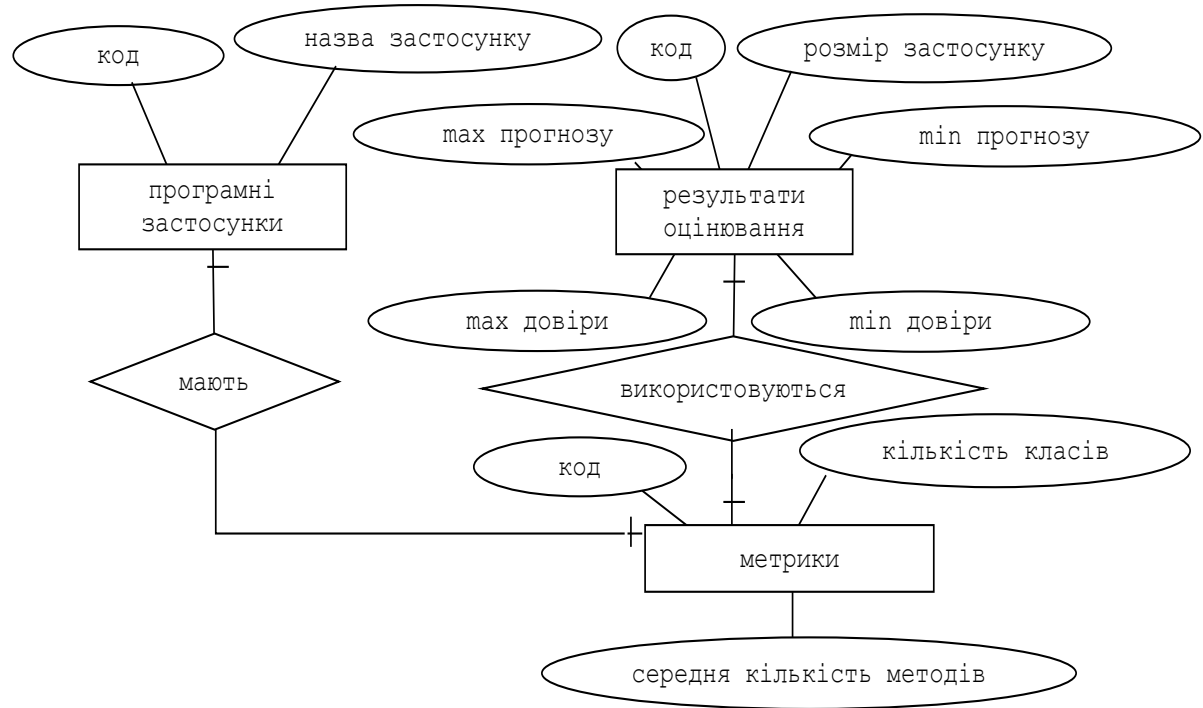


Рисунок 3.2 – Концептуальна модель даних ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій

3.2.3 Розробка моделі переходів станів ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій

Модель переходів станів показує стани інформаційного об'єкта, можливі переходи між ними та їх умови.

Модель переходів станів ПЗ оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій представлена на рисунку 3.3.

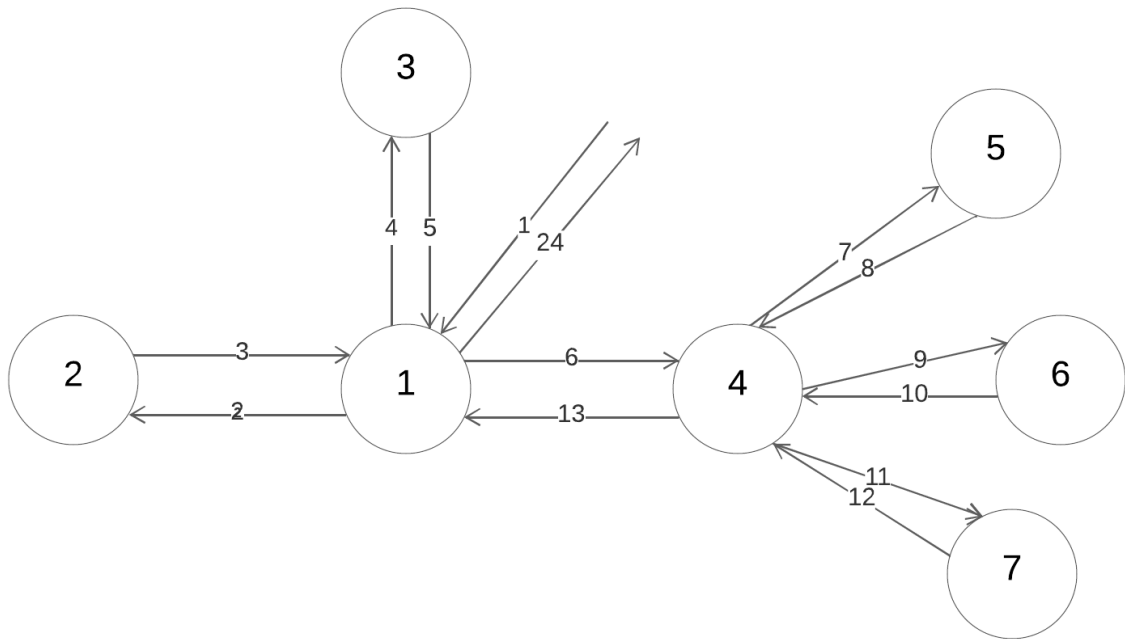


Рисунок 3.3 – Модель переходів станів ПЗ оцінювання розміру програмних застосунків на TYPEScript для криптовалютних операцій

В моделі можна виділити наступні стани:

1 – головне меню;

2 – занесення інформації для оцінювання;

3 – зміна інформації для оцінювання;

4 – оцінювання розміру програмних застосунків на TYPEScript для криптовалютних операцій;

5 – довірчий інтервал;

6 – інтервал прогнозування;

7 – розмір застосунку.

Розглянемо події на переходах:

1 – вхід у меню;

2 – вибір занесення інформації для оцінювання;

4 – вибір зміни інформації для оцінювання;

6 – вибір оцінювання розміру програмних застосунків на TYPEScript для криптовалютних операцій;

- 7 – вибір розрахунку довірчого інтервалу;
- 9 – вибір розрахунку інтервалу прогнозування;
- 11 – вибір розрахунку розміру застосунку;
- 3, 5, 8, 10,12,13 – повернення до вибору;
- 14 – завершення роботи програми.

На етапі аналізу вимог та визначення специфікацій діаграма переходів станів демонструє поведінку програмної системи, що розробляється, при отриманні управляючих впливів.

3.2.4 Розробка інтерфейсу користувача ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій

Інтерфейс користувача – це інтерфейс, за допомогою якого людина може керувати програмним забезпеченням або апаратним оснащенням. Інтерфейс користувача повинен бути зручними у використанні, щоб взаємодія з ним відбувалася на максимально інтуїтивному рівні. Інтерфейси програмного забезпечення також називають графічними інтерфейсами користувача

Графічний інтерфейс користувача є найбільш популярним інтерфейсом. Він є вікном, в якому містяться різні елементи управління. Взаємодія користувача з програмою відбувається за допомогою миші та за допомогою клавіатури.

Також є можливість використовувати кнопки. Це вікно є чимось на зразок шлюзу між користувачем і програмним забезпеченням. У графічному інтерфейсі користувача поширені типові елементи керування. Вони дозволяють стандартизувати процес взаємодії з різними програмами у різних операційних системах.

Для роботи з програмним забезпеченням будуть задіяні наступні кнопки:
New, Edit, Run, Exit:

- New – для введення нових вихідних даних;

- Edit – для корегування введених даних;
- Run – для прогнозування;
- Exit – для завершення роботи.

На рисунку 3.4 представлено вікно для занесення інформації для прогнозування.

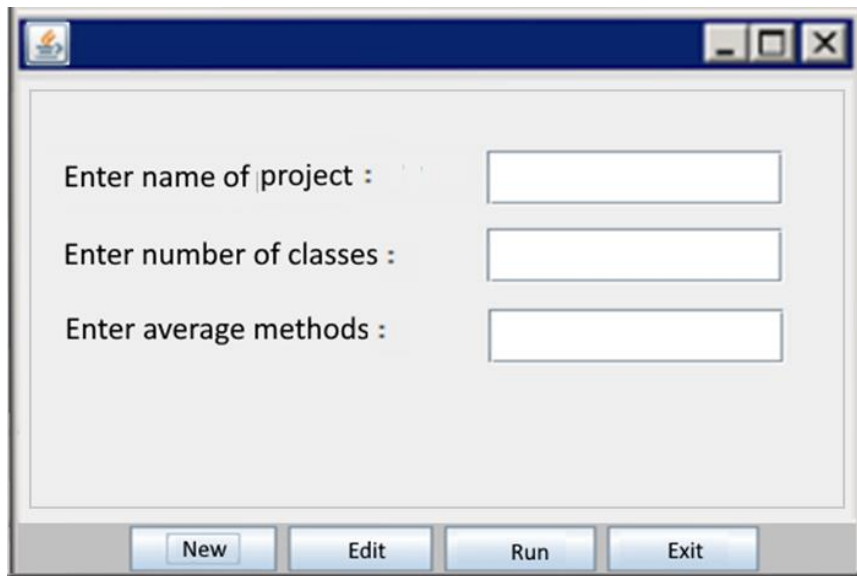


Рисунок 3.4 – Вікно для занесення інформації для прогнозування

В вікні необхідно ввести наступну інформацію: назву проекту для оцінювання (name of project), число класів (number of classes), середню кількість методів (average methods).

На рисунку 3.5 представлено вікно для виведення результатів прогнозування. В вікні з результатами прогнозування буде представлена інформація про прогнозований розмір програмного застосунку на TypeScript для криптовалютних операцій (thousand lines of code) в тисячах строк коду (KLOC), довірчий інтервал (confidence interval) та інтервал прогнозування (prediction interval).

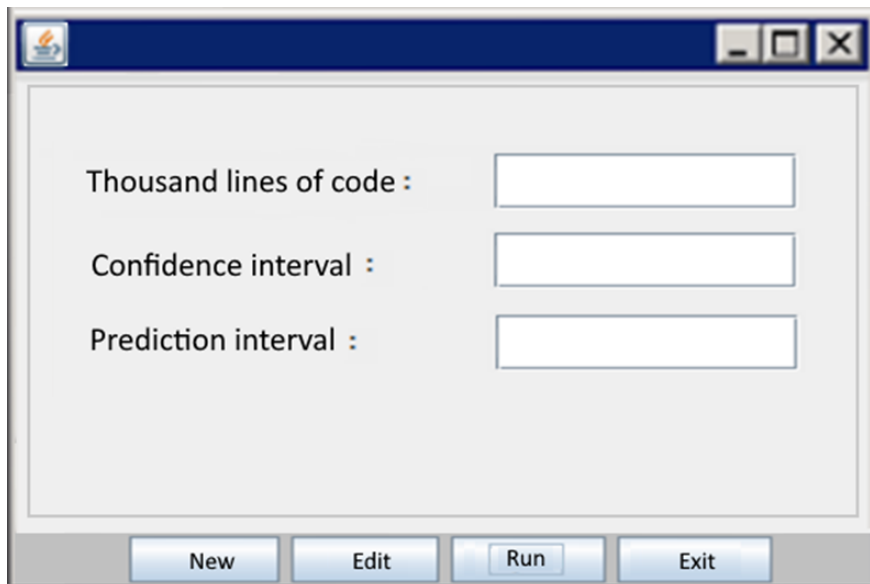


Рисунок 3.5 – Вікно для виведення результатів прогнозування

Таким чином, був побудований проект користувацького інтерфейсу.

3.3 Розробка технічного проекту програмного забезпечення для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій

В рамках технічного проекту виконується наступний перелік робіт [20]:

- розробляється алгоритм роботи з даними з урахуванням обраної моделі даних;
- формується ієрархічна структура ПЗ на основі декомпозиції складних процесів;
- розробляється специфікація програмного забезпечення.

Рішення, що прийняті на даному етапі, значною мірою визначаються обраними засобами проектування, операційним середовищем, технологією робіт, вимогами до технічного забезпечення обробки інформації.

3.3.1 Розробка ієрархії діаграм потоків даних ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій

Контекстний процес програмного забезпечення для прогнозування розміру програмних застосунків на TypeScript для криптовалютних операцій, що наведений на рисунку 3.1, буде деталізуватися з допомогою діаграм потоків даних (DFD) до підпроцесів, які підлягають реалізації. При деталізації повинні виконуватись наступні правила:

- правило балансування – при деталізації процесу деталізуюча діаграма як зовнішні джерела/приймачі даних може мати тільки ті компоненти (процеси, зовнішні сутності, накопичувачі даних), з якими має інформаційний зв'язок деталізований процес на батьківській діаграмі;

- правило нумерації – при деталізації процесів має підтримуватися їх ієрархічна нумерація. Наприклад, процеси, що деталізують процес із номером 1, отримують номери 1.1, 1.2, 1.3 тощо.

Деталізація контекстного процесу ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій наведена на рисунку 3.5, на якому представлена діаграма потоків даних першого рівня.

Процес «Оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій» деталізується наступними підпроцесами: 1 – «Занесення інформації для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій»; 2 – «Зміна інформації для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій»; 3 – «Оцінювання». Процеси 1,2 являються простими. Для них після побудови ієрархії діаграм потоків даних необхідно розробити специфікації. Процес 3 – «Оцінювання» являється складним та потребує подальшої деталізації.

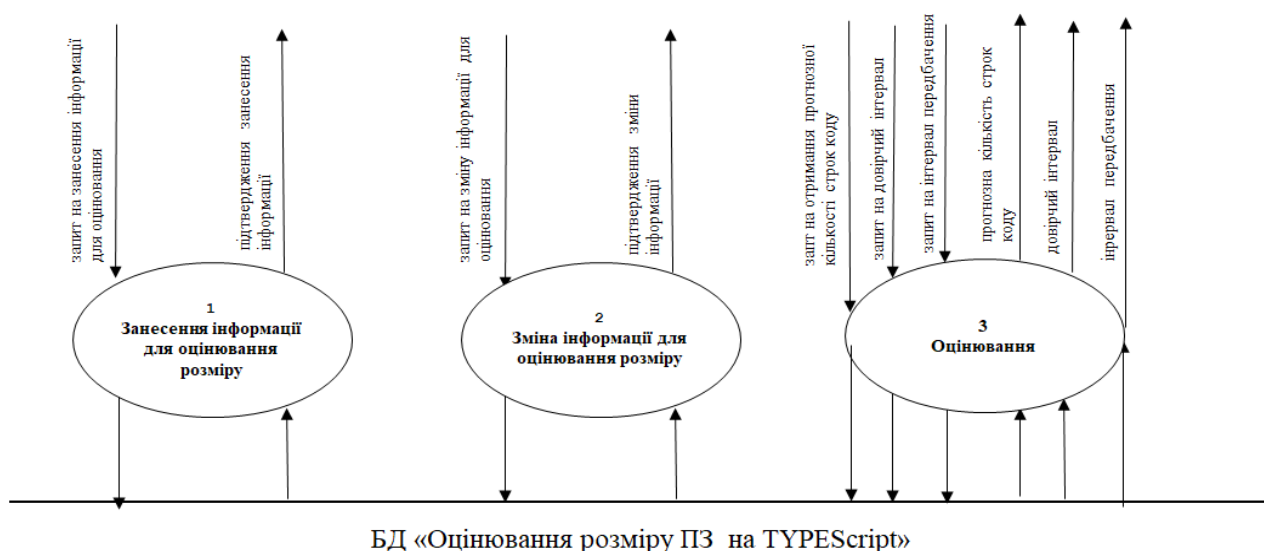


Рисунок 3.5 – Деталізація контекстного процесу «Оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій»

Виконаємо деталізацію процесу 3 – «Оцінювання». Вказаний процес деталізується підпроцесами: 3.1 «Отримання прогнозованої кількості строк коду», 3.2 «Отримання довірчого інтервалу», 3.3 «Отримання інтервалу передбачення». Деталізація процесу 3 «Оцінювання» представлена на діаграмі потоків даних другого рівня (рис. 3.6). Всі підпроцеси на рисунку 3.6 являються простими.

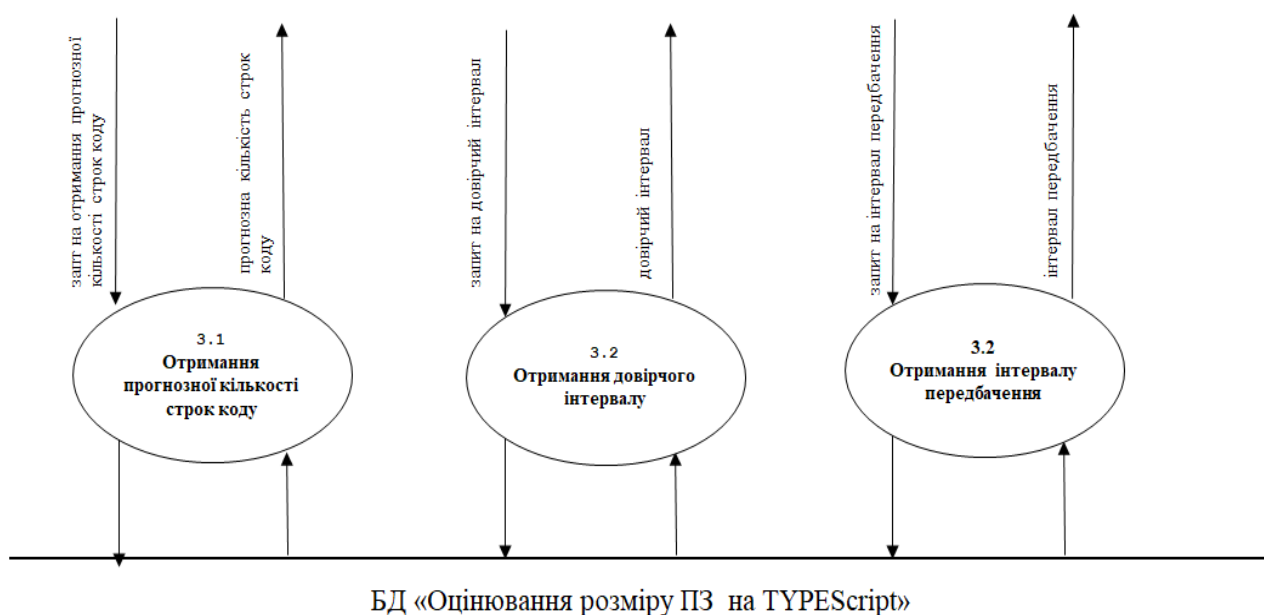


Рисунок 3.6 – Деталізація процесу «Оцінювання»

Відповідність потоків на діаграмах потоків даних 0, 1 та 2 рівнів представлена в таблиці 3.2.

Таблиця 3.2 – Таблиця відповідності потоків на діаграмах потоків даних 0, 1 та 2 рівнів

Потоки на DFD 0-го рівня	Потоки на DFD 1-го рівня	Потоки на DFD 2-го рівня	Складові потоків
1	2	3	4
Запит на оцінювання розміру ПЗ на TYPEScript для криптовалютних операцій	Запит на отримання прогнозованої кількості строк коду	Запит на отримання прогнозованої кількості строк коду	запит
	Запит на довірчий інтервал	Запит на довірчий інтервал	запит
	Запит на інтервал прогнозування	Запит на інтервал прогнозування	запит
Запит на занесення інформації для оцінювання	Запит на занесення інформації для оцінювання	—	Назва застосунку, кількість класів та середня кількість методів
Запит на зміну інформації для оцінювання	Запит на зміну інформації для оцінювання	—	Назва застосунку, кількість класів та середня кількість методів

Продовження табл. 3.2

1	2	3	4
Результат оцінювання розміру ПЗ на TypeScript для криптовалютних операцій	Прогнозна кількість строк коду	Прогнозна кількість строк коду	Кількість строк коду
	Довірчий інтервал	Довірчий інтервал	Межі довірчого інтервалу
	Інтервал передбачення	Інтервал передбачення	Межі інтервалу передбачення
Підтвердження занесення інформації для оцінювання	Підтвердження занесення інформації для оцінювання	—	Назва застосунку, кількість класів та середня кількість методів
Підтвердження зміни інформації для оцінювання	Підтвердження зміни інформації для оцінювання	—	Назва застосунку, кількість класів та середня кількість методів

Таким чином, побудували ієрархію діаграм потоків даних, в якій виділили 5 простих підпроцесів, для яких необхідно розробити мініспецифікації.

Мініспецифікація процесу 1 «Занесення інформації для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій»

Вхідні дані: назва застосунку, кількість класів, середня кількість методів.

Послідовність дій:

- вивести форму для введення інформації;
- ввести інформацію для оцінювання;
- вивести введену інформацію для оцінювання на екран.

Вихідні дані: назва застосунку, кількість класів, середня кількість методів.

Мініспецифікація процесу 2 «Зміна інформації для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій»

Вхідні дані: назва застосунку, кількість класів, середня кількість методів.

Послідовність дій:

- вивести існуючу інформацію для оцінювання (кількість класів, середня кількість методів);
- ввести зміни в інформацію для оцінювання;
- вивести змінену інформацію для оцінювання на екран.

Вихідні дані: назва застосунку, кількість класів, середня кількість методів.

Мініспецифікація процесу 3.1 «Отримання прогнозованої кількості строк коду»

Вхідні дані: запит на отримання прогнозованої кількості строк коду.

Послідовність дій:

- сформулювати запит до відповідних довідників;
- розрахувати прогнозоване значення;
- вивести прогнозоване значення на екран.

Вихідні дані: прогнозоване значення кількості строк коду.

Мініспецифікація процесу 3.2 «Отримання довірчого інтервалу»

Вхідні дані: запит на довірчий інтервал.

Послідовність дій:

- сформулювати запит до відповідних довідників;
- розрахувати межі довірчого інтервалу;

- вивести межі довірчого інтервалу на екран.

Вихідні дані: довірчий інтервал.

Мініспецифікація процесу 3.3 «Отримання інтервалу передбачення»

Вхідні дані: запит на інтервал передбачення.

Послідовність дій:

- сформулювати запит до відповідних довідників;
- розрахувати межі інтервалу передбачення;
- вивести межі інтервалу передбачення на екран.

Вихідні дані: інтервал передбачення.

3.3.2 Структура представлення даних ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій

В якості моделі даних для ПЗ оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій була обрана реляційна модель. В ній сутності представляються у вигляді таблиць, які можуть бути пов'язані між собою зв'язками 1:1 та 1:N. Атрибутом у таблиці є будь-який стовпець, а записом (кортежем) – рядок таблиці. Ключем є стовпець, значення якого однозначно визначає рядок таблиці.

Кожна сутність має ключове поле, що являється первинним ключем, або ж ключове поле є частиною складеного первинного ключа. Таблиці пов'язуються між собою за допомогою механізму зовнішнього ключа [21].

Логічна модель ПЗ представлена на рис. 3.7.

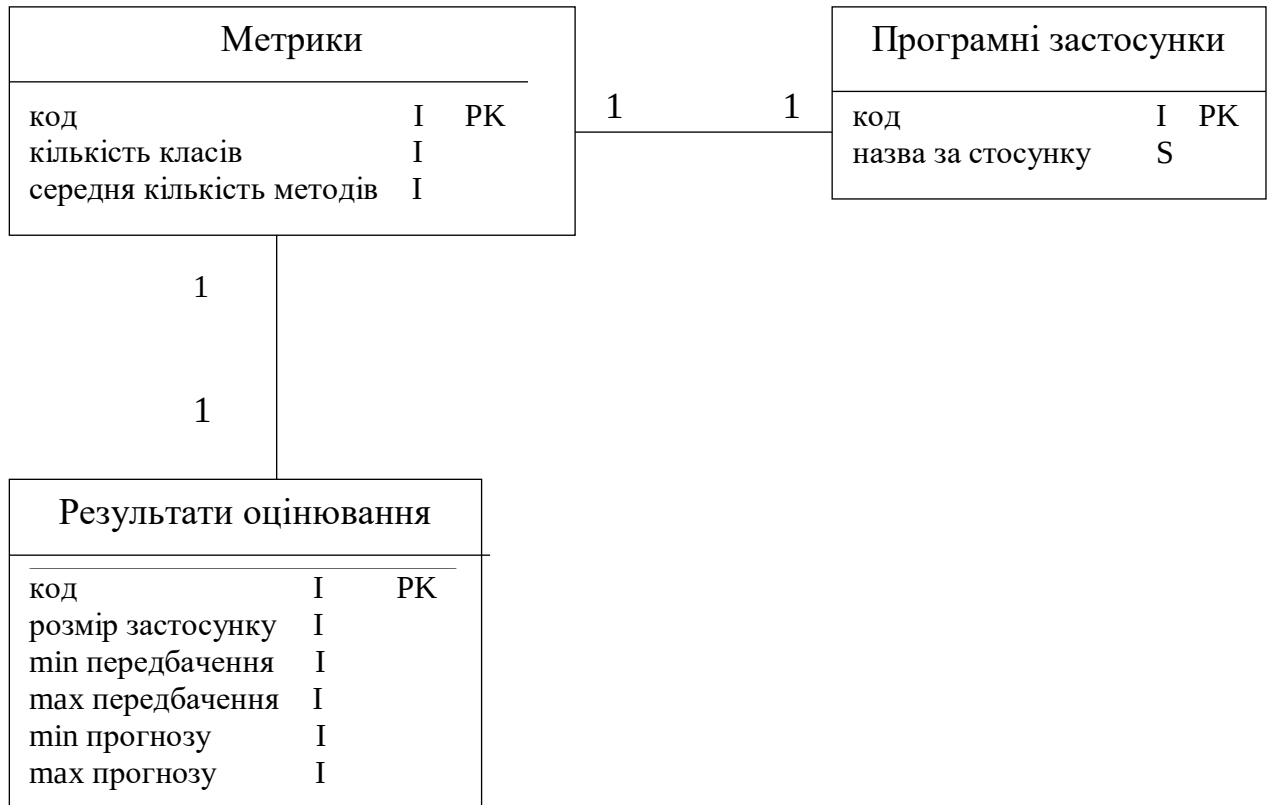


Рисунок 3.7 – Логічна модель даних ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій

Представлена модель даних задовольняє умовам третьої нормальної форми [21].

3.4 Розробка робочого проекту ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій

На етапі робочого проектування розробники зобов'язані закінчити розробляти ПЗ, завершити тестування, випробування та завершити робочу документацію. Цей етап вимагає узгоджених та координуваних дій всіх розробників [20].

3.4.1 Вибір мови програмування та засобів розробки ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій

Java – популярна мова програмування, яка має наступні технічні особливості та переваги: простота, об’єктно-орієнтований підхід, безпека, продуктивність, надійність, незалежність від апаратної частини та ОС, динамічність та адаптованість, зручні та ефективні мережеві можливості. Не треба забувати, що Java – це тріо, що складається з мови програмування, потужного універсального обробника та значної бібліотеки. Всі ці напрацювання доступні програмістам. Їм не потрібно розробляти з нуля багато необхідних процедур (доступ до мережі, баз даних тощо).

Звичайно, недоліки у Java теж є. Наприклад, платне комерційне користування. Але на тлі таких значних переваг мінуси виглядають лише причіпками. А плюси роблять мову незамінною і перспективною.

Java найбільш затребувана у таких напрямках, як

- веб-розробка (масштабні бізнес-проекти);
- програми для ПК (десктопний софт);
- комп’ютерні ігри (наприклад, Minecraft);
- програми для мобільних пристроїв (ОС Android);
- наукові дослідження та розробки;
- промисловий програмінг.

Виходячи з вищевикладеного, в якості мови програмування була обрана Java.

Однією з найпопулярніших СКБД на сьогодні є MySQL, яка вільно розповсюджується.

Примітно, що результатом величезної поширеності MySQL стала поява в інтернеті безлічі посібників із освоєння системи, а також величезної кількості різноманітних плагінів та розширень, що спрощують роботу з MySQL.

Переваги MySQL:

- простота у використанні;
- великий функціонал;

- безпека;
- масштабованість;
- швидкість.

До недоліків цієї СУБД належать: недостатня надійність, низька швидкість розробки. Як і багатьом іншим програмним продуктам з відкритим кодом, MySQL не вистачає деякої технічної досконалості, що часом позначається на ефективності процесів розробки.

Але переваг у MySQL набагато більше чим недоліків. Тому в якості СКБД обрали MySQL.

3.4.2 Кодування та тестування ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій

Програмне забезпечення написане на мові Java. Для створення та роботи з базою обрали MySQL. Текст програми наведено в додатку Д. Інструкція користувача знаходиться в додатку Г, Опис програми в додатку Б.

Розглянемо тестування ПЗ. В роботі використовувався один із методів «чорного ящика», а саме метод еквівалентного розбиття

Ми розділяємо функціонал (часто діапазон можливих значень, що вводяться) на групи еквівалентних за своїм впливом на систему значень. Такий поділ допомагає переконатися у правильному функціонуванні цілої системи – одного класу еквівалентності, перевібивши лише один елемент цієї групи. Ця техніка полягає у розбитті всього набору тестів на класи еквівалентності з подальшим скороченням кількості тестів.

Техніка рекомендує проведення тестів всім класів еквівалентності, хоча б по одному тесту кожного класу. Техніка аналізу класів еквівалентності прагне як скорочувати кількість тестів, а й зберігати прийнятне тестове покриття.

У якості прикладу розглянемо тестування функції «Занесення інформації для оцінювання розміру». Класи еквівалентності даної функції наведені в табл. 3.3

Таблиця 3.3 – Класи еквівалентності функції «Занесення інформації для оцінювання розміру»

Вхідні умови	Класи коректних даних	Класи некоректних даних
Назва програмного застосунку	1 Набір символів	–
Кількість класів	2 Ціле число >0	3 Ціле число ≤ 0
		4 Не ціле число
		5 Не число
Середня кількість методів	6 Дійсне число >0	7 Дійсне число ≤ 0
		8 Не число

Для кожного класу еквівалентності достатньо обрати один тест.

Тести по правильним класам еквівалентності. Для правильних класів еквівалентності, якщо можливо, обираємо один тест на всі правильні класи. Тести по правильним класам еквівалентності знаходяться в таблиці 3.4.

Таблиця 3.4 – Тести по правильним класам

Тест	Клас	Значення	Результат
1	1	Проект 1	Успішно зберегли інформацію для оцінювання розміру
	2	10	
	6	4,5	

Тести по неправильним класам.

У таблиці 3.5 знаходяться тести по неправильним класам. На кожен неправильний клас потрібен свій тест.

Таблиця 3.5 – Тести по неправильним класам

Тест	Клас	Значення	Результат
1	3	-143	Повідомлення про помилку
2	4	1,45	Повідомлення про помилку
3	5	aaa	Повідомлення про помилку
4	7	-1,34	Повідомлення про помилку
5	8	аупп	Повідомлення про помилку

За результатами тестування програма працює вірно.

3.4.3 Випробування ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій

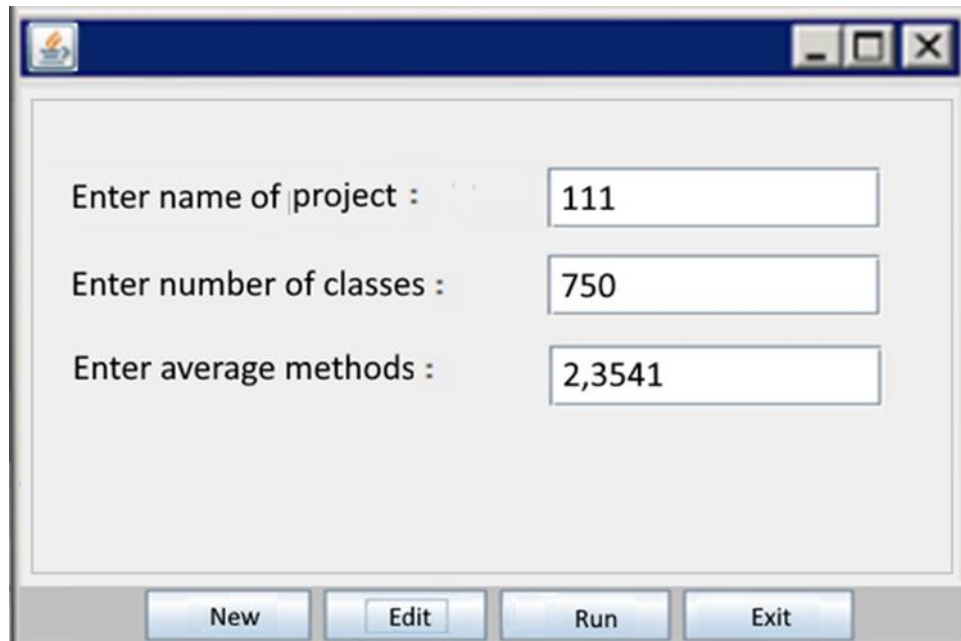
Випробування ПЗ проводилося згідно програми та методики випробувань, яка наведена в додатку В. Метою проведення випробувань являється перевірка відповідності реальної поведінки ПЗ його очікуваній поведінці на скінченному наборі тестів, які вибрані певним чином.

При випробуваннях перевірили всі функції. Як приклад, розглянемо випробування наступних функцій:

1) Функція «Зміна інформації для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій»

- Запустили програмне забезпечення.
- Ввели дані для оцінювання розміру програмних застосунків (назву застосунку, кількість класів, середню кількість методів).
- Введені дані вивелись на екран.
- Повернулися до вводу даних (кнопка «Edit»).
- Ввели змінені дані застосунків (назву застосунку, кількість класів, середню кількість методів).
- Змінені дані вивелися на екран.

Отримали очікуваний результат (рис. 3.8).



Enter name of project :	111
Enter number of classes :	750
Enter average methods :	2,3541

New Edit Run Exit

Рисунок 3.8 – Вікно зі зміненою інформацією для оцінювання розміру програмного застосунку

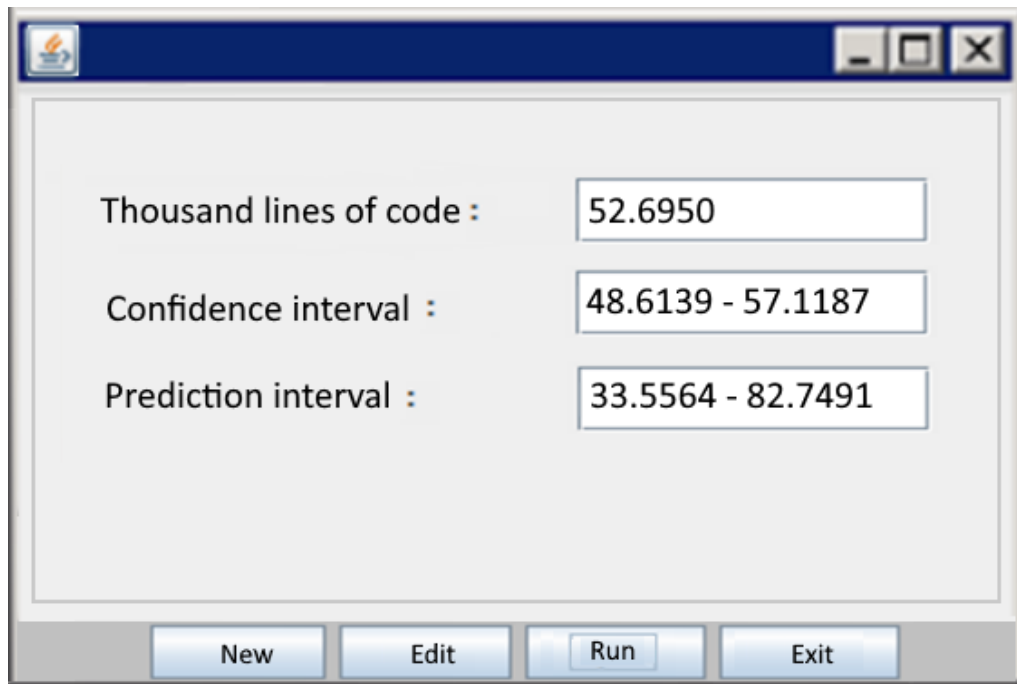
2) Функція «Оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій»

Запустили програмне забезпечення.

– Ввели назву застосунку, кількість класів та середню кількість методів, для яких необхідно виконати оцінювання розміру програмного застосунку.

– Натиснули кнопку «Run».

– Переглянули дані для оцінювання розміру програмного застосунку (рис.3.9).



The image shows a graphical user interface window with a blue title bar and standard Windows window controls (minimize, maximize, close). The main area is light gray and contains three rows of text and input fields:

- Row 1: "Thousand lines of code :" followed by a text box containing "52.6950".
- Row 2: "Confidence interval :" followed by a text box containing "48.6139 - 57.1187".
- Row 3: "Prediction interval :" followed by a text box containing "33.5564 - 82.7491".

At the bottom of the window is a gray bar containing four buttons: "New", "Edit", "Run", and "Exit".

Label	Value
Thousand lines of code :	52.6950
Confidence interval :	48.6139 - 57.1187
Prediction interval :	33.5564 - 82.7491

Рисунок 3.9 – Оцінювання розміру програмного застосунку

В результаті проведених випробувань впевнилися, що ПЗ працює відповідно до вимог, які були висунуті в технічному завданні.

4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВІД РОЗРОБКИ ТА ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНИХ ЗАСТОСУНКІВ НА TYPESCRIPT ДЛЯ КРИПТОВАЛЮТНИХ ОПЕРАЦІЙ

4.1 Вступ

Дана робота присвячена розробці ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій. Його можна використовувати для прогнозування розміру програмних застосунків в комп'ютерних фірмах.

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.

4.2 Розрахунок витрат на створення й експлуатацію програми для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій

Витрати на розробку системи складаються з витрат на:

- заробітну платню розробника;

- амортизацію ЕОМ, на якій виконується розробка;
- експлуатацію цієї ЕОМ;
- засоби розробки;
- матеріали і комплектуючі.

Виходячи з того, що основна заробітна платня розробників програмного забезпечення складає 10000 грн./міс., вартість сучасної ПЕОМ складає 12000 грн. і вартість кіловат-години електроенергії складає 1,68 грн., розрахуємо вартість розробки системи.

Розрахунок заробітної плати:

$$З_{\text{пл}} = ЗП_{\text{роз}} \cdot T_{\text{роз}} \quad (4.1)$$

де $ЗП_{\text{роз}}$ – зарплатня розробника за місяць; $T_{\text{роз}}$ – тривалість розробки (дослідження, створення, налагодження і впровадження).

Для розробки даної системи необхідно 4 місяці (за експертною оцінкою часу на розробку аналогічних систем). Кількість розробників – 1 людина.

З цього випливає, що загальна сума витрат на заробітну платню складе:

$$З_{\text{пл}} = ЗП_{\text{роз}} \cdot T_{\text{роз}} = 10000 \cdot 4 = 40000 \text{ (грн.)}$$

Витрати на амортизацію ЕОМ, на якій виконується розробка, розраховується за формулою:

$$З_{\text{амор}} = Ц_{\text{варт}} \cdot A \cdot T_{\text{роз}} \quad (4.2)$$

де $Ц_{\text{варт}} = 12000$ грн. – балансова вартість ЕОМ; $A = 60\%$ - амортизація за рік; термін служби ЕОМ – 5 років; $T_{\text{роз}} = 0,33$ року – час, необхідний для розробки системи.

$$З_{\text{амор}} = 12000 \cdot 0,6 \cdot 0,33 = 2376 \text{ (грн.)}$$

Витрати на експлуатацію ЕОМ, на якій виконується розробка, полягають в оплаті споживаної нею електричної енергії і розраховуються по формулі:

$$Z_{амор} = P_{ЕОМ} \cdot T_{розр} \cdot N_{рд} \cdot N_{рч} \cdot E_{ел} \quad (4.3)$$

де $P_{ЕОМ} = 0,4$ кВт./год. – потужність ЕОМ; $T_{розр} = 4$ місяці – тривалість розробки; $N_{рд} = 22$ дні – число робочих днів у місяці; $N_{рч} = 8$ годин – число годин у робочому дні; $E_{ел} = 1,68$ грн. – вартість 1 кВт/год електроенергії:

$$Z_{амор} = 0,4 \cdot 4 \cdot 22 \cdot 8 \cdot 1,68 = 102,73 \text{ (грн.)}$$

Витрати на матеріали і комплектуючі вироби враховують витрати на папір для друкувальних пристроїв, на картридж і тонер для принтера, а також на непередбачені витрати. Розрахунок приведений у таблиці 4.1 (ціни взяті відповідно до прайс-листів та договірних цін на даний вид матеріалів).

Таблиця 4.1 – Розрахунок витрат на матеріали і комплектуючі вироби

№ з/п	Найменування виробів	Вартість, грн.
1	Папір для принтера	200,00
2	Картридж та тонер для принтера	250,00
3	Література (доступ до Internet)	140,00
4	Непередбачені витрати	60,00
Разом		650,00

Загальний кошторис витрат на створення системи з урахуванням вищевказаного представлений у таблиці 4.2.

Таблиця 4.2 – Загальний кошторис витрат на створення системи

№ з/п	Найменування витрат	Вартість, грн.
1	Заробітна платня основна	40 000,00
2	Заробітна плата додаткова (20% від п.1)	8 000,00
3	Відрахування на соціальне страхування (38% від пп. 1 та 2)	18 240,00
4	Витрати на амортизацію ЕОМ	2376,00
5	Витрати на експлуатацію ЕОМ	102,73
6	Витрати на матеріали і комплектуючі вироби	650,00
7	Адміністративні витрати (50% від основної заробітної платні)	20 000,00
Разом на створення системи		89368,73

До витрат на впровадження системи можна віднести витрати на придбання технічного забезпечення, вартість програмного забезпечення, вартість навчання кадрів, витрати на монтаж и настроювання мережі.

Оскільки параметри технічних засобів, які вже є, відповідають вимогам, то їх вартість при розрахунку витрат враховувати не будемо.

Виходячи з вимог до програмного забезпечення, а також проаналізувавши цінову політику, можемо прийняти наступне (таблиця 4.3).

Таблиця 4.3 – Перелік програмного забезпечення, необхідного для впровадження системи

№	Найменування ПО	Кількість	Вартість, грн.
1	MS Windows 7 Professional	1	2689,00
2	Microsoft Office	безкоштовно	
Разом		2689,00	

Витрати на програмне забезпечення склали 2689 грн.

Згідно з досвідом створення аналогічних програм, приймаємо, що вартість підготовки кадрів дорівнює 500,00 грн. Витрати на супроводження програмного забезпечення дорівнюватимуть 1% від вартості програми и становлять 268,90 грн.

Інвестиційні витрати на впровадження системи з урахуванням вартості навчання кадрів і витрати на супровід представлені в таблиці 4.4.

Таблиця 4.4 – Інвестиційні витрати на впровадження системи

№	Найменування інвестицій	Вартість, грн.
1	Вартість устаткування	0,00
2	Вартість програмного забезпечення	2689,00
3	Вартість підготовки кадрів	500,00
4	Супроводження системи	268,90
Разом на впровадження системи		3457,90

Разом загальна сума витрат на створення і впровадження системи складає 92 826,63 грн.

4.3 Розрахунок економічної ефективності розробки і впровадження ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій

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

Економічна ефективність впровадження підсистеми оцінювання трудомісткості розбирання списку товарів в залежності від кількості ключів очікується за рахунок вивільнення робочого часу працівників та підвищення продуктивності праці. Крім того, не піддається прямій грошовій оцінці

зменшення кількості помилкових та необережних рішень, підвищення оперативності керування, поліпшення організації роботи та своєчасне отримання необхідної інформації тощо.

Використання даної програми дозволяє вивільнити 0,5 робочого часу. Оскільки з системою працює один робітник, то система умовно вивільнить 0,5 робітника.

Визначимо пряму економічну ефективність, ґрунтуючись на тому, що впровадження системи вивільнить 0,5 робітника.

Зарплата 0,5 робітника у рік складає $\Delta C = 10000 \cdot 12 \cdot 0,5 = 60000$ (грн.). Річний економічний ефект розраховується за формулою 4.4:

$$E_{\text{рік}} = \Delta C - C_{\text{супр}} - E_n \cdot k, \quad (4.4)$$

де $C_{\text{супр}}$ – вартість супроводження системи; E_n – нормативний коефіцієнт економічної ефективності капітальних вкладень у галузь – для обчислювальної техніки приймається $= 0,5$; k – додаткові капітальні вкладення з урахуванням витрат на проектування, створення і функціонування системи.

$$E_{\text{рік}} = 60000 - 268,90 - 0,5 \cdot 92826,63 = 13317,79 \text{ (грн.)}$$

Строк окупності системи розраховується за формулою 4.5:

$$I = \frac{k}{\Delta C - C_{\text{супр}}}. \quad (4.5)$$

Отже

$$I \approx 1,55 \text{ (року)}.$$

Тобто, система окупиться через 1,55 року, що приблизно дорівнює 19 місяцям.

4.4 Висновки за розділом 4

У цьому розділі були здійснені розрахунки витрат на створення й експлуатацію системи оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій, а також розрахунки економічної ефективності та строку окупності даного ПЗ. Виходячи з розрахунків, впровадження такого ПЗ окупить себе за 19 місяців. Отже, можна зробити висновок, що дана розробка програмного забезпечення економічно вигідна та обґрунтована.

5 ОХОРОНА ПРАЦІ

5.1 Загальні визначення та актуальність питання охорони праці

Охорона праці є системою правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних та лікувально-профілактичних заходів та засобів, спрямованих на збереження здоров'я і працездатності людей в процесі праці.

Як відомо, цілком безпечних і нешкідливих виробництв не існує, тому основною задачею охорони праці є звести до мінімуму імовірність поразки чи захворювання працюючого з одночасним забезпеченням комфорту при максимальній продуктивності праці. Основний принцип у галузі охорони праці – це пріоритет життя і здоров'я працівника по відношенню до результатів виробничої діяльності. При цьому роботодавці зобов'язані забезпечувати здорові і безпечні умови праці для своїх працівників, вживати всіх заходів щодо запобігання ризиків на робочих місцях, навчати працівників правилам і нормам охорони праці і виконувати інші вимоги законодавства.

Турбота про охорону праці є справою державної важливості. 14 жовтня 1992 року був прийнятий Закон України "Про охорону праці", у якому закріплені найважливіші положення в області охорони праці, і який поширюється на всіх суб'єктів підприємницької діяльності незалежно від форм власності.

Ст.4 Закону "Про охорону праці" формулює принципи державної політики у сфері охорони праці. Ці принципи в силу вказаного їх визначення, слід кваліфікувати і як принципи інституту охорони праці, які підлягають застосуванню при відсутності відповідних конкретних правових норм, і як такі, що визначають напрямки подальшого розвитку інституту охорони праці, і обов'язки держави, які вона взяла на себе і має додержувати при здійсненні своїй політики у майбутньому. З урахуванням цього принципи, зазначені ст. 4

Закону "Про охорону праці" мають певне регулятивне значення. В статті закріплені такі принципи:

- пріоритет життя і здоров'я працівників відносно результатів виробничої діяльності підприємства, повна відповідальність власника за створення безпечних і нешкідливих умов праці;

- комплексне вирішення завдань охорони праці на засадах національних програм з цих питань, а також з урахуванням інших напрямків економічної і соціальної політики, досягнень в галузі науки, техніки охорони здоров'я;

- соціальний захист працівників, повне відшкодування шкоди особам потерпілим від нещасних випадків на виробництві та професійних захворювань;

- установлення єдиних нормативів з охорони праці для всіх підприємств незалежно від форм власності і видів їх діяльності;

- використання економічних методів управління охороною праці. проведення політики пільгового оподаткування, то сприяє створенню безпечних та нешкідливих умов праці, участь держави в фінансуванні заходів з охорони праці;

- здійснення навчання населення, професійної підготовки та підвищення кваліфікації працівників з питань охорони праці;

- забезпечення координації діяльності державних органів, установ, організацій та об'єднань громадян, що вирішують різні проблеми охорони здоров'я, гігієни та безпеки праці, а також співробітництво.

Державне керування органами охорони праці здійснюють: Кабінет Міністрів України, державний комітет України по нагляду за охороною праці, міністерство й інші централізовані органи, місцеві ради народних депутатів і їхні виконавчі комітети. Державний нагляд за дотриманням законодавства й інших нормативних актів по охороні праці здійснюють: державний комітет України по нагляду за охороною праці, державний комітет України по атомній

і радіаційній безпеці, органи державного пожежного нагляду, органи і служби санітарно епідемічного нагляду. А трудові колективи в особі своїх уповноважених і профспілки здійснюють суспільний контроль за охороною праці.

Вищий нагляд за точним виконанням законів про працю міністерствами і відомствами, підприємствами, установами, громадянами і посадовими особами покладений на Генерального Прокурора України.

Велике значення в створенні здорових і безпечних умов праці має стандартизація. Вона дозволяє вжити діючих заходів по підвищенню технічного рівня й упорядкуванню розробки нормативно-технічної документації по безпеці праці. З 1972 року створюється Система стандартів по безпеці праці (ССБП), що являє собою комплекс великого числа взаємозалежних стандартів, спрямованих на забезпечення безпеки праці. Ця система встановлює загальні вимоги і норми по видах небезпечних і шкідливих виробничих факторів, загальні вимоги безпеки до виробничого устаткування і процесів, вимоги до засобів захисту працюючих, методи оцінки безпеки праці.

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

Однією із характерних особливостей сучасного розвитку суспільства є зростання сфер діяльності людини, в яких використовуються інформаційні технології. Широке розповсюдження отримали персональні комп'ютери. Однак їх використання загостило проблеми збереження власного та суспільного здоров'я, вимагає удосконалення існуючих та розробки нових підходів до організації робочих місць, проведення профілактичних заходів для запобігання розвитку негативних наслідків впливу ПК на здоров'я

користувачів. Тому питання охорони праці при роботі з ПК є досить актуальним.

5.2 Аналіз небезпечних і шкідливих факторів при роботі з ПК

У сучасному житті комп'ютер широко застосовується в житті людини: і удома, і в офісі, і в магазині, і у виробництві, і навіть в побутовій техніці – іншими словами комп'ютери міцно увійшли до повсякденного життя людей і їх використання постійно збільшується.

В офісах комп'ютери в основному використовуються як допоміжні засоби обробки інформації, і таке введення комп'ютерних технологій принципово змінило характер праці офісних працівників і вимоги до організації і охорони праці.

Недотримання вимог безпеки приводить до того, що через деякий час роботи за комп'ютером співробітник починає відчувати певний дискомфорт: у нього виникають головні болі і різь в очах, з'являються втома і дратівливість. У деяких людей порушується сон, погіршується зір, починають хворіти руки, шия, поперек і тому подібне.

З точки зору впливу небезпечних і шкідливих факторів при роботі з ПК можна виділити наступні:

- вплив електромагнітного та електростатичного випромінювань;
- вплив недостатньої освітленості робочого місця;
- вплив електричного струму;
- вплив шуму;
- вплив несприятливих умов мікроклімату;
- вплив нераціонального розташування устаткування і неправильної організації робочого місця.

Тривале перебування людини в зоні комбінованого впливу різних несприятливих факторів може призвести до професійного захворювання. Розглянемо більш детально вплив означених факторів на користувача ПК.

Вплив електромагнітного та електростатичного випромінювань

При роботі ПК виникають два типи випромінювань: електростатичне та електромагнітне. Навколо електростатично-зарядженого монітора підвищується концентрація пилу. Електромагнітне випромінювання створюється магнітними котушками. Результати медичних досліджень показують, що електризований пил викликає запалення шкіри, а електромагнітне випромінювання приводить до порушення біологічних процесів (порушення сну, зниження кров'яного тиску, уповільнення серцевої діяльності, зміни складу крові).

Вплив недостатньої освітленості робочого місця

Напружена зорова робота внаслідок нераціонального освітлення може бути причиною функціональних порушень у зоровому аналізаторі і привести до розладу зору, а у важких випадках – і до повної втрати. Втома органів зору залежить від ступеня напруженості процесів, що супроводжують зорове сприйняття. При роботі оператора ПК одним з основних навантажень на організм є навантаження на зір, тому дуже важливо приділити особливу увагу освітленості робочого місця.

У залежності від джерела світла виробниче висвітлення може бути трьох видів: природним, штучним і сполученим. При роботі з ПК вдень у ясну погоду використовується природне освітлення, яке здійснюється через світлові віконні прорізи. При нестачі природного світла, а також при роботі у вечірній час використовується штучне освітлення, джерелами якого є газорозрядні лампи. При такому освітленні величезне значення має правильне розташування джерел світла, їхній тип і кількість.

Правильна освітленість робочого місця сприяє високій працездатності користувача ПК і зводить до мінімуму шкідливий вплив на органи зору.

Освітленість робочого місця, що рекомендується для користувача ПК складає 400–750 лк.

Вплив електричного струму

Робота комп'ютерів і периферійних пристроїв можлива завдяки електричному струмові, до джерел якого вони підключаються, тому існує небезпека ураження користувача струмом. Щоб уникнути нещасних випадків варто строго дотримуватися правил електробезпеки.

Небезпека електричного струму на відміну від інших небезпек збільшується за рахунок того, що людина не в змозі без спеціальних приладів виявити напругу дистанційно, а також швидкоплинності поразки – небезпека виявляється, коли людина вже уражена. Ураження струмом приводить до різних порушень в організмі, викликаючи як місцеву поразку тканин і органів, так і загальне ураження організму. Людина відчуває дратівну дію перемінного струму промислової частоти силою 0,6–1,5 мА і постійного струму 5–7 мА. Ці струми не представляють серйозної небезпеки для організму людини, тому що при такій силі струму можливе самостійне звільнення людини від контакту зі струмоведучими частинами. Для перемінного струму промислової частоти сила невідпускаючого струму знаходиться в межах 6–20 мА. Постійний струм не викликає невідпускаючого ефекту, але приводить до сильних болючих відчуттів, сила такого струму 15–80 мА і більше.

Вплив шуму

Шумом прийнято називати всякий небажаний для людини звук, що заважає сприйняттю корисних сигналів. Шум являє собою безладне сполучення звуків різної інтенсивності і частоти. Він викликає труднощі з розпізнанням колірних сигналів, знижує швидкість сприйняття кольору, гостроту зору, зорову адаптацію, порушує сприйняття візуальної інформації, зменшує на 5–12% продуктивність праці. Тривалий вплив шуму з рівнем звукового тиску 90 дБ знижує продуктивність праці на 30–60%.

Робота ПК може супроводжуватися шумами, викликаними деякими рухливими пристроями усередині комп'ютера, а також підключеними до

нього. До таких пристроїв можна віднести вентилятори, пристрої читання дисків, принтери. Виникаючий шум, у залежності від інтенсивності, може привести до підвищення нервової напруги і зниження працездатності.

Вплив несприятливих умов мікроклімату

Мікрокліматичні параметри (температура, відносна вологість і швидкість руху повітря) впливають на самопочуття і здоров'я людини, а також на надійність роботи засобів обчислювальної техніки.

При підвищеній температурі і вологості в приміщенні відбувається перегрівання тіла, що погіршує самопочуття і працездатність. Варто враховувати, що ПК, які знаходяться в приміщенні, сприяють збільшенню температури.

Для комфортної роботи приймається температура $23 (\pm 2)^{\circ}\text{C}$ й $18 (\pm 2)^{\circ}\text{C}$ в теплий і холодний час року відповідно, при відносній вологості $55 (\pm 5)\%$.

Вплив нераціонального розташування устаткування і неправильної організації робочого місця.

Робоче місце – це частина простору, у якій користувач ПК здійснює трудову діяльність і проводить велику частину робочого часу. Важливу роль грає планування робочого місця, яке повинне задовольняти вимогам зручності виконання робіт і економії енергії і часу користувача, зручності обслуговувань ЕОМ, дотримання правил охорони праці.

При плануванні робочого місця необхідно враховувати зони досяжності при розташуванні моніторів, клавіатури, маніпулятора «миша». Ці зони встановлюються на підставі антропологічних даних людського тіла. Під час роботи користувач ПК не повинен піддаватися впливові сторони подразників, якими можуть бути неадекватне фарбування пристроїв ЕОМ або приміщення.

Відповідно до законодавчих актів площа робочого місця користувача ПК повинна складати не менше $4,5\text{м}^2$. У приміщеннях повинні проводитися щоденне вологе прибирання і систематичне провітрювання після кожної години роботи. Устаткування, що виробляє багато шуму (друкуючі пристрої,

сканери, сервери і тому подібні), рівні шуму якого перевищують нормативні, повинне розміщуватися поза робочими місцями співробітників.

Робочі столи слід розміщувати так, щоб монітори були орієнтовані бічною стороною до світлових отворів, щоб природне світло падало переважно зліва.

При розміщенні робочих місць відстань між робочими столами має бути не менше 2,0 м-коду, а відстань між бічними поверхнями відеомоніторів — не менше 1,2 м. Робочі місця співробітників, що виконують творчу роботу і що вимагає значної розумової напруги або високої концентрації уваги, рекомендується ізолювати один від одного перегородками заввишки від 1,5 м.

Конструкція робочого столу повинна забезпечувати оптимальне розміщення на робочій поверхні використовуваного устаткування. Висота робочої поверхні столу повинна складати 725 мм, робоча поверхня столу повинна мати ширину 800..1400 мм і глибину 800..1000 мм. Робочий стіл повинен мати простір для ніг заввишки не менше 600 мм, шириною — не менше 500 мм, завглибшки на рівні колін — не менше 450 мм і на рівні витягнутих ніг — не менше 650 мм.

Конструкція робочого стільця або крісла повинна забезпечувати підтримку раціональної робочої пози працівника і дозволяти змінювати позу з метою зниження статичної напруги м'язів шийно-плечової області і спини. Робочий стілець або крісло мають бути підйимально-поворотними, регульованими по висоті і кутам нахилу сидіння і спинки, а також відстані спинки від переднього краю сидіння, при цьому регулювання кожного параметра має бути незалежним, легко здійснюваним і мати надійну фіксацію.

Клавіатуру слід розташовувати на поверхні столу на відстані 100..300 мм від краю, зверненого до користувача, або на спеціальній поверхні, відокремленій від основної.

Екран відеомонітора повинен знаходитися від очей користувача на відстані 600..700 мм, але не ближче 500.

5.3 Розрахунок системи загального освітлення приміщення з ПК

Раціональне освітлення приміщень – один з найбільш важливих факторів, від яких залежить ефективність трудової діяльності людини. Гарне освітлення необхідне для виконання більшості задач.

До систем освітлення висувають наступні вимоги:

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

В приміщенні з ПК необхідно організувати змішане освітлення (комбінування природного і штучного освітлення). Природне освітлення створюється бічним освітленням через вікно. Штучне освітлення використовується при недостатньому природному освітленні. У даному приміщенні використовується загальне штучне освітлення. Розрахунок виконаємо за методом світлового потоку з урахування потоку, віддзеркаленого від стін та стелі.

Необхідно зробити розрахунок загального освітлення виробничої приміщення. Довжина $A = 9$ м, ширина $B = 6$ м, висота $H = 3$ м. Висота робочої поверхні $h_p = 1$ м. Для освітлення використовується світильні УПМ-15. Мінімальна освітленість лампи наколювання по нормах $E_{\min} = 100$ лк. Коефіцієнт відображення стелі $S_n = 50\%$, стін $S_c = 30\%$, робочої поверхні $S_p = 10\%$. Напруга мережі 220 В.

Відстань від стелі до робочої поверхні:

$$H_0 = H - h_p = 3 - 1 = 2 \text{ м.}$$

Відстань від стелі до світильника:

$$h_c = 0,2 \cdot H_0 = 0,2 \cdot 2 = 0,4 \text{ м.}$$

Висота підвісу світильника над освітленою поверхнею:

$$h = H_0 - h_c = 2 - 0,4 = 1,6 \text{ м.}$$

Висота підвісу світильника над підлогою:

$$H_n = h + h_p = 1,6 + 1 = 2,6 \text{ м.}$$

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

Відстань між центрами світильників:

$$L = 1,5 \cdot h = 1,5 \cdot 1,6 = 2,4 \text{ м.}$$

Необхідна кількість світильників:

$$N = S / L^2 = (9 \cdot 6) / (2,4)^2 \approx 9,375.$$

Приймаємо кількість світильників: 10 – 2 ряди по 5 світильників.

Індекс приміщення:

$i = (A \cdot B) / (h \cdot (A + B)) = (9 \cdot 6) / (1,6 \cdot (9 + 6)) = 2,25$ (найближче значення в таблиці 2,5).

По таблиці коефіцієнтів використання світлового потоку при $i=2,25$, $S_n=50\%$, $S_c=30\%$, $S_p=10\%$ для світильника типу «УПМ-15» коефіцієнт використання світлового потоку $\eta=0,53$.

Світловий потік однієї лампи:

$$\Phi = (E_{\min} \cdot S \cdot K_3 \cdot Z) / (N \cdot \eta) = (100 \cdot 54 \cdot 1,5 \cdot 1,1) / (10 \cdot 0,53) = 1681 \text{ лм,}$$

де K_3 – коефіцієнт запасу, який враховує запилення світильників і знос джерел світла в процесі експлуатації; для приміщень, освітлюваних люмінесцентними лампами $K_3=1,5$ за умови чищення світильників не рідше двох разів на рік;

Z – коефіцієнт нерівномірності освітлення ($Z=1,1$);

За знайденим світовим потоком із таблиці «Параметри ламп накаливання загального призначення з розрахунковими напругами 130 і 220 В» обираємо лампу Г215-225-150 накаливання потужністю 150 Вт, яка має світовий потік 2090 лк, що найбільше близький до розрахункового.

Фактична освітленість:

$$E_{\phi} = E_{\min} \cdot \Phi_{\text{л}} / \Phi_{\text{р}} = 100 \cdot 2090 / 1681 = 124,3 \text{ лк.}$$

Загальна потужність:

$$P_{\text{заг}} = P_{\text{л}} \cdot N = 150 \cdot 10 = 1500 \text{ Вт} = 1,5 \text{ кВт.}$$

Таким чином, для оптимальної роботи в приміщенні необхідно встановити 2 ряди по 55 світильники типу Г215-225-150. Загальна потужність усіх світильників складе 1,5 кВт.

5.4 Розробка заходів щодо зменшення впливу шкідливих та небезпечних факторів при роботі з персональним комп'ютером

Заходи щодо зменшення впливу

Заходи щодо усунення небезпеки поразки електричним струмом зводяться до правильного розміщення устаткування та електричних кабелів. Інші заходи щодо забезпечення електробезпеки, збігаються з загальними заходами пожежо- та електробезпеки.

В якості профілактичних заходів для забезпечення пожежної безпеки слід використовувати скриту електромережу, надійні розетки з пожежобезпечних матеріалів, силові мережі живлення устаткування виконувати кабелями, розрахованими на підключення в 3-5 разів більшого навантаження, включати й виключати живлення обладнання за допомогою штатних вимикачів. Треба регулярно робити очистку внутрішніх частин комп'ютерів, іншого устаткування від пилу, розташовувати комп'ютери на окремих не спалюваних столах. Для запобігання іскріння необхідно рідше встромляти і виймати штепсельні вилки з розеток.

Екран дисплея повинен бути розташованим перпендикулярно до напрямку погляду. Якщо він розташований під кутом, то стає причиною сутулості. Відстань від дисплея до очей повинна трохи перевищувати звичну відстань між книгою та очима. Перед екраном монітора, особливо старих типів, повинен бути спеціальний захисний екран. При його відсутності треба сидіти на відстані витягнутої руки від монітора. Ще одним моментом, який

стосується зору, є необхідність створення неоднорідного поля зору. Для цього можна розвісити на поверхнях (стінах) плакати та картини, виконані у спокійних тонах. Наприклад, пейзажі.

Важливою є форма спинки крісла, яка повинна повторювати форму спини. Висота крісла повинна бути такою, щоб користувач не почував тиску на куприк або стегна. Крісло бажано обладнати бильцями. Його потрібно встановити так, щоб не треба було тягтися до клавіатури. Періодично користувачу необхідно рухатися, вчасно змінювати положення тіла і робити перерви у роботі.

При напруженій роботі за комп'ютером щогодини необхідно робити перерву на 15 хвилин через кожну годину і треба займатися іншою справою. Декілька разів на годину бажано виконувати серію легких вправ для розслаблення.

Техніка безпеки при роботі на комп'ютері

Загальні положення:

1) При виконанні робіт на персональних комп'ютерах (ПК) 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. В подальшому вони проходять повторні інструктажі з охорони праці на робочому місці один раз на півріччя.

3) При роботі на комп'ютерах, до складу яких входять відеодисплейні термінали (ВДТ), на працівників можуть діяти наступні шкідливі та погрожуючі життю і здоров'ю виробничі фактори:

а) фізичні:

- підвищений рівень шуму (від вентилятора мікропроцесора та плат);
- високий рівень напруги в електричній мережі, торкання до якої може викликати нещасний випадок;
- підвищений рівень статичної електрики;

- підвищений рівень електромагнітного випромінювання;
- підвищений рівень напруженості електричного поля;
- пряма і відбита від екрана блескність;
- несприятливе розподілення яскравості об'єктів в полі зору.

б) психофізіологічні:

- фізичні перевантаження статичної та динамічної дії;
- нервово-психічні перевантаження (розумове перенапруження, перенапруження зорових аналізаторів, монотонність праці, емоційні перевантаження).

4) Основним обладнанням робочого місця користувача ПК є: ВДТ, 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 з ВДТ повинні бути розташовані на відстані не менше 1,5 м від стіни з вікнами, від інших стін – на відстані 1 м, 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) Робочі місця, обладнані ВДТ, слід розташовувати таким чином, щоб запобігти попаданню 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 антивідблисківі сітки, 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 (в режимі введення даних, редагування тексту і читання з екрану ВДТ) найбільш фізіологічним є зображення чорних знаків на світлому (білому) фоні.

7) Розташовувати ВДТ на робочому місці слід так, щоб поверхня екрану знаходилась в 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 ВДТ.

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 – в межах 15-20 град.

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

Вимоги безпеки перед початком роботи:

1) Перевірити надійність встановлення апаратури на робочому столі. ВДТ не повинен стояти на краю столу. Повернути ВДТ 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), зручний для користувача нахил його спинки.

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 вимкненому електричному живленні.

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

8) Відрегулювати яскравість світіння екрану ВДТ, фокусировку, контрастність. Не слід робити яскравість екрану занадто великою, тому що це втомлює очі (і зменшує термін роботи електронно-променевої трубки).

Рекомендується:

- яскравість світіння екрану – не менше 100 кд/ кв.м (кандел на кв.м);
- відношення яскравості екрану ВДТ до яскравості оточуючих його поверхонь в робочій зоні – не більше 3:1;
- мінімальний розмір точки світіння – не більше 0,4 мм для монохромного ВДТ 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 ВДТ і системний блок. Вимкнути стабілізатор (або блок безперервного живлення

при його наявності). Штепсельні вилки витягнути з розеток. Накрити клавіатуру кришкою для попередження попадання 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, ВДТ, 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 пунктом 5.1. 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 по евакуації людей з приміщення і розпочати гасіння пожежі первинними засобами пожежогасіння.

6 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

6.1 Вступ

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

Охорона навколишнього природного середовища складається з:

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

Охороні підлягають наступні об'єкти:

- природні екологічні системи, озоновий шар атмосфери;
- земля, її надра, поверхневі й підземні води, атмосферне повітря, ліси й інша рослинність, тваринний мир, мікроорганізми, генетичний фонд, природні ландшафти.

Особливо охороняються державою природні заповідники, природні заказники, національні природні парки, пам'ятники природи, рідкі види, що перебувають під загрозою зникнення, рослин, тварин і місця їхнього перебування.

Основними принципами охорони навколишнього середовища є: пріоритет забезпечення сприятливих екологічних умов для життя, праці й відпочинку населення; науково обґрунтоване сполучення екологічних і економічних інтересів суспільства; облік законів природи й можливостей самовідновлення й самоочищення її ресурсів.

6.2 Види забруднення навколишнього природного середовища

Різноманітне втручання людини в природні процеси в біосфері можна згрупувати по наступних видах забруднень, розуміючи під ними будь-які небажані для екосистем антропогенні зміни:

- інгредієнті (інгредієнт – складова частина складного з'єднання або суміші) забруднення як сукупність речовин, кількісно або якісно далеких природним біогеоценозам;
- параметричне забруднення (параметр навколишнього середовища - одне з її властивостей, наприклад рівень шуму, освітленості, радіації й т.д.), пов'язане зі зміною якісних параметрів навколишнього середовища;
- біоценотичне забруднення, що полягає у впливі на склад і структуру популяції живих організмів;
- стаціонально-деструкційне забруднення (стація – місце перебування популяції, деструкція – руйнування), що представляє собою зміну ландшафтів і екологічних систем у процесі природокористування.

Основні зусилля на сьогоднішній день спрямовані на зниження рівня матеріального й енергетичного забруднення навколишнього середовища.

6.3 Природоохоронна діяльність підприємств

Природоохоронної є будь-яка діяльність, спрямована на збереження якості навколишнього середовища на рівні, що забезпечує стійкість біосфери. До неї ставиться як великомасштабна діяльність, здійснювана на загальнодержавному рівні, так і діяльність окремих підприємств по очищенню від шкідливих речовин стічних вод і газів, що відходять, зниженню норм використання природних ресурсів і т.д.

Існує два основних напрямки природоохоронної діяльності. Перше - очищення шкідливих викидів. Цей шлях «у чистому виді» малоефективний, тому що з його допомогою далеко не завжди вдається повністю припинити

надходження шкідливих речовин у біосферу. До того ж скорочення рівня забруднення одного компонента навколишнього середовища веде до посилення забруднення іншого.

Другий напрямок – усунення самих причин забруднення, що вимагає розробки маловідходних, а в перспективі й безвідходних технологій виробництва, які дозволяти б комплексно використовувати вихідну сировину й утилізувати максимум шкідливих для біосфери речовин.

Однак далеко не для всіх галузей знайдені прийнятні техніко-економічні рішення по різкому скороченню кількості відходів, що утворюються, і їхньої утилізації, тому в цей час доводиться працювати по обох зазначених напрямках.

6.4 Розробка заходів щодо зменшення забруднення

Одним із заходів щодо зменшення забруднення навколишнього середовища є дотримання стандартів на монітори та ЕОМ. Існують дві основні групи стандартів і рекомендацій – з безпеки й ергономіки.

До першої групи ставляться стандарти UL, CSA, DHHS, CE, скандинавські SEMRO, DEMKO, NEMKO, а також FCC Class B. Із другої групи найбільш відомі MPR-II, TCO'92, TCO'95, ISO 9241-3, EPA Energy Star, TUV Ergonomie. От деякі з них:

1) FCC Class B – цей стандарт розроблений канадською федеральною комісією з комунікацій для забезпечення прийнятного захисту навколишнього середовища від впливу радіоперешкод у замкнутому просторі. Устаткування, що відповідає вимогам FCC Class B, не повинне заважати роботі теле- і радіо апаратури.

2) MPR-II – цей стандарт був випущений в 1990р. Шведським національним департаментом і затверджений ЄЕС. MPR-II накладає обмеження на випромінювання від комп'ютерних моніторів і промислової техніки, використовуваної в офісі.

3) TCO'92 (TCO'95) – рекомендація, розроблена Шведською конференцією профспілок і Національною радою індустріального й технічного розвитку Швеції (NUTEK), регламентує взаємодія з навколишнім середовищем. Вона вимагає зменшення електричних і магнітних полів до технічно можливого рівня з метою захисту користувача. Для того, щоб одержати сертифікат TCO'92, монітор повинен відповідати стандартам низького випромінювання (Low Radiation), тобто мати низький рівень електромагнітного поля, забезпечувати автоматичне зниження енергоспоживання при довгому не використанні, відповідати європейським стандартам пожежної й електричної безпеки.

4) TUV Ergonomie – німецький стандарт ергономіки. Монітори відповідають цьому стандарту, пройшли випробування згідно EN 60950 (електрична безпека) і ZN 1/618 (ергономічна облаштованість робочих місць, оснащених дисплеями), а також відповідають шведському стандарту MPR-II.

5) EPA Energy Star VESA DPMS – відповідно до цього стандарту монітор повинен підтримувати три енергозберігаючих режими – очікування (stand-by), припинення (suspend) і «очікування» (off). У режимі очікування зображення на екрані пропадає, але внутрішні компоненти монітора функціонують у нормальному режимі, а енергоспоживання знижується до 80% від робочого стану. У режимі припинення, як правило, відключаються високовольтні вузли, а споживання електроенергії падає до 30 Вт і менш. І нарешті в режимі так званого «очікування» монітор споживає не більше 8 Вт, а функціонує в нього тільки мікропроцесор. При натисканні будь-якої клавіші клавіатури або русі миші монітор переходить у нормальний режим роботи.

Величезною екологічною проблемою є накопичення електронного сміття. Екологічна політика Європейського Співтовариства пов'язана з прийняттям та реалізацією шести Програм дій з навколишнього середовища. Значна увага в ЄС приділяється правової охорони навколишнього середовища від відходів різного походження.

Утилізація офісної техніки, в тому числі і комп'ютерів, є обов'язковою процедурою для всіх працюючих підприємств. Неправильне проведення цієї

процедури (викид на смітник) здатне викликати маленьку «техногенну катастрофу». Електронні компоненти оргтехніки містять ртуть, свинець, миш'як, які під впливом зовнішніх факторів і органічних складових (а їх теж використовують в комп'ютерах) перетворюються на сильнодіючі отрути, отруйні навколишнє середовище.

Крім цього в моніторах і системних блоках комп'ютерів є дорогоцінні метали. Тому утилізацію комп'ютерів можуть проводити тільки спеціальні організації. Завдяки комплексному характеру цієї процедури цінні матеріали (чорні, кольорові і дорогоцінні метали) надходять назад у виробничий цикл, а інші переробляються. Якщо у підприємства немає дозволу на самостійну переробку, то необхідна допомога професійної організації.

За законом для списання та утилізації обладнання і ПК юридична особа повинна звернутися до одного з спеціалізованих підприємств, які займаються виробництвом, ремонтом і обслуговуванням персональної техніки, з проханням про проведення експертизи, і отримання висновку про те, що дана модель морально застаріла, знята з виробництва і ремонту не підтягає. Тільки після цього можна укласти договір з відповідною організацією про утилізацію відходів.

Ліцензію на утилізацію відходів, що містять дорогоцінні метали, видає Міністерство фінансів, кольорові і чорні – Міністерство промислової політики, а полімери та отруйні матеріали – Міністерство екології та природних ресурсів. Процес дуже заплутаний і складний, але саме такий офіційний порядок утилізації оргтехніки у компаній, що бажають уникнути конфліктів з чинним законодавством. Утилізація обладнання потребує грошей, однак такі витрати неминучі.

У Німеччині за утилізацію старого ПК (залежно від моделі) необхідно заплатити від 15 до 30 євро. У Японії вартість переробки одного ПК складає \$40. Плата за переробку одного комп'ютера в Україну – близько 100 грн. Згідно довідковими даними, в персональному комп'ютері, містяться дорогоцінні метали, (Au – 0,053-0,072 г, Ag – 0,8-1,1 г), кольорові і чорні метали (Al – 0,1-0,4 кг, Cu – 0,1-0,2 кг, Br – 3-4 кг), а також полімери і скло.

ВИСНОВКИ

Кваліфікаційну роботу присвячено побудові математичної моделі для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій та розробці програми для її реалізації.

Під час виконання магістерської роботи були поставленні задачі, які було виконано у повному обсязі:

- Аналіз існуючих методів та моделей оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій.

- Збирання емпіричних даних для побудови математичної моделі оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій.

- Формулювання гіпотези про форму зв'язку.

- Нормалізацію вихідних даних, так як емпіричні дані для побудови математичної моделі оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій мають нормальний розподіл лише в окремих випадках.

- Перевірку нормалізованих даних на викиди.

- Побудову лінійної регресійної моделі, довірчих інтервалів та інтервалів прогнозування для нормалізованих даних.

- Отримання нелінійної регресійної моделі оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій, її довірчих інтервалів та інтервалів прогнозування через зворотне перетворення.

- Оцінювання якості регресійного аналізу.

- Інтерпретацію одержаних результатів.

- Прогнозування невідомих значень розміру програмних застосунків на TypeScript для криптовалютних операцій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Методы оценки трудозатрат, длительности и стоимости выполнения программного проекта [Електронний ресурс] // Режим доступу: <https://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/> (дата звернення: 20.10.2022).
2. Tan H. B. K., Zhao Y., Zhang H.. Estimating LOC for information systems from their conceptual data models. *Proceedings of the 28th International Conference on Software Engineering (ICSE '06)*. Shanghai, China, May 20-28, 2006. P. 321-330.
3. Макарова Л. М., Приходько Н. В., Кудін О. О. Побудова нелінійної регресійної моделі для оцінювання розміру веб-додатків, реалізованих мовою Java. *Вісник Херсонського національного технічного університету*. Херсон, 2019. № 2(69). С. 145-153.
4. Макарова Л. М., Латанська Л. О., Нікітін О. В., Нікітіна О.Ю. Математичні моделі для оцінювання розміру програмного забезпечення для крос-платформної розробки мобільних застосунків із використанням платформи Xamarin. *Вчені записки ТНУ імені В.І.Вернадського. Серія: Технічні науки*. Київ, 2022. Том 33(72). № 1. С. 150-156.
5. Приходько Н. В., Приходько С. Б. Нелінійна регресійна модель для оцінювання розміру програмного забезпечення інформаційних систем на базі VB. *Інформаційні технології та компютерна інженерія*. 2018. № 3. С. 37-42.
6. Приходько С.Б., Приходько Н.В. Смикодуб Т.Г. Чотирьохфакторна нелінійна регресійна модель для оцінювання розміру Java-застосунків з відкритим кодом. *Вчені записки ТНУ імені В.І. Вернадського. Серія: технічні науки*. Київ, 2020. Том 31(70). Ч. 1. № 2. С. 157-162.
7. Приходько С. Б., Ворона М. В. Оцінювання розміру PHP-застосунків з відкритим кодом за нелінійними регресійними моделями з різними факторами *Зб. наук. пр. НУК*. Миколаїв, 2021. № 1 (484). С. 92-98.

8. Латанська Л.О., Павлюк К.А. Регресійні моделі в задачах оцінювання розміру програмних проектів на ранніх стадіях розробки. *Сучасні інформаційні системи та технології*: Матеріали V Всеукраїнської науково-практичної інтернет-конференції студентів, аспірантів та молодих вчених за тематкою: Сучасні комп'ютерні системи та мережі в управлінні (Хмельницький, 30 листопада 2022 року) / Під редакцією А.А. Григорової. Херсон: Видавництво ФОП Вишемирський В.С., 2022. С. 140-141.
9. Титов А.И. Выбор метрики размера проекта в модели оценки трудоемкости разработки программ. *Интеллектуальные технологии на транспорте*. 2016. Вып. 5. С. 31-38.
10. Звездин С.В. Формирование метрик кода программной системы. *Информатка, телекоммуникации и управление*. 2008. № 6. С. 48-52.
11. Новичков А., Шамрай А., Черников А. Метрики кода и их практическая реализация в Subversion и ClearCase. [Електронний ресурс] // Режим доступу: <http://gdznet.com/userhelps/metriki-koda-i-ix-prakticheskaya-realizaciya-v-subversion-i-clearcase-chast-1-metriki.html> (дата звернення: 20.11.2022).
12. Boehm B.W. Software engineering economics. Prentice-Hall, 1981. 320 p.
13. Boehm, B.W. The COCOMO 2.0 Software Cost Estimation Model. American Programmer, 2000. 586 p.
14. Johnson Kim. Software cost estimation. Metrics and models. University of Calgary, 2001. 115 p.
15. Chatterjee S., Price B. Regression Analysis by Example. New York: John Wiley & Son, 1977. 228 p.
16. Буреева Н. Н. Многомерный статистический анализ с использованием ППП "STATISTICA". Учебно-методический материал по программе повышения квалификации «Применение программных средств в научных исследованиях и преподавании математики и механики». Нижний Новгород, 2007. 112 с.

17. Приходько С. Б., Макарова Л. М., Пугаченко К. С. Методичні вказівки та завдання до виконання лабораторних робіт з дисципліни "Обробка експериментальних даних на комп'ютері". Миколаїв: НУК, 2018. 76 с.
18. Кобзарь А.И., Катышев П.К., Пересецкий А.А. Прикладная математическая статистика. Для инженеров и научных работников. М.: ФИЗМАТЛИТ, 2006. 816 с.
19. Магнус Я.Р. Эконометрика. Начальный курс: Учеб. 6-е изд., перераб. и доп. М.: Дело, 2004. 576 с.
20. Вендров А.М. Проектирование программного обеспечения экономических информационных систем. М.: Финансы и статистика, 2006. 544 с.
21. Карпова Т.С. Базы данных: модели, разработка, реализация. СПб.: Питер, 2001. 304с.

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ПЗ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНИХ ЗАСТОСУНКІВ НА TYPESCRIPT ДЛЯ КРИПТОВАЛЮТНИХ ОПЕРАЦІЙ

Вступ

Назва програми – ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій.

Програмне забезпечення являється програмою, яка буде надавати змогу користувачу провести оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій.

1. Підстави для розробки

Документом, на підставі якого ведеться розробка програмного продукту, є завдання на кваліфікаційну роботу «Нелінійна регресійна модель для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій 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 для криптовалютних операцій і побудови інтервалу довіри та інтервалу прогнозування 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:

- занесення даних;
- корегування даних;
- оцінювання розміру програмного застосунку;
- розрахунок верхньої та нижньої меж довірчого інтервалу;
- розрахунок верхньої та нижньої меж інтервалу прогнозування.

3.1.2 Вимоги до організації вхідних та вихідних даних

Вихідні дані зберігаються у таблицях бази даних. Внесення вхідних даних виконується шляхом їх введення у спеціальних формах.

Результуючі дані (прогнозне значення, довірчий інтервал та інтервал прогнозування) повинні виводитися в відповідних місцях на створеній програмою формі.

3.2 Вимоги до надійності

3.2.1 Вимоги до забезпечення надійності функціонування програми

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

3.2.2 Відмова виконання програмного продукту через некоректні дії користувача

Відмова програми можлива внаслідок некоректних дій користувача при взаємодії з операційною системою. Щоб уникнути виникнення відмов програми за вказаною вище причиною 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), що використовуються 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 та зберігання програми відсутні.

4 Вимоги до програмної документації

Склад програмної документації:

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

5 Техніко-економічні показники

Економічна ефективність впровадження програмного забезпечення розрахована у розділі 5. Згідно з отриманими результатами впровадження

даного ПЗ є доцільним, а термін його окупності становить приблизно 19 місяців.

6 Стадії та етапи розробки програмного забезпечення

Стадії та етапи розробки ПЗ для оцінювання розміру програмних застосунків на TYPEScript для криптовалютних операцій зазначені в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки програмного забезпечення

Стадії розробки	Етапи робіт	Контрольні точки
Технічне завдання	Обумовлення необхідності розробки	З 07.09.2022 до 10.09.2022
	Розробка та затвердження технічного завдання	З 11.09.2022 до 15.09.2022
Ескізний проект	Розробка ескізного проекту	З 16.09.2022 до 10.10.2022
	Затвердження ескізного проекту	З 11.10.2022 до 14.10.2022
Технічний проект	Розробка технічного проекту	З 15.10.2022 до 05.11.2022
	Затвердження технічного проекту	З 06.11.2022 до 10.11.2022
Робочий проект	Розробка ПЗ	З 11.11.2022 до 21.11.2022
	Розробка програмної документації	З 22.11.2022 до 01.12.2022
	Випробування програми	З 02.12.2022 до 05.12.2022

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 виконавач сумісно з замовником підписують акт приймання-здачі програми в експлуатацію.

ДОДАТОК Б

ОПИС ПЗ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНИХ ЗАСТОСУНКІВ НА TYPESCRIPT ДЛЯ КРИПТОВАЛЮТНИХ ОПЕРАЦІЙ

1 Загальні відомості

1.1 Позначення і найменування програми

Найменування: ПЗ для оцінювання розміру програмних застосунків на TypeScript для криптовалютних операцій. Позначення: «Size_Project»,

1.2 Мови програмування

ПЗ написане на мові програмування Java. Модулі, реалізовані на інших мовах програмування, не використовуються.

2 Функціональне призначення

2.1 Класи вирішуваних завдань і призначення програми

ПЗ «Size_Project» повинно забезпечувати виконання наступних функцій:

- занесення даних;
- корегування даних;
- прогнозування розміру;
- розрахунок верхньої та нижньої меж довірчого інтервалу;
- розрахунок верхньої та нижньої меж інтервалу прогнозування.

2.2 Функціональні обмеження

Файли повинні бути формату EXE.

3 Використовувані технічні засоби

- ПК з процесором не нижче : AMD FX-6300;
- Обсяг оперативної пам'яті – не менше 2 Гб;

– Необхідний обсяг на жорсткому диску – не менше 600 Мб.

4 Виклик і завантаження

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

- 1) за допомогою файлу запуску: Size_Project.exe.
- 2) з використанням командного рядку: Size_Project.exe

5 Вихідні та результуючі дані

Вихідні дані програми (назва проекту для оцінювання (name of project), число класів (number of classes), середня кількість методів (average methods)) повинні вводитися в спеціальну форму.

Результуючі дані програми (прогнозований розмір програмного застосунку (thousand lines of code (в тисячах строк коду – KLOC)), довірчий інтервал (confidence interval) та інтервал прогнозування (prediction interval)) мають відображатися на екрані користувача.

ДОДАТОК В

ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ ПЗ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНИХ ЗАСТОСУНКІВ НА TYPESCRIPT ДЛЯ КРИПТОВАЛЮТНИХ ОПЕРАЦІЙ

1 Об'єкт випробувань

Об'єктом випробувань є ПЗ для оцінювання розміру програмних застосунків на для TypeScript криптовалютних операцій.

2 Мета випробувань

Метою випробувань є перевірка програмного забезпечення на наявність помилок і відповідність реалізованих функцій зазначеним у технічному завданні, яке наведене у додатку А. У випадку прийнятного рівня відповідності програмного забезпечення вимогам, відсутності виявлених помилок і грубих недоліків, програмне забезпечення приймається в експлуатацію.

3 Вимоги до програми

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

Для проведення випробувань розглядали всі функції.

4 Вимоги до програмної документації

Для проведення випробувань надаються наступні документи:

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

5 Засоби і порядок випробувань

Випробування проводилось в Windows 10.

Для роботи із програмою необхідний наступний склад технічних засобів з мінімальними параметрами:

- ПК з процесором не нижче AMD FX-6300;
- обсяг оперативної пам'яті не менше 2 Гб;
- необхідний обсяг на жорсткому диску не менше 600 Мб.

Порядок проведення випробувань.

Випробування програмного забезпечення повинні виконуватися в наступному порядку:

- виконуються інструкції до кожної функції;
- робиться аналіз отриманих результатів і встановлюється відповідність програмного продукту вимогам і системним документам.

При виявленні помилок у роботі системи складається їх перелік і обговорюється термін їх виправлення розробником. Після цього замовник проводить повторне тестування в повному обсязі (можливе використання нових чи додаткових тестів).

6 Методи випробувань

Випробування будемо проводити за стратегією «чорного ящика». Всі функції програми будуть проходити випробування. Розглянемо випробування функцій:

Функція «Корегування інформації для прогнозування розміру програмних застосунків на TypeScript для криптовалютних операцій»

- Запустити програмне забезпечення.
- Ввести дані для прогнозування розміру.
- Повернутися до вводу даних.
- Ввести змінені дані.

Очікуваний результат: змінені дані повинні вивестися на екран.

Функція «Прогнозування розміру програмних застосунків на TypeScript для криптовалютних операцій»

- Запустити програмне забезпечення.
- Ввести дані для прогнозування розміру: назву проекту для оцінювання (name of project), число класів (number of classes), середню кількість методів (average methods).
- Натиснути кнопку «Run».

Очікуваний результат: Розраховане прогнозне значення розміру, інтервал прогнозування та інтервал довіри мають вивестися на екран

ДОДАТОК Г
ІНСТРУКЦІЯ КОРИСТУВАЧА ПЗ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ
ПРОГРАМНИХ ЗАСТОСУНКІВ НА TYPESCRIPT ДЛЯ
КРИПТОВАЛЮТНИХ ОПЕРАЦІЙ

Для запуску програми необхідно запустити файл Size_Project.exe. Після запуску програми відкриється вікно для введення інформації (рис. Г.1).

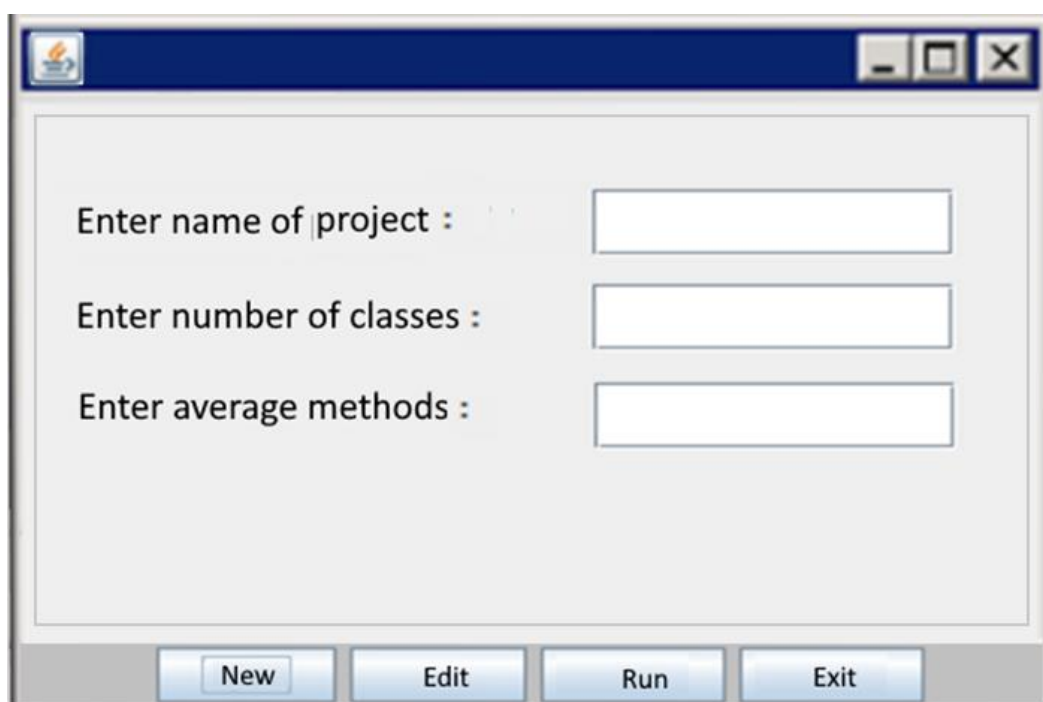
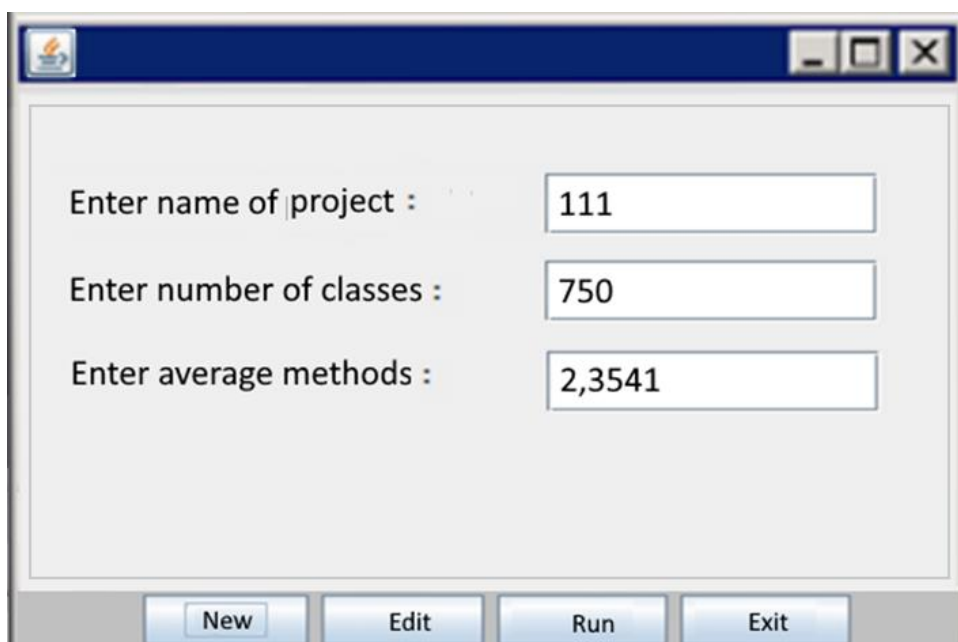


Рисунок Г.1– Вікно введення інформації для оцінювання розміру програмного застосунку

В вікні необхідно ввести наступну інформацію: назву проекту для оцінювання (name of project), число класів (number of classes), середню кількість методів (average methods).

Вікно з вихідними даними представлено на рис. Г.2: назва проекту – 111; кількість класів – 750; середня кількість методів на клас – 2,3541.

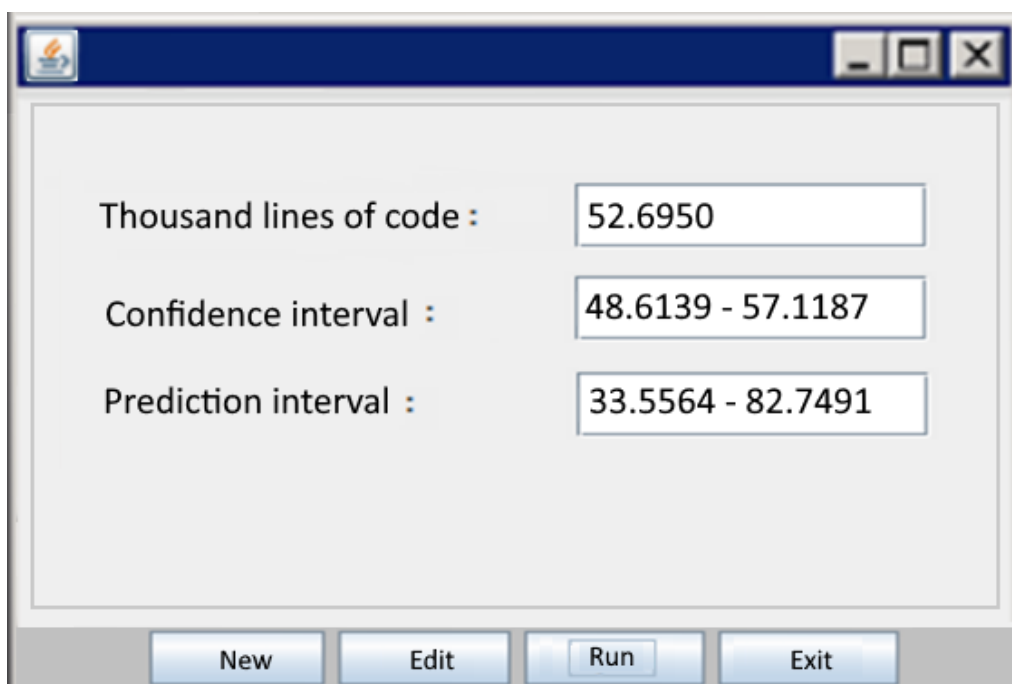


A screenshot of a software window with a blue title bar and standard Windows window controls. The window contains three input fields with labels to their left: 'Enter name of project :', 'Enter number of classes :', and 'Enter average methods :'. The input fields contain the values '111', '750', and '2,3541' respectively. At the bottom of the window, there are four buttons: 'New', 'Edit', 'Run', and 'Exit'.

Label	Value
Enter name of project :	111
Enter number of classes :	750
Enter average methods :	2,3541

Рисунок Г.2 – Вікно з заданою інформацією для оцінювання розміру програмного застосунку

Після введення даних для виконання оцінювання необхідно натиснути на кнопку Run. Отримаємо результати оцінювання, які представлені на рис. Г.3.



A screenshot of a software window showing the results of an evaluation. It has the same blue title bar and controls as the previous window. The window displays three output fields with labels to their left: 'Thousand lines of code :', 'Confidence interval :', and 'Prediction interval :'. The output fields contain the values '52.6950', '48.6139 - 57.1187', and '33.5564 - 82.7491' respectively. At the bottom, the buttons are 'New', 'Edit', 'Run', and 'Exit'.

Label	Value
Thousand lines of code :	52.6950
Confidence interval :	48.6139 - 57.1187
Prediction interval :	33.5564 - 82.7491

Рисунок Г.3 – Вікно з результатами оцінювання

В вікні представлена наступна інформація:

- прогнозований розмір програмного застосунку (thousand lines of code (в тисячах строк коду – KLOC)) – 52,6950;
- довірчий інтервал (confidence interval) – [48,6139; 57,1187];
- інтервал прогнозування (prediction interval) – [33,5564; 82,7491].

Для корегування вихідних даних необхідно натиснути Edit, для введення нових даних для розрахунку – New, для завершення робіт програми – Exit.

ДОДАТОК Д

ТЕКСТ ПЗ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПРОГРАМНИХ ЗАСТОСУНКІВ НА TYPESCRIPT ДЛЯ КРИПТОВАЛЮТНИХ ОПЕРАЦІЙ

```

/**
 * Конструктор принимающий в качестве параметра название проекта.
 * @param title название проекта.
 * @throws EmptyLineException пустая строка.
 * @throws NullPointerException пустая ссылка.
 */
public      NameProj(String      title)      throws      EmptyLineException,
NullPointerException{
    if(title == null)
        throw new NullPointerException();

    title = title.trim();
    if(title.length()==0)
        throw new EmptyLineException();

    this.title = title;
}

}

DanDljaOcen.java
package com.jcs.mat;

import com.jcs.exceptions.*;
import com.jcs.Predictor;

```

```
public interface DanDljaOcen {
```

```
    default double getTStjudent(){  
        return 2.0210;  
    }
```

```
    default double getSS(){  
        return 0.0948;  
    }
```

```
    default int    getN(){  
        return 43;  
    }
```

```
/**  
 * Параметр регрессии B0  
 */  
double B0 = -0.9347;
```

```
/**  
 * Параметр регрессии B1  
 */  
double B1 = 0.8751;
```

```
/**  
 * Параметр регрессии B2  
 */  
double B2 = 0.3723;
```

```
/**
```

* Возвращает обратную матрицу от умножения матриц центрованных регрессоров.

* @return обратная матрица.

*/

```
default double[][] getInverseMatix(){
```

```
    double Obrat[][] = {
```

```
        {0.3044,-0.2896},
```

```
        {-0.2896,0.9307},
```

```
    };
```

```
    return Obrat;
```

```
}
```

```
default double[] getVectorMiddle(){
```

```
    double sred[] = {2.7637, 0.4965};
```

```
    return sred;
```

```
}
```

```
/**
```

* Нормализация исходных данных.

* @param x1 количество классов.

* @param x2 среднее количество методов.

* @return массив нормализованных данных.

* @throws EqualsZeroException равны нулю.

* @throws LessThanZeroException меньше нуля.

*/

```
default double[] toNormalVector(double x1,double x2)throws
EqualsZeroException, LessThanZeroException{
    if(x1<0.0 || x2<0.0)
```



```

        throw new EqualsZeroException();
    if(x1==0.0 || x2==0.0)
        throw new LessThanZeroException();

    double m[] = new double[2];

    m[0]  =  Math.log(x1);
    m[1]  =  Math.log(x2);

    return m;
}

/**
 * Нелинейное уравнение.
 * @param x1    количество классов.
 * @param x2    среднее количество методов.
 * @return      оценка размера кода.
 * @throws EqualsZeroException    равны нулю.
 * @throws LessThanZeroException  меньше нуля.
 */
default double CodeSize(double x1,double x2)throws EqualsZeroException,
LessThanZeroException{
    if(x1<0.0 || x2<0.0)
        throw new EqualsZeroException();
    if(x1==0.0 || x2==0.0)
        throw new LessThanZeroException();

    return Math.pow(10, B0)*Math.pow(x1, B1)*Math.pow(x2, B2);
}

```

```

/**
 * Нелинейное уравнение.
 * @param sc    начальные данные (X1,X2).
 * @return      оценка размера кода.
 * @throws NullPointerException пустой указатель.
 */
default double CodeSize(Predictor sc)throws NullPointerException{
    if(sc == null)
        throw new NullPointerException();

    return Math.pow(10, B0)*Math.pow(sc.getX1(), B1)*Math.pow(sc.getX2(),
B2);
}

/**
 * Возвращает нормализованную оценку длины кода,
 * исходные данные не нормализованы.
 * @param x1    количество классов.
 * @param x2    среднее количество методов.
 * @return      нормализованная оценка длины кода.
 * @throws EqualsZeroException    равны нулю.
 * @throws LessThanZeroException  меньше нуля.
 */
default      double      NormalCodeSize(double      x1,double      x2)throws
EqualsZeroException, LessThanZeroException{
    if(x1<0.0 || x2<0.0)
        throw new EqualsZeroException();
    if(x1==0.0 || x2==0.0)
        throw new LessThanZeroException();

```

```

        return B0+Math.log(x1)*B1+Math.log(x2)*B2;
    }

    /**
     * Возвращает нормализованную оценку длины кода,
     * исходные данные не нормализованы.
     * @param sc начальные данные (X1,X2).
     * @return нормализованная оценка длины кода.
     * @throws NullPointerException пустой указатель.
     */
    default double NormalCodeSize(Predictor sc)throws NullPointerException{
        if(sc == null)
            throw new NullPointerException();

        return B0+Math.log(sc.getX1())*B1+Math.log(sc.getX2())*B2;
    }
}

```