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
_____ проф. Приходько С.Б.
«___» _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «магістр»

на тему: Нелінійна регресійна модель для оцінювання тривалості розробки застосунків на мові C# для платформи PC, та створення програми для її реалізації.

Виконав: студент групи 6151мз
_____Кольцов Є.В.
(підпис, ПІБ)
Керівник роботи:
доцент кафедри ПЗАС, к.т.н, доцент.
(посада, науковий ступень вчене звання)
_____Пухалевич А.В.
(підпис, ПІБ)

Миколаїв – 2024 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)
« 14 » жовтня 2024 р.

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

Студенту Кольцову Євгенію Вікторовичу _____
(Прізвище, ім'я, по батькові)

1. Тема роботи: Нелінійна регресійна модель для оцінювання тривалості розробки застосунків на мові C# для платформи PC, та створення програми для її реалізації. _____
Керівник роботи доцент кафедри ПЗАС, к.т.н, доцент Пухалевич А.В._____
Затверджені наказом ректора № 1070-УЧ від « 11 » _____10_____ 2024 р.
2. Термін подання роботи: 09.12.2024 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; Розділ з розробки 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 Титульний слайд, Актуальність теми, Мета і завдання дослідження, Об'єкт і предмет дослідження, Методи дослідження, Наукова новизна одержаних результатів, Практичне значення одержаних результатів, Апробація результатів досліджень та публікації, Емпіричні дані, Обґрунтування необхідності подальшого дослідження, Нелінійна регресійна модель для оцінювання тривалості розробки застосунків на мові C# для платформи PC, Діаграма прецедентів, Діаграма діяльності, Діаграма станів, Екранні форми, Висновки.

6. Консультанти розділів роботи

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

7. Дата видачі завдання 14.10.2024 р.

К А Л Е Н Д А Р Н И Й П Л А Н

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу ВСТУП	16.10.2024	НС*
2.	Підготовка розділу з огляду літератури та обґрунтування необхідності проведення досліджень за обраною темою	18.10.2024	НС*
3.	Підготовка розділу (ів) за результатами власних досліджень	21.10.2024	НС*
4.	Підготовка розділу з розробки програмного забезпечення	27.11.2024	
5.	Підготовка організаційно-економічного розділу	29.11.2024	
6.	Підготовка розділу з охорони праці	02.12.2024	
7.	Підготовка розділу з охорони навколишнього середовища	04.12.2024	
8.	Підготовка розділу ВИСНОВКИ	05.12.2024	
9.	Оформлення списку використаних джерел та додатків	06.12.2024	
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.)	09.12.2024	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку)
11.	Підготовка презентації та доповіді	13.12.2024	
12.	Попередній захист роботи на засіданні кафедри ПЗАС	16.12.2024	
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	19.12.2024	

* - за результатами наукового стажування (НС), яке було з 02.09.2024 по 13.10.2024 р.

Студент _____ Кольцов Є.В.
(ПІБ)

Керівник роботи _____ Пухалевич А.В.
(ПІБ)

РЕФЕРАТ

Кольцов Євгеній Вікторович

«Нелінійна регресійна модель для оцінювання тривалості розробки застосунків на мові C# для платформи PC, та створення програми для її реалізації»

Кваліфікаційна робота на здобуття освітнього рівня магістра зі спеціальності 121 «Інженерія програмного забезпечення». Національний університет кораблебудування імені адмірала Макарова. Миколаїв, 2024 р.

Обсяг роботи: 89 стор., 8 табл., 4 рис., 21 використане джерело, 5 додатків.

Актуальність теми: Найчастіше тривалість розробки визначається залежністю від її трудомісткості. Для цього використовуються регресійні моделі, побудовані із застосуванням нормалізуючих перетворень. Проте більшість існуючих моделей орієнтовані на інші мови програмування або враховують лише платформу, для якої створюється ПЗ, що може знизити достовірність оцінювання тривалості розробки застосунків саме на C# для платформи PC. Тому розробка математичної моделі для достовірного оцінювання тривалості розробки програмних застосунків на C# для платформи PC є важливим і актуальним завданням.

Мета та завдання дослідження: підвищення достовірності оцінювання тривалості розробки застосунків на мові C# для платформи PC, за рахунок удосконалення нелінійної регресійної моделі для оцінювання та розробки програми для її реалізації, що забезпечить більшу достовірність прогнозування тривалості розробки застосунків на мові C# для платформи PC на ранніх стадіях розробки. Завдання: проаналізувати існуючі моделі для оцінювання тривалості розробки програмних застосунків, побудувати нелінійну регресійну модель для оцінювання тривалості розробки застосунків на мові C# для платформи PC, розробити проект програми для оцінювання тривалості розробки застосунків на мові C# для платформи PC.

Об'єкт дослідження: процес оцінювання тривалості розробки застосунків на мові C# для платформи PC.

Предмет дослідження: нелінійна регресійна модель для оцінювання тривалості розробки застосунків на мові C# для платформи PC.

Методи дослідження: методи теорії ймовірностей, математичної статистики, математичного моделювання, регресійного аналізу, об'єктно-орієнтованого програмування.

Наукова новизна: удосконалено нелінійну регресійну модель для оцінювання тривалості розробки застосунків на мові C# для платформи PC за рахунок застосування нормалізуючого перетворення десяткового логарифму і викидів регресії, що дозволило підвищити достовірність оцінювання тривалості розробки застосунків на мові C# для платформи PC в порівнянні з існуючими моделями COCOMO та ISBSG.

Практичне значення одержаних результатів: полягає у розробці програми для оцінювання тривалості розробки застосунків на мові C# для платформи PC.

Апробація результатів дослідження: Основні положення і результати досліджень пройшли апробацію на V Всеукраїнській науково-практичній інтернет-конференції «Інформаційні технології: моделі, алгоритми, системи (ITMAS-2024)» (30-31 жовтня 2024, м. Миколаїв).

Публікації: За результатами досліджень опубліковано одну наукову працю - матеріали конференції.

Ключові слова: програмне забезпечення, оцінювання розміру, нелінійна регресійна модель, метрики програмного забезпечення, C#.

ABSTRACT

Koltsov Yevhenii Viktorovych

«A nonlinear regression model for estimating the development duration of C# applications for the PC platform and creating software for its implementation»

Qualification work for obtaining a master's degree in the speciality 121 «Software Engineering». Admiral Makarov National University of Shipbuilding. Mykolaiv, 2024.

Volume of work: 89 pages, 8 tables, 4 figures, 21 references, 5 appendices.

Relevance of the topic of the work: Often the duration of development is determined by the dependence on its efforts. For this, regression models built using normalizing transformations are used. However, most existing models are focused on other programming languages or take into account only the platform for which the software is created, which can reduce the reliability of estimating the duration of application development in C # for the PC platform. Therefore, the development of a mathematical model for a reliable assessment of the duration of the development of software applications in C # for the PC platform is an important and urgent task.

Purpose and objectives of the study: higher reliability of estimating the duration of developing applications in C # for the PC platform, by improving a non-linear regression model for estimating and developing an application for its implementation, which will provide greater reliability of predicting the duration of developing applications in C # for the PC platform at early stages of development. Task: to analyze existing models for estimating the duration of development of software applications, to build a nonlinear regression model for estimating the duration of development of applications in C # for the PC platform, to develop a software project for estimating the duration of development of applications in C # for the PC platform.

The object of the study: process of estimating the duration of application development in C # for the PC platform.

Subject of the study: a nonlinear regression model to estimate the duration of C # application development for the PC platform.

Methods of the study: methods of probability theory, mathematical statistics, mathematical modeling, regression analysis, object-oriented programming.

Scientific novelty of the results obtained: improved nonlinear regression model for estimation of duration of application development in C # language for PC platform due to application of normalizing transformation of decimal logarithm and regression outliers, which allowed to increase reliability of estimation of duration of application development in C # language for PC platform in comparison with existing COCOMO and ISBSG models.

Practical significance of the obtained results: developing a program to estimate the duration of application development in C # for the PC platform.

Testing of the research results: The main provisions and results of the research were tested at the V All-Ukrainian Scientific and Practical Internet Conference “Information Technologies: Models, Algorithms, Systems (ITMAS-2024)” (October 30-31, 2024, Mykolaiv).

Publications: According to the results of the research, one scientific work was published - the conference materials.

Keywords: software, size estimation, nonlinear regression model, software metrics, C#.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

МНК – метод найменших квадратів

ПЗ – програмне забезпечення

COCOMO – Constructive Cost Model

ISBSG – International Software Benchmarking Standards Group

FP – functional points

KLOC – thousand lines of code

MMRE – Mean Magnitude of Relative Error

Pred – Percentage of Prediction

SMD – squared Mahalanobis distance

UML – Unified Modeling Language

ВСТУП

Актуальність теми. На теперішній час мова програмування C# є одним із найпоширеніших інструментів для створення програмних застосунків на платформі PC. У зв’язку з цим перед розробниками часто постає практична задача інженерії програмного забезпечення, пов’язана з оцінюванням тривалості розробки таких застосунків. Достовірність такого оцінювання має велике значення, адже від цього залежить вибір замовником виконавця: замовник надає перевагу розробникам, які пропонують більш реалістичні терміни. У той же час, занадто оптимістична оцінка тривалості може призвести до порушення строків виконання проекту та штрафних санкцій для виконавця.

Найчастіше тривалість розробки визначається залежністю від її трудомісткості. Для цього використовуються регресійні моделі, побудовані із застосуванням нормалізуючих перетворень [1]. Було розроблено низку нелінійних регресійних моделей [2 – 5], які довели свою ефективність. Проте більшість існуючих моделей орієнтовані на інші мови програмування або враховують лише платформу, для якої створюється ПЗ, що може знизити достовірність оцінювання тривалості розробки застосунків саме на C# для платформи PC. Тому розробка математичної моделі для достовірного оцінювання тривалості розробки програмних застосунків на C# для платформи PC є важливим і актуальним завданням.

Мета і завдання дослідження. Метою дослідження є підвищення достовірності оцінювання тривалості розробки застосунків на мові C# для платформи PC, за рахунок удосконалення нелінійної регресійної моделі для оцінювання та розробки програми для її реалізації, що забезпечить більшу достовірність прогнозування тривалості розробки застосунків на мові C# для платформи PC на ранніх стадіях розробки.

Для досягнення поставленої мети необхідно вирішити такі завдання дослідження:

– проаналізувати основні метрики, які використовуються для оцінювання тривалості розробки програмних застосунків;

– проаналізувати існуючі моделі для оцінювання тривалості розробки програмних застосунків;

– обґрунтувати необхідність подальшого дослідження за обраною темою;

– проаналізувати математичну модель для оцінювання тривалості розробки програмних застосунків;

– побудувати нелінійну регресійну модель для оцінювання тривалості розробки застосунків на мові C# для платформи PC;

– виконати постановку задачі на розробку програми для оцінювання тривалості розробки застосунків на мові C# для платформи PC;

– розробити ескізний проект програми для оцінювання тривалості розробки застосунків на мові C# для платформи PC;

– розробити технічний проект програми для оцінювання тривалості розробки застосунків на мові C# для платформи PC;

– розробити робочий проект програми для оцінювання тривалості розробки застосунків на мові C# для платформи PC.

Об'єктом дослідження є процес оцінювання тривалості розробки застосунків на мові C# для платформи PC.

Предметом дослідження є нелінійна регресійна модель для оцінювання тривалості розробки застосунків на мові C# для платформи PC.

Методи дослідження Для вирішення поставлених завдань були застосовані методи теорії ймовірностей, математичної статистики, математичного моделювання, регресійного аналізу, об'єктно-орієнтованого програмування.

Наукова новизна отриманих результатів Удосконалено нелінійну регресійну модель для оцінювання тривалості розробки застосунків на мові C# для платформи PC за рахунок застосування нормалізуючого перетворення десяткового логарифму і викидів регресії, що дозволило підвищити достовірність оцінювання тривалості розробки застосунків на мові C# для платформи PC в порівнянні з існуючими моделями COCOMO та ISBSG

Практичне значення одержаних результатів полягає у розробці програми для оцінювання тривалості розробки застосунків на мові C# для платформи PC.

Особистий внесок здобувача. Кваліфікаційна робота є самостійно виконаною працею. Усі результати, викладені у роботі, отримані автором особисто. У праці, яка опублікована у співавторстві [6], здобувачеві належить: побудова нелінійної регресійної моделі для оцінювання тривалості розробки застосунків на мові C# для платформи PC.

Апробація результатів досліджень. Основні положення і результати досліджень пройшли апробацію на V Всеукраїнській науково-практичній інтернет-конференції «Інформаційні технології: моделі, алгоритми, системи (ITMAS-2024)» (30-31 жовтня 2024, м. Миколаїв).

Публікації. За результатами досліджень опубліковано одну наукову працю – матеріали конференції.

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ТА МОДЕЛЕЙ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ РОЗРОБКИ ПРОГРАМНИХ ЗАСТОСУНКІВ

1.1 Аналіз основних метрик, які використовуються для оцінювання тривалості розробки програмних застосунків

Метрики ПЗ є важливим інструментом у процесі його створення, оскільки вони забезпечують кількісну оцінку різних аспектів ПЗ і його розробки. Це дає розробникам можливість об'єктивно оцінювати, аналізувати та планувати трудовитрати й терміни виконання проєктів [7].

Серед найпоширеніших метрик, що застосовуються для оцінювання тривалості розробки, виділяють:

– Тривалість розробки – час, необхідний для створення ПЗ;

– Трудомісткість розробки – вимірюється як обсяг ресурсів (зазвичай в людино-годинах чи людино-місяцях), витрачених на реалізацію проєкту;

Для оцінки трудомісткості часто використовуються такі показники:

– KLOC (кількість рядків коду) – вимірює розмір ПЗ за кількістю тисяч рядків вихідного коду програми [8];

– FP (функціональні точки) – оцінює функціональні можливості системи, враховуючи кількість елементарних функцій, які виконуються ПЗ [9].

Ці метрики дозволяють точніше прогнозувати обсяг роботи, необхідний для реалізації проєкту, і допомагають оптимізувати процеси управління розробкою.

Варто також звернути увагу на додаткові аспекти, які впливають на оцінювання трудомісткості, тривалості та загальної ефективності розробки ПЗ. Метрики не лише дозволяють прогнозувати певні параметри, але й допомагають

контролювати якість, виявляти проблеми на ранніх етапах розробки проєкту та підвищувати продуктивність команди [10].

Використання метрик ПЗ має низку переваг:

– з точки зору прогнозування: дозволяє оцінити час і ресурси, необхідні для виконання проєкту;

– з точки зору оптимізації: допомагає виявити неефективні процеси та знайти шляхи їх покращення;

– з точки зору контролю якості: забезпечує об’єктивну оцінку якості коду та стабільної прогнозованої поведінки ПЗ;

– з точки зору прийняття управлінських рішень: надає дані для обґрунтування вибору технологій, підходів чи змін у проєкті.

Однак, наряду з перевагами, застосування метрик ПЗ має і певні труднощі:

– залежність від мови програмування, наприклад, метрика KLOC, яка може не бути показовою для скриптових мов або мов із високим рівнем абстракції;

– нестандартність отриманих значень: у різних проєктах метрики можуть мати різне значення, що ускладнює їхнє порівняння;

– труднощі в інтерпретації: некоректна інтерпретація отриманих метрик може призвести до помилкових управлінських рішень стосовно проєкту ПЗ.

Використання метрик ПЗ дозволяє не лише покращити процес розробки, але й підвищити ефективність та якість кінцевого продукту. Однак для їх успішного застосування важливо обирати відповідні метрики для кожного конкретного проєкту ПЗ.

1.2 Аналіз існуючих моделей для оцінювання тривалості розробки програмних застосунків

Оцінювання тривалості розробки ПЗ є важливим завданням у програмній інженерії, яке потребує використання адекватних та точних моделей. Такі моделі дозволять ефективно передбачати часові витрати на розробку ПЗ, враховувати ресурси та підвищувати ефективність реалізації проєкту. Серед основних моделей, що використовуються для оцінювання тривалості розробки ПЗ, можна відмітити наступні.

Метод функціональних точок (FP method) базується на оцінці кількості функцій, які повинні бути реалізовані. Він дозволяє оцінити тривалість розробки без прив’язки до мови програмування або платформи, що робить його універсальним. Метод враховує обсяг вхідних даних, обсяг вихідних даних, кількість запитів до системи, складність інтеграції з іншими системами та підходить для ранніх етапів оцінювання, коли обсяг коду ще не визначений [9].

Метод Delphi є розширенням експертного підходу. Група експертів незалежно оцінює тривалість, а потім результати обговорюються для досягнення консенсусу. Метод Delphi підвищує достовірність оцінювання, зменшуючи суб’єктивний вплив окремих експертів [11].

Але основними моделями для оцінювання тривалості розробки програмних застосунків є регресійні моделі, які визначають залежність тривалості розробки від різних параметрів, таких як: трудомісткість, кількість рядків коду KLOC, кількість функціональних точок FP та інших.

Лінійні регресійні моделі прості у використанні, передбачають лінійну залежність між обраними параметрами, але застосовувати їх можливо тільки в тому випадку, коли емпіричні дані для їх побудови відповідають нормальному закону розподілу, що, як було показано в низці робіт [3–5], справедливо не завжди.

Нелінійні регресійні моделі враховують складніші взаємозв'язки між параметрами, але вони і складніші в побудові. Для їх побудови часто використовуються нормалізуючі перетворення, які приводять початкові дані до вигляду, що відповідає нормальному закону розподілу. Це спрощує побудову регресійних моделей та підвищує достовірність прогнозів, які робляться на їх основі. Наприклад, нелінійні залежності між емпіричними даними можна перевести в лінійні за допомогою логарифмів або інших перетворень.

І, звичайно, для оцінювання тривалості розробки програмних застосунків широко використовується модель COCOMO, яка є одною із найвідоміших моделей оцінювання тривалості. Вона використовує параметри проєкту для прогнозування трудомісткості, яка потім переводиться у часові витрати на розробку проєкту [12]. Модель COCOMO має три рівні складності:

– базовий, який використовує лише деякі ключові параметри, такі як KLOC;

– проміжний, який враховує додаткові фактори, наприклад, досвід команди;

– розширений, який аналізує всі етапи розробки ПЗ та деталі, так, як дизайн, тестування та інтеграція.

Модель COCOMO представляє собою нелінійну регресійну модель, яка побудована на основі нормалізуючого перетворення десяткового логарифму.

Модель ISBSG – це модель оцінювання трудомісткості та тривалості розробки ПЗ, яка базується на аналізі великого обсягу історичних даних з реальних проєктів та дозволяє створювати прогнози для нових проєктів.

Одним із ключових параметрів, на якому базується ISBSG, є функціональні точки. Ця методологія використовується для вимірювання обсягу роботи в термінах функціональності, яку надає ПЗ, а не кількості рядків коду. Модель дозволяє адаптувати прогнози для проєктів різних типів, незалежно від мови програмування, платформи чи технології розробки.

Модель ISBSG широко використовується, коли потрібно оцінити нові проєкти, порівняти продуктивність проєкту зі схожими проєктами, або у питаннях навчання та дослідження закономірностей у розробці ПЗ [13].

Модель ISBSG також представляє собою нелінійну регресійну модель, яка побудована на основі нормалізуючого перетворення натурального логарифму.

Зі зростанням популярності гнучких методологій розробки ПЗ та систем штучного інтелекту, метрики оцінювання розробки ПЗ набувають нових форм. Це і використання машинного навчання для прогнозування тривалості і виявлення потенційних ризиків проєктів, і інтеграція з інструментами аналізу великих даних для більш точного оцінювання продуктивності команд і проєктів, і розробка метрик для хмарних застосунків, мікросервісів та AI-додатків.

Але потрібно також відмітити, що більшість моделей розроблені для інших мов програмування, наприклад, Java чи Python, або орієнтовані виключно на платформу, а не на специфіку мови C#. Це знижує точність оцінок, оскільки в загальних моделях не враховуються особливості C#, наприклад, типізація чи специфічні бібліотеки, що можуть суттєво впливати на обсяг роботи, та інструменти і технології, притаманні для C#.

1.3 Обґрунтування необхідності подальшого дослідження за обраною темою

Оскільки C# – мова програмування високого рівня, яка активно використовується для розробки застосунків для платформи PC завдяки інтеграції з .NET Framework та доступом до широкого спектру бібліотек і функціональності, що пришвидшує розробку, має ряд зручностей для розробки застосунків завдяки Windows Form, WPF (Windows Presentation Foundation) та ASP.NET, проте C# також має свої виклики, які впливають на оцінювання

тривалості розробки, такі як залежність від архітектури програми, особливості платформи та інтеграції з іншими системами.

Тому для обґрунтування необхідності подальшого дослідження за обраною темою, а саме побудовою нелінійної регресійної моделі для оцінювання тривалості розробки застосунків на мові C# для платформи PC, потрібно в першу чергу здійснити побудову існуючих моделей за емпіричними даними наявних застосунків на мові C# для платформи PC.

Для побудови моделей використаємо емпіричні дані репозитарію ISBSG релізу 2021 року по відповідним завершеним програмним проєктам. Візьмемо емпіричні дані з 38 проєктів розробки застосунків на мові C# для платформи PC.

Використовуючи модель COCOMO, візьмемо наступне рівняння для базового рівня складності [14]:

$$Y = 0{,}371 * X^{0{,}38}. \quad (1.1)$$

Використовуючи модель ISBSG, візьмемо наступне рівняння [14]:

$$Y = 0{,}662 * X^{0{,}328}. \quad (1.2)$$

У рівняннях (1.1) та (1.2) використовуються такі позначення змінних:

Y – тривалість проекту в місяцях,

X – трудомісткість проекту в людино-годинах.

Також використаємо модель для розрахунку залежності тривалості проєкту від його трудомісткісті, яка наведена у роботі [3]:

$$\hat{Y} = 10^{-0{,}3964} * X^{0{,}3108}. \quad (1.3)$$

Для порівняння побудованих моделей між собою, використаємо такі параметри якості.

– *MMRE* – середнє значення відносної похибки, яке розраховується за формулою [15]:

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} MRE_i\,, \tag{1.4}$$

де: MRE_i – величина відносної похибки,

n – кількість значень у вибірці.

Значення *MRE* розраховується за формулою [15]:

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

де: y_i – емпіричне значення y,

$\hat{y}i$ – значення y, розраховане за рівнянням регресії.

– *Pred*(0,25) – рівень прогнозування, який розраховується за формулою [15]:

$$Pred\,(0{,}25) = \frac{k}{n}\,, \tag{1.5}$$

де k – кількість значень з $MRE \le 0{,}25$,

n – кількість значень у вибірці.

Параметри якості моделей, побудованих за рівняннями (1.1) – (1.3), та розраховані за формулами (1.4), (1.5), наведено у таблиці 1.1.

Таблиця 1.1 – Параметри якості побудованих моделей

Параметр	Модель COCOMO	Модель ISBSG	Модель (1.3)
MMRE	0,5773	0,6614	0,5594
Pred(0,25)	0,2632	0,2105	0,2105

Враховуючи отримані результати, які наведені у таблиці 1.1, не можна використовувати моделі, за якими вони були отримані, для оцінювання тривалості розробки застосунків на мові C# для платформи PC, оскільки жодна із побудованих моделей не задовільняє по обом параметрам якості, які повинні бути такими: середня величина відносної похибки $MMRE<0,25$, рівень прогнозування $Pred(0,25)>0,75$.

Розробка спеціалізованої моделі для оцінювання тривалості розробки застосунків на мові C# для платформи PC дозволить не лише покращити достовірність оцінювання, але й оптимізувати процес розробки відповідних програмних застосунків, сприятиме ефективному управлінню розробкою, підвищенню довіри замовників і зниженню ризиків.

2 РОЗРОБКА НЕЛІНІЙНОЇ РЕГРЕСІЙНОЇ МОДЕЛІ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ РОЗРОБКИ ЗАСТОСУНКІВ НА МОВІ C# ДЛЯ ПЛАТФОРМИ PC

2.1 Математична модель для оцінювання тривалості розробки програмних застосунків

Для оцінювання тривалості розробки програмних застосунків можна використовувати регресійні моделі, як лінійні, так і нелінійні.

В загальному випадку лінійна регресійна модель має такий вигляд [15]:

$$z_y = \hat{z}_y + \varepsilon = \hat{b}_0 + \hat{b}_1 z_x + \varepsilon, \quad (2.1)$$

де: z_y – залежна змінна,

$\hat{z}_y$ – значення залежної змінної z_y, розраховане за лінійною регресійною моделлю,

ε – випадкова похибка, нормально розподілена випадкова величина,

$\hat{b}_0$, $\hat{b}_1$, – оцінки коефіцієнтів лінійної регресії,

z_x – незалежна змінна.

Перевірку випадкової похибки ε на відповідність нормальному закону розподілу можна виконати за критерієм згоди Пірсона χ^2 згідно з [15].

Оцінки коефіцієнтів лінійної регресії $\hat{b}_0$ та $\hat{b}_1$, можна знайти за допомогою МНК згідно з [15]:

$$b_1 = \frac{n\sum_{i=1}^{n} z_{xi} z_{yi} - \sum_{i=1}^{n} z_{xi} \cdot \sum_{i=1}^{n} z_{yi}}{n\sum_{i=1}^{n} z_{xi}^2 - \left(\sum_{i=1}^{n} z_{xi}\right)^2}, \quad (2.2)$$

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

Однак використовувати лінійну регресійну модель можна тільки у випадку нормального закону розподілу випадкових величин.

В загальному випадку нелінійна регресійна модель має такий вигляд [15]:

$$\boldsymbol{y} = \hat{\boldsymbol{y}} + \boldsymbol{\varepsilon} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{\varepsilon}, \quad (2.4)$$

де: y – залежна змінна,

$\hat{y}$ – значення залежної змінної y, розраховане за нелінійною регресійною моделлю,

ε – випадкова похибка, нормально розподілена випадкова величина,

$f(x)$ – функція, яка визначає конкретний вигляд регресійної моделі,

x – незалежна змінна.

В загальному випадку, коли емпіричні дані не відповідають нормальному закону розподілу випадкових величин, потрібно застосовувати відповідні нормалізуючі перетворення, які дозволять перейти від ненормально розподілених випадкових величин до нормально розподілених випадкових величин та застосувати до них лінійну регресійну модель (2.1).

До таких нормалізуючих перетворень належать логарифмічні перетворення, перетворення Бокса-Кокса, перетворення Джонсона та деякі інші.

Найпростішими є логарифмічні нормалізуючі перетворення, одне з яких, а саме перетворення за десятковим логарифмом [15], і буде застосоване для побудови моделі:

$$\boldsymbol{z = \log_{10}(x)}\,, \tag{2.5}$$

де z – нормалізоване значення випадкової величини,

x – початкове значення випадкової величини.

Для кожного нормалізуючого перетворення існує зворотне перетворення. Для нормалізуючого перетворення за десятковим логарифмом (2.5) таким зворотним перетворенням буде наступне [15]:

$$\boldsymbol{x = 10^{z}}. \tag{2.6}$$

Згідно з [15], у разі застосування в якості нормалізуючого, перетворення за десятковим логарифмом (2.5), нелінійну регресійну модель (2.4) можна представити у такому вигляді:

$$\boldsymbol{\hat{y} = 10^{\varepsilon+\hat{b}_0} x^{\hat{b}_1}}. \tag{2.7}$$

Перш, ніж прийняти рішення про вид регресійної моделі, потрібно перевірити емпіричні дані на відповідність багатовимірному нормальному закону розподілу. Виконати це можна, зокрема, за допомогою багатовимірного ексцесу β_2, оцінка якого визначається як [16, 17]:

$$\hat{\beta}_2 = \frac{1}{n}\sum_{i-1}^{n}\{(X_i - \bar{X})^T S_n^{-1}(X_i - \bar{X})\}^2, \tag{2.8}$$

де: X_i – i-ий компонент даних,

$\bar{X}$ – вектор вибіркових середніх,

n – кількість елементів у вибірці,

S_n – коваріаційна матриця, яка визначається як [17]:

$$S_n = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})\,(X - \bar{X})^T. \quad (2.9)$$

Якщо багатовимірні дані розподілені за нормальним законом, для них можна розрахувати граничне значення β_2 за наступною формулою [17]:

$$\beta_2 = m(m+2), \quad (2.10)$$

де m – кількість параметрів у багатовимірних даних.

Якщо багатовимірні дані дійсно розподілені за нормальним законом, розрахункове значення багатовимірного ексцесу β_2, розраховане за формулами (2.8) та (2.9), буде менше за значення, розраховане за формулою (2.10).

Для знаходження та видалення викидів із емпіричних даних, можна застосувати квадрат відстані Махаланобіса SMD, який має вигляд [15]:

$$d_i^2 = (X_i - \bar{X})^2 S_n^{-1}(X_i - \bar{X}), \quad (2.11)$$

де: X_i – i-ий компонент даних,

$\bar{X}$ – вектор вибіркових середніх,

n – кількість елементів у вибірці,

S_N – коваріаційна матриця, яка розраховується згідно з (2.9),

та методику, яка представлена, зокрема, в [18, 19].

Для висновку, чи є точка даних викидом, використаємо значення квантилю розподілу Пірсона χ^2, обчислене при рівні значущості α=0,005.

2.2 Побудова нелінійної регресійної моделі для оцінювання тривалості розробки застосунків на мові C# для платформи PC

Емпіричними даними, що використовувалися при побудові нелінійної регресійної моделі для оцінювання тривалості розробки застосунків на мові C# для платформи PC в залежності від трудомісткості розробки є дані репозитарію ISBSG з релізу 2021 року [13]. Для мови C# кількість емпіричних даних була взята у кількості 38 проєктів.

Ці дані було перевірено на відповідність багатовимірному нормальному закону розподілу згідно з (2.8) – (2.10): $\hat{\beta}_2 = 8{,}12$, граничне значення β_2=8 при m=2. Отже, ці дані не розподілені за нормальним законом, і для подальшої побудови нелінійної регресійної моделі згідно з (2.7), потрібно застосувати нормалізуюче перетворення десяткового логарифму згідно з (2.5).

Перевіримо нормалізовані дані на викиди за допомогою квадрату відстані Махаланобіса SMD згідно з (2.11) та (2.9). Максимальне значення $d_i^2 = 10{,}78$ при значенні квантилю розподілу Пірсона χ^2=10,60. Отже, нормалізовані дані містять викид. Видаляємо його з набору даних та повторюємо перевірку доти, доки з набору нормалізованих даних не будуть видалені всі викиди, і побудова моделі починається з першого етапу..

Якщо після побудови лінійної регресійної моделі закон розподілу випадкової похибки є не був нормальним, то рядок з найбільшим за модулем значенням величини відкидається, і побудова моделі починається з першого етапу.

Останнім етапом видалення викидів є перевірка того, що значення емпіричних даних не виходять за межі інтервалу прогнозування моделі. Якщо

такі значення є, вони відкидаються, і побудова моделі починалась з першого етапу.

Результати перевірки набору нормалізованих даних на викиди наведено в таблиці 2.1.

Таблиця 2.1 – Результати перевірки набору нормалізованих даних на викиди

№ ітерації	Кількість застосунків n	Перевірка на багатовимірну нормальність	Перевірка на викиди за допомогою SMD	Кількість викидів та їх обгрунтування
1	38	$\hat{\beta}_2 = 8{,}12$	$d_i^2 \max = 10{,}78$	1, SMD
2	37	$\hat{\beta}_2 = 6{,}68$	$d_i^2 \max = 7{,}37$	1, ε не є $N\left(0, \sigma_\varepsilon^2\right)$
3	36	$\hat{\beta}_2 = 6{,}53$	$d_i^2 \max = 8{,}08$	1, ε не є $N\left(0, \sigma_\varepsilon^2\right)$
4	35	$\hat{\beta}_2 = 5{,}96$	$d_i^2 \max = 6{,}09$	1, ε не є $N\left(0, \sigma_\varepsilon^2\right)$
5	34	$\hat{\beta}_2 = 5{,}79$	$d_i^2 \max = 5{,}15$	1, ε не є $N\left(0, \sigma_\varepsilon^2\right)$
6	33	$\hat{\beta}_2 = 5{,}86$	$d_i^2 \max = 6{,}06$	1, ε не є $N\left(0, \sigma_\varepsilon^2\right)$
7	32	$\hat{\beta}_2 = 5{,}77$	$d_i^2 \max = 5{,}06$	1, ε не є $N\left(0, \sigma_\varepsilon^2\right)$
8	31	$\hat{\beta}_2 = 5{,}87$	$d_i^2 \max = 5{,}43$	1, інтервал прогнозування
9	30	$\hat{\beta}_2 = 5{,}94$	$d_i^2 \max = 5{,}69$	1, ε не є $N\left(0, \sigma_\varepsilon^2\right)$
10	29	$\hat{\beta}_2 = 5{,}96$	$d_i^2 \max = 6{,}47$	1, інтервал прогнозування
11	28	$\hat{\beta}_2 = 5{,}77$	$d_i^2 \max = 4{,}99$	1, інтервал прогнозування
12	27	$\hat{\beta}_2 = 5{,}94$	$d_i^2 \max = 5{,}30$	1, інтервал прогнозування
13	26	$\hat{\beta}_2 = 6{,}13$	$d_i^2 \max = 6{,}13$	1, інтервал прогнозування
14	25	$\hat{\beta}_2 = 6{,}10$	$d_i^2 \max = 5{,}68$	1, інтервал прогнозування
15	24	$\hat{\beta}_2 = 6{,}33$	$d_i^2 \max = 6{,}72$	1, інтервал прогнозування
16	23	$\hat{\beta}_2 = 5{,}87$	$d_i^2 \max = 5{,}38$	-

Отже, після відкидання з емпіричних даних викидів залишилось 23 значення, для яких будуємо лінійну регресійну модель згідно з (2.1), коефіцієнти лінійної регресії b_0 та b_1 знаходимо згідно з (2.2), (2.3):

$$z_y = 0{,}9920 + 0{,}0465 z_x + \varepsilon.$$

Відповідно, нелінійна регресійна модель для оцінювання тривалості розробки застосунків на мові C# для платформи PC, побудована згідно з (2.7) та (2.6), має остаточний вигляд:

$$\hat{y} = 10^{\varepsilon + 0{,}9920} \; x^{0{,}0465}.$$

Розрахуємо параметри якості побудованої моделі для оцінювання тривалості розробки застосунків на мові C# для платформи PC *MMRE* та *Pred*(0,25), обчислені за формулами (1.4), (1.5): *MMRE*=0,1139, *Pred*(0,25)=0,7826, та порівняємо їх з даними, наведеними у таблиці 1.1.

Із наведених даних видно, що параметри якості побудованої моделі для оцінювання тривалості розробки застосунків на мові C# для платформи PC значно краще наведених у таблиці 1.1 та повністю задовільняють необхідним вимогам: середня величина відносної похибки *MMRE*<0,25, рівень прогнозування *Pred*(0,25)>0,75.

Для підвищення достовірності оцінювання тривалості розробки застосунків на мові C# для платформи PC, окрім розробки спеціалізованих регресійних моделей, адаптованих до особливостей C#, можна також застосовувати:

– включення параметрів, що враховують інтеграцію з .NET Framework та особливості платформи Windows;

– використання комбінованих методів, які поєднують FP, регресійні моделі та експертні оцінки;

– використання машинного навчання для аналізу даних попередніх проєктів, що створювалися на C#, для автоматизації прогнозування.

3 РОЗРОБКА ПРОЕКТУ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ РОЗРОБКИ ЗАСТОСУНКІВ НА МОВІ C# ДЛЯ ПЛАТФОРМИ PC

3.1 Постановка задачі на розробку програми для оцінювання тривалості розробки застосунків на мові C# для платформи PC

В сучасній розробці програмного забезпечення важливо точно оцінювати час, необхідний для виконання проекту. Це дозволяє ефективно планувати ресурси, визначати терміни виконання, контролювати процеси розробки та мінімізувати ризики. Для підвищення достовірності оцінювання тривалості розробки застосунків на мові C# для платформи PC в даній роботі удосконалено нелінійну регресійну модель для проектів даного типу.

Наступним етапом кваліфікаційної роботи є розробка програми для реалізації удосконаленої моделі для оцінювання тривалості розробки застосунків на мові C# для платформи PC. Виходячи з вище викладеного, програма має задовольняти наступним вимогам.

Вимоги до складу виконуваних функцій

Програмне забезпечення повинно мати можливість виконання наступних функцій:

– Внесення параметрів моделі.

– Внесення довірчої ймовірності.

– Внесення інформації про застосунок (назви та трудомісткості застосунку, що розробляється).

– Оцінювання тривалості розробки застосунку.

– Збереження результатів оцінювання в базу даних.

– Перегляд збережених результатів оцінювання.

Вимоги до інформаційної та програмної сумісності

Операційна система Windows 7 чи вище, або інша, в якій встановлена версія JRE не нижче версії Java 1.8.

Вимоги до складу та параметрів технічних засобів

Для функціонування ПЗ рекомендується наступний склад технічних засобів з мінімальними характеристиками:

– ПК з процесором Core i3;

– Оперативна пам'ять –1 Гб;

– Обсяг на жорсткому диску –500 Мб.

Детальна постановка задачі сформульована в технічному завданні, яке наведене в Додатку А.

3.2 Розробка ескізного проекту програми для оцінювання тривалості розробки застосунків на мові C# для платформи PC

Ескізний проект є важливим етапом для отримання загального уявлення про майбутню систему та забезпечує основу для подальших етапів розробки. Він служить для формулювання основних вимог до проекту, його структури та функціональних компонентів, а також для визначення напрямків подальшої розробки.

Діаграма прецедентів ПЗ для оцінювання тривалості розробки застосунків на мові C# для платформи PC

Діаграма прецедентів показує взаємодію користувачів (акторів) з системою, описуючи основні сценарії використання. Вона дає змогу визначити, які завдання або операції система повинна виконувати, а також хто саме їх буде ініціювати. Таким чином діаграма прецедентів є важливим інструментом на

етапі проектування, оскільки вона дозволяє на ранніх етапах зібрати і організувати основні функціональні вимоги до системи.

Перед розробкою діаграми прецедентів необхідно виявити акторів та варіанти використання. Опис акторів ПЗ для оцінювання тривалості розробки застосунків на мові C# для платформи PC наведено в таблиці 3.1. В даній системі буде один актор, якого названо Користувач.

Таблиця 3.1 – Опис акторів ПЗ для оцінювання тривалості розробки застосунків на мові C# для платформи PC

Актори	Короткий опис
Користувач	Усі операції в системі будуть виконуватися одним користувачем, який може вносити параметри моделі, значення довірчої ймовірності, інформації про застосунок, оцінювати час, необхідний для розробки нового проекту, зберігати результати оцінювання в базу даних та переглядати збережені результати оцінювання.

Опис прецедентів ПЗ для оцінювання тривалості розробки застосунків на мові C# для платформи PC наведено в таблиці 3.2.

Таблиця 3.2 – Опис прецедентів ПЗ для оцінювання тривалості розробки застосунків на мові C# для платформи PC

Прецедент	Короткий опис
1	2
Внесення параметрів моделі	Призначений для внесення параметрів розробленої моделі для виконання розрахунків

Продовження табл.3.2

1	2
Внесення довірчої ймовірності	Призначений для внесення значення довірчої ймовірності
Внесення інформації про застосунок	Призначений для внесення назви та трудомісткості розроблюваного застосунку для оцінювання його тривалості
Оцінювання тривалості	Призначений для оцінювання тривалості розробки в залежності від заданої трудомісткості, визначення довірчого інтервалу та інтервалу прогнозування для тривалості
Збереження результатів оцінювання	Призначений для збереження в базу даних результатів оцінювання
Перегляд збережених результатів оцінювання	Призначений для перегляду збережених в базу даних результатів оцінювання

Діаграма прецедентів ПЗ для оцінювання тривалості розробки застосунків на мові C# для платформи PC з виявленим актором, варіантами використання та зв’язками між ними наведена на рисунку 3.1. Актор (користувач) з варіантами використання поєднується зв’язками «асоціація».

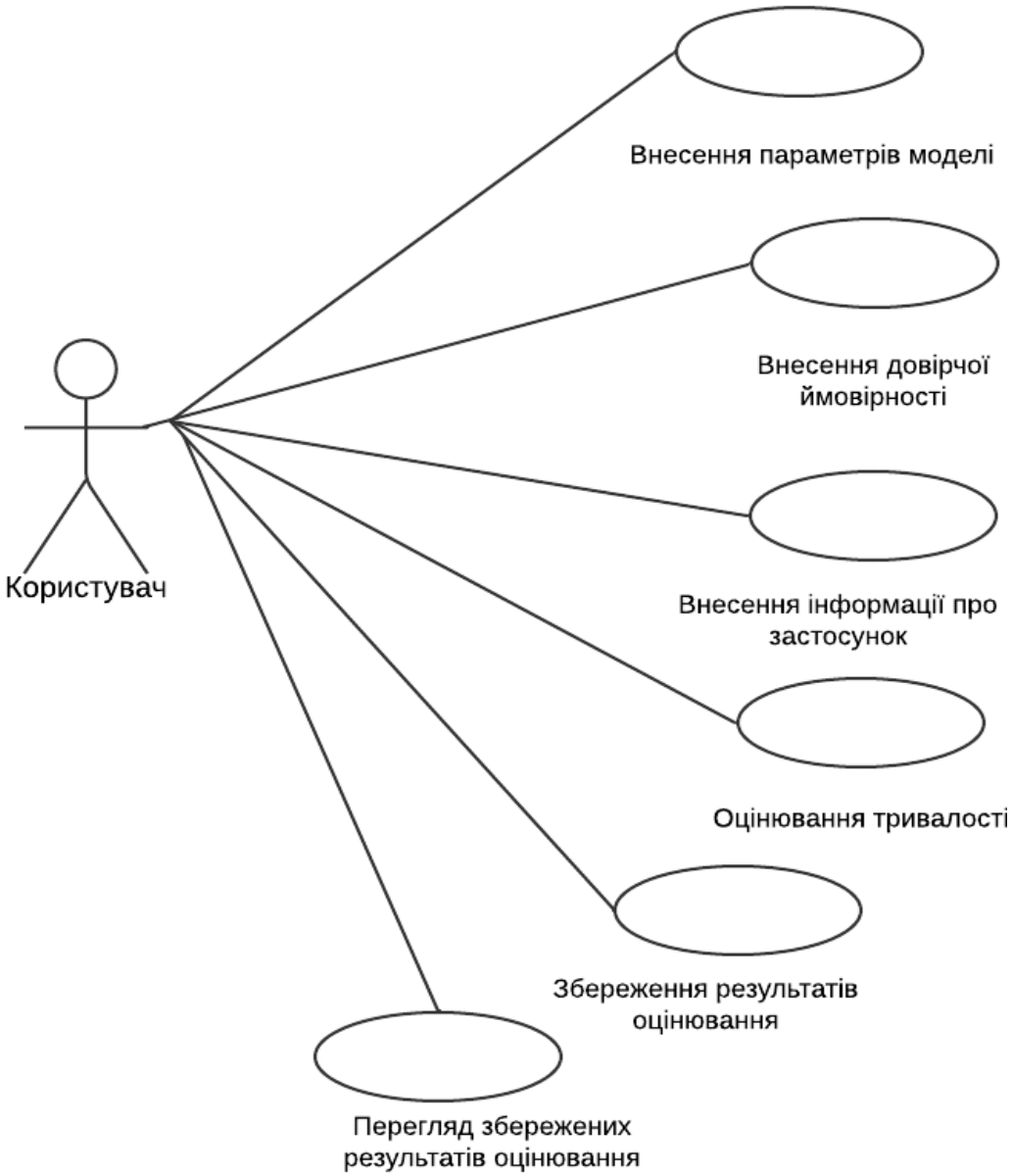

Рисунок 3.1 – Діаграма прецедентів ПЗ для оцінювання тривалості розробки застосунків на мові C# для платформи PC

Специфікації варіантів використання ПЗ для оцінювання тривалості розробки застосунків на мові C# для платформи PC

Для детального опису функціональності кожного варіанту використання системи розробляються специфікації варіантів використання.

– *Специфікація прецеденту «Внесення параметрів моделі»*

Назва: Внесення параметрів регресії.

Актор: Користувач.

Передумови: Відсутні.

Основний сценарій: Внесення параметрів моделі (N, b_0, b_1, AVGe, SSE, Se) у поля форми, перевірка параметрів на коректність, відображення параметрів у формі.

Альтернативний сценарій: Внесення параметрів моделі (N, b_0, b_1, AVGe, SSE, Se) у поля форми, перевірка параметрів на коректність. При внесенні некоректних даних з’явиться повідомлення про помилку.

Постумови: Відсутні.

Спеціальні вимоги: Відсутні.

– *Специфікація прецеденту «Внесення довірчої ймовірності».*

Назва: Внесення довірчої ймовірності.

Актор: Користувач.

Передумови: Відсутні.

Основний сценарій: Внесення значення довірчої ймовірності (Conf) у відповідне поле форми, перевірка значення на коректність, відображення значення у формі.

Альтернативний сценарій: Внесення значення довірчої ймовірності (Conf) у відповідне поле форми, перевірка значення на коректність. При внесенні некоректних даних з’явиться повідомлення про помилку.

Постумови: Відсутні.

Спеціальні вимоги: Значення довірчої ймовірності повинно бути додатнім числом, що не перевищує 1.

– *Специфікація прецеденту «Внесення інформації про застосунок»*

Назва: Внесення інформації про застосунок.

Актор: Користувач.

Передумови: Відсутні.

Основний сценарій: Внесення назви та трудомісткості розроблюваного застосунку для оцінювання його тривалості, перевірка даних на коректність, відображення даних у формі.

Альтернативний сценарій: Внесення назви та трудомісткості розроблюваного застосунку для оцінювання його тривалості, перевірка даних на коректність. При внесенні некоректних даних з’явиться повідомлення про помилку.

Постумови: Відсутні.

Спеціальні вимоги: Трудомісткість має бути додатнім числом.

– *Специфікація прецеденту «Оцінювання тривалості»*

Назва: Оцінювання тривалості.

Актор: Користувач.

Передумови: Наявність внесених даних для оцінювання.

Основний сценарій: Для початку оцінювання необхідно натиснути на кнопку «Виконати оцінювання». У формі з’являються результати оцінювання (оцінка тривалості, довірчий інтервал тривалості, інтервал передбачення тривалості).

Альтернативний сценарій: Відсутній.

Постумови: Відсутні.

Спеціальні вимоги: Відсутні.

– *Специфікація прецеденту «Збереження результатів оцінювання».*

Назва: Збереження результатів оцінювання.

Актор: Користувач.

Передумови: Наявність форми з результатами оцінювання.

Основний сценарій: Призначений для збереження результатів оцінювання в базу даних. Починається з натиснення кнопки «Зберегти оцінки». В базу зберігаються результати (назва C# застосунку, трудомісткість, оцінка

тривалості, довірчий інтервал тривалості, інтервал передбачення тривалості, дата розрахунку).

Альтернативний сценарій: Відсутній.

Постумови: Відсутні.

Спеціальні вимоги: Відсутні.

– Специфікація прецеденту «Перегляд збережених результатів оцінювання».

Назва: Перегляд збережених результатів оцінювання.

Актор: Користувач.

Передумови: Відсутні.

Основний сценарій: Призначений для перегляду збережених результатів оцінювання в базі. Починається з вибору пункту меню «Збережені результати». У вікні з’являється форма з інформацією про збережені результати (назва C# застосунку, трудомісткість, оцінка тривалості, довірчий інтервал тривалості, інтервал передбачення тривалості, дата розрахунку).

Альтернативний сценарій: Призначений для перегляду збережених результатів оцінювання в базі. Починається з вибору пункту меню «Збережені результати». При відсутності результатів оцінювання в базі даних, з‘явиться відповідне повідомлення.

Постумови: Відсутні

Спеціальні вимоги: Відсутні

Діаграми діяльності ПЗ для оцінювання тривалості розробки застосунків на мові C# для платформи PC

На рисунку 3.2 представлена діаграма діяльності для варіанту використання «Внесення інформації про застосунок»

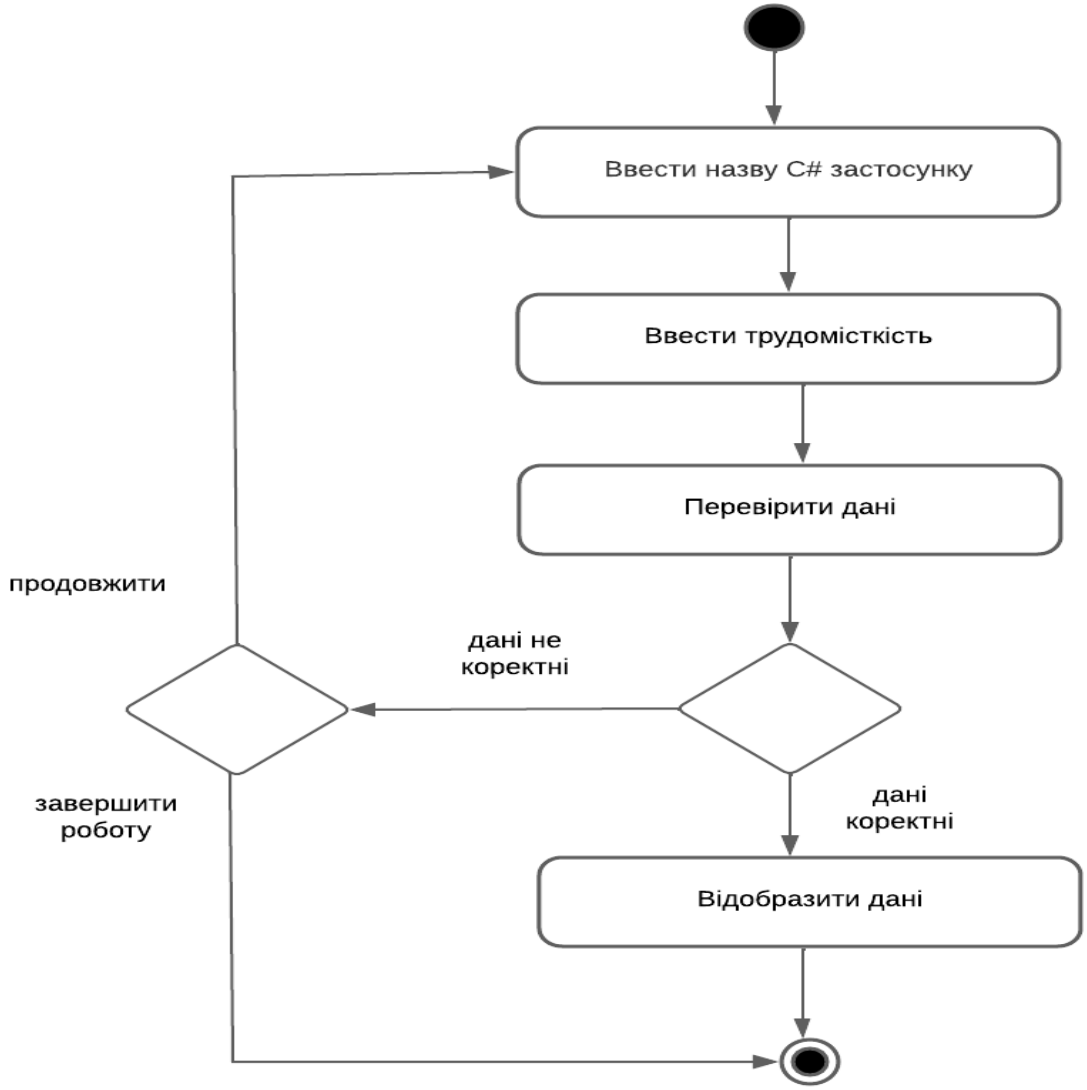

Рисунок 3.2 – Діаграма діяльності для варіанту використання «Внесення інформації про застосунок» ПЗ для оцінювання тривалості розробки застосунків на мові C# для платформи PC

Проект користувацького інтерфейсу ПЗ для оцінювання тривалості розробки застосунків на мові C# для платформи PC

Проект користувацького інтерфейсу програмного забезпечення призначений для розробки та візуалізації елементів взаємодії користувача з системою. Основною метою проекту користувацького інтерфейсу є забезпечення зручного, інтуїтивно зрозумілого та ефективного способу

взаємодії користувачів з програмою, що сприяє покращенню користувацького досвіду. Проект користувальницького інтерфейсу ПЗ для оцінювання тривалості розробки застосунків на мові C# для платформи PC представлений на рисунку 3.3.

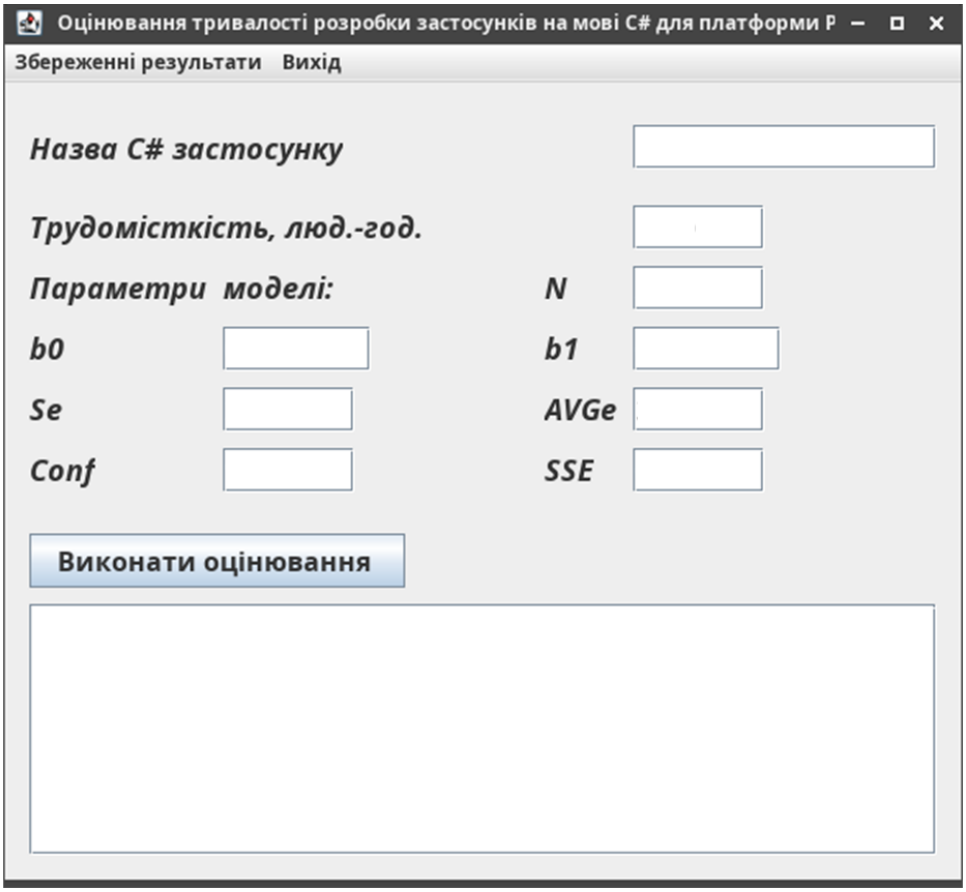

Рисунок 3.3 – Проект користувальницького інтерфейсу для ПЗ для оцінювання тривалості розробки застосунків на мові C# для платформи PC

3.3 Розробка технічного проекту програми для оцінювання тривалості розробки застосунків на мові C# для платформи PC

Технічний проект програмного забезпечення – це ключовий документ, що визначає всі аспекти розробки програмного продукту, включаючи технічні вимоги, алгоритми та інші компоненти. Він є основою для створення якісного

та ефективного програмного забезпечення, що відповідає вимогам замовника і забезпечує безперебійну роботу продукту на всіх етапах життєвого циклу. Деталізація всіх аспектів дозволяє знизити ризики під час реалізації та забезпечити успішну роботу програмного продукту.

Діаграма станів ПЗ для оцінювання тривалості розробки застосунків на мові C# для платформи PC

Діаграму станів показано на рисунку 3.4.

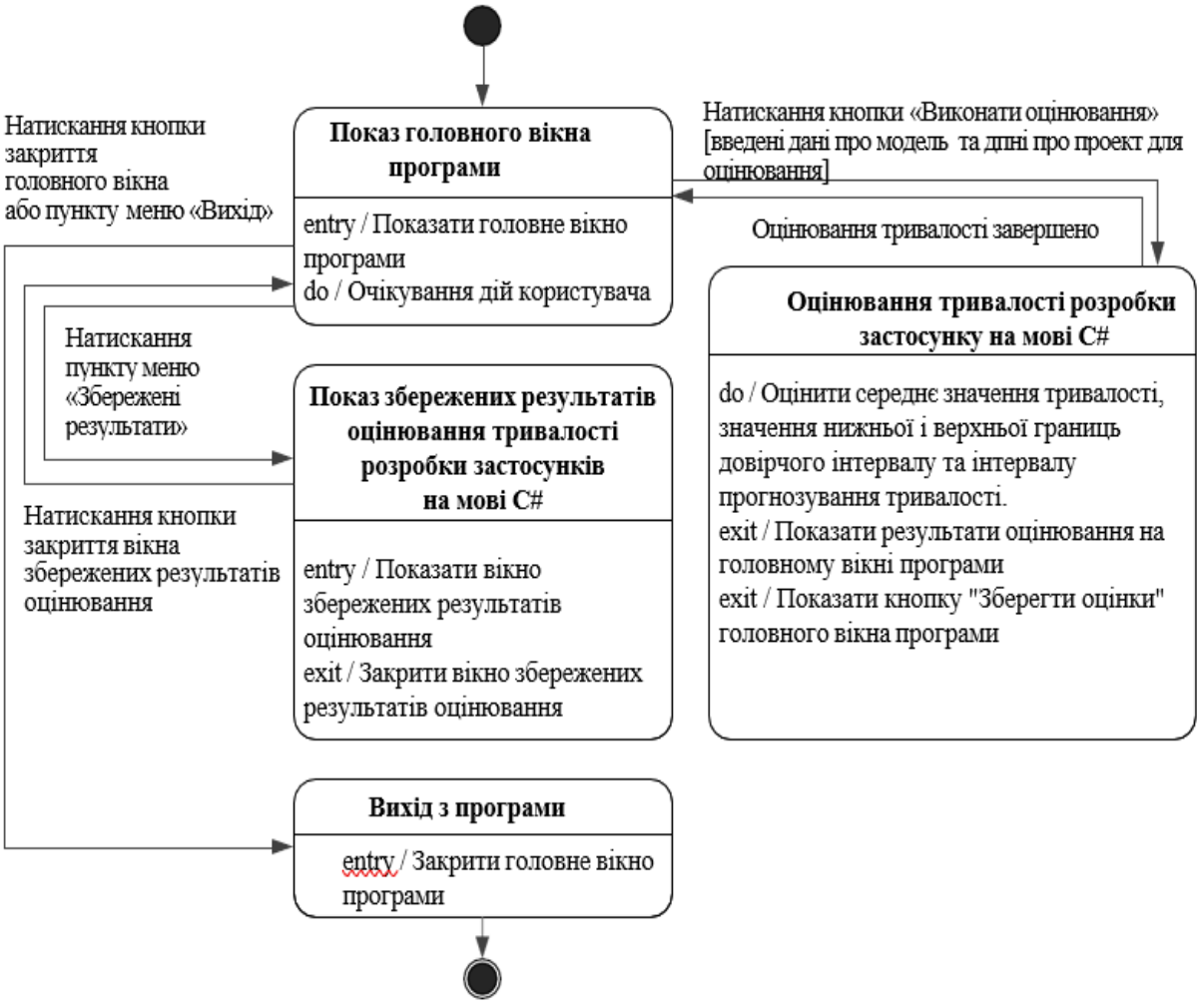

Рисунок 3.4 – Діаграма станів ПЗ для оцінювання тривалості розробки застосунків на мові C# для платформи PC

Логічна модель даних ПЗ для оцінювання тривалості розробки застосунків на мові C# для платформи PC

Дані з результатами оцінювання будуть зберігатися в базі даних, яка буде містити одну таблицю з наступною інформацією:

- номер по порядку, число, PK;
- назва C# застосунку, строка;
- трудомісткість, число
- оцінка тривалості, число,
- довірчий інтервал тривалості, число,
- інтервал передбачення тривалості, число,
- дата розрахунку, дата/час.

3.4 Розробка робочого проекту програми для оцінювання тривалості розробки застосунків на мові C# для платформи PC

Робочий проект програмного забезпечення є важливим етапом на шляху до створення успішного програмного продукту. Він містить всі конкретні деталі, необхідні для розробки та тестування ПЗ, і визначає всі етапи, що повинні бути виконані перед релізом програмного забезпечення. Правильно спланований робочий проект забезпечує не тільки ефективність розробки, але й високу якість кінцевого продукту, що відповідає вимогам замовника, і готовий до експлуатації.

Вибір засобів розробки програмного забезпечення

Java є популярним вибором для розробки програмного забезпечення. Ця мова програмування має наступні переваги:

– Платформонезалежність

Java має особливість, що дозволяє виконувати програми на будь-якій платформі, на якій доступна віртуальна машина Java (JVM).

– Масштабованість та продуктивність

Java надає можливості для розробки масштабованих додатків, що здатні обробляти великий обсяг даних і великі навантаження.

– Об'єктно-орієнтований підхід

Java є об'єктно-орієнтованою мовою, що дозволяє розробляти програмне забезпечення з чіткою структурою та повторно використовуваними компонентами. Це спрощує підтримку та розширення системи, а також полегшує командну розробку завдяки чіткому поділу на класи та об'єкти. ООП принципи, такі як інкапсуляція, наслідування та поліморфізм, забезпечують високу гнучкість і розширюваність коду.

– Велика екосистема бібліотек та фреймворків

Java має величезну кількість бібліотек, інструментів і фреймворків, які значно спрощують процес розробки.

– Безпека

Java пропонує численні механізми безпеки, такі як перевірка доступу, управління правами доступу та інші функції, які дозволяють створювати безпечні програми.

– Підтримка багатозадачності

Java має вбудовану підтримку багатозадачності, що дозволяє ефективно розробляти додатки, які повинні виконувати кілька завдань одночасно.

– Розвинена документація та підтримка спільноти

Java має величезну і активну спільноту розробників, що дозволяє швидко знаходити рішення для різних проблем.

– Підтримка для розробки мобільних додатків (Android)

Java є однією з основних мов програмування для розробки мобільних додатків для платформи Android, що робить її ідеальним вибором для розробки мобільних рішень.

– Простота використання та підтримка

Java є відносно простою мовою для вивчення, має чіткий синтаксис та стандартні бібліотеки, що дозволяє розробникам швидко освоїти її і ефективно використовувати для вирішення різних завдань.

Таким чином, вибір Java як мови програмування забезпечує високий рівень продуктивності, безпеки та гнучкості, що робить її відмінним варіантом для розробки надійних та масштабованих програмних рішень.

Саме тому в якості мови програмування була обрана Java.

Розробка та тестування ПЗ для оцінювання тривалості розробки застосунків на мові C# для платформи PC

Для розробки ПЗ для оцінювання тривалості розробки застосунків на мові C# для платформи PC використовувалася мова програмування Java. Текст програмного забезпечення наведено в Додатку Б.

Тестування програмного забезпечення виконано методом еквівалентного розбиття, який являється одним із найбільш вживаних методів функціонального тестування.

Метод еквівалентного розбиття є технікою тестування, яка використовується для ефективного створення тестових випадків. Основною ідеєю цього методу є розбиття простору вхідних значень на еквівалентні класи, для яких припускається, що система поводиться однаково. Тестування кожного класу дає змогу зменшити кількість тестових випадків, при цьому забезпечуючи покриття важливих варіантів поведінки програми.

Типи еквівалентних класів:

– Коректні значення: значення, які є припустимими для системи згідно з її специфікацією.

– Некоректні значення: значення, які повинні бути відхилені системою. Це можуть бути значення, що виходять за межі дозволеного діапазону, або значення, що не відповідають вимогам.

Всі функції програми були протестовані. Далі наводиться приклад тестування функції «Внесення інформації про застосунок». Класи еквівалентності вхідних даних для функції «Внесення інформації про застосунок» представлені в таблиці 3.3.

Таблиця 3.3 – Класи еквівалентності вхідних даних для функції «Внесення інформації про застосунок»

Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
Дані про застосунок	1 Всі дані про застосунок задані вірно	2 Відсутність одного чи декількох даних
		3 Одне чи більше даних задані невірно

Тести класів еквівалентності вхідних даних для функції «Внесення інформації про застосунок» представлені в таблиці 3.4.

Таблиця 3.4 – Тести для класів еквівалентності для функції «Внесення інформації про застосунок»

№	Вхідні дані	№ покритого класу	Вихідні дані
1	Всі дані про застосунок задані вірно	1	Дані про застосунок відобразилися у формі
2	Відсутність одного чи декількох даних	2	Повідомлення про помилку
3	Одне чи більше даних задано невірно	3	Повідомлення про помилку

По результатам тестування можна зробити висновок про те, що програма працює згідно вимог, сформульованих у технічному завданні.

4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВІД РОЗРОБКИ І ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ РОЗРОБКИ ЗАСТОСУНКІВ НА МОВІ C# ДЛЯ ПЛАТФОРМИ PC

4.1 Вступ

Дана робота присвячена підвищенню достовірності оцінювання тривалості розробки застосунків на мові C# для платформи PC та розробці програми для її реалізації. Розроблене програмне забезпечення можна використовувати в галузі розробки ПЗ, а також управління програмними проектами.

Створення програмного забезпечення вимагає одноразових витрат на його розробку, придбання необхідних технічних засобів та поточних витрат на функціонування продукту. Економія від функціонування ПЗ визначається з урахуванням витрат на його експлуатацію. Відношення цієї економії до витрат на створення ПЗ характеризує економічну ефективність капітальних вкладень. Економічні показники визначаються по діючим на момент розрахунку оптовим цінам, тарифам та ставкам заробітної плати.

4.2 Розрахунок витрат на створення й експлуатацію програмного забезпечення для оцінювання тривалості розробки застосунків на мові C# для платформи PC

Витрати на розробку ПЗ складаються з витрат на зарплату розробника, на амортизацію комп`ютера, на якому виконується розробка, на експлуатацію

цього комп`ютера, на засоби розробки та витратні матеріали і комплектуючі. Розробка ПЗ виконується програмістом, місячний оклад якого складає 30000 грн. Додаткова заробітна плата складає 20% від основної. Виходячи з цього, основна і додаткова заробітна плата розроблювача системи 36000 грн/міс, а вартість сучасного комп`ютера складає 40000 грн.

Вартість розробки ПЗ розраховується за наступною формулою:

$$С_{пр} = (З_{зп} + З_{сз} + З_{зг} + З_{е}) * Т + З_{м}, \quad (4.1)$$

де Т – тривалість розробки, міс;

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

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

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

$З_{е}$ – витрати на електроенергію;

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

$З_{е}$ при споживанні потужності 0,75 кВт, тривалості роботи на місяць, рівної 22*8 = 176 годин, вартості кіловат-години електроенергії 4,32 становить:

$$З_{е} = 176*4,32*0,75 = 570,24 \text{ грн.}$$

Витрати на допоміжні матеріали приведені в таблиці 4.1

Таблиця 4.1 – Витрати на допоміжні матеріали

Пункти витрат	Сума
Папір	200,00
Флеш-накопичувач	250,00
Непередбачувані витрати	500,00
Картридж для принтера	600,00
Разом	1052,00

Витрати на розробку ПЗ представлені в табл. 4.2.

Таблиця 4.2 – Витрати на розробку ПЗ

№	Найменування витрат	Одиниця	Кількість
1	Тривалість розробки (Т)	Міс.	6
2	Основна і додаткова заробітна плата ($З_{зп}$)	Грн.	36000,00
3	Відрахування на соціальні заходи ($З_{сз}$)	Грн.	13680,00
4	Загальногосподарські витрати ($З_{зг}$)	Грн.	3000,00
5	Витрати на допоміжні матеріали ($З_{м}$)	Грн.	1550,00
6	Витрати на електроенергію ($З_{е}$)	Грн.	570,24

За формулою (4.1) розрахуємо вартість програми:

$$С_{пр} = (36000,00+13680,00+3000,00+570,24)*6+1550 = 321051,44 \text{ грн.}$$

Амортизаційні відрахування складають 60% балансної вартості на рік:

$$А_{об} = 40000,00*0,6 = 24000,0 \text{ грн.}$$

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

$$В_{м} = 40000,00*0,05 = 2000,00 \text{ грн.}$$

Річний обсяг робіт комп’ютера у годинах визначається наступним способом:

$$Ф_{м} = 264,5*Т_{З} \quad (4.2)$$

де $Т_{З}$ – середнє місячне завантаження устаткування (близько 4 годин);

264,5 – середня кількість робочих днів у році.

Отже, річний обсяг роботи комп’ютера розрахований за формулою (4.2) складає:

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

Витрати на електроенергію $З_е$ при 1058 годинах роботи устаткування в рік складуть:

$$З_е = 1058*4,32*0,52 = 924,27 \text{ грн.}$$

Експлуатаційні витрати для комп’ютера за рік складуть:

$$З_{зр} = 20999,4 + 1749,95 + 924,27 = 28376,7 \text{ грн.}$$

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

$$З_{се} = 321051,44 + 28376,7 = 349428,14 \text{ грн.}$$

4.3 Економічна ефективність розробки та впровадження програмного забезпечення

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

Крім багатьох інших негативних ефектів, ручна обробка інформації спричиняє наступні негативні економічні ефекти: високі витрати на складування паперових документів (сейфи, шафи, папки й ін.); підвищені витрати на канцтовари; витрати, пов'язані з роботою виявлення раніше допущених помилок. До числа основних факторів, що визначають приріст прибутку в зв'язку з упровадженням програмного забезпечення, відносяться: підвищення продуктивності праці; вивільнення робочого часу.

Крім того, не піддається прямій грошовій оцінці підвищення оперативності керування, якості одержуваних результатів, поліпшення організації праці і т.д. Обов'язковою умовою визначення економічної ефективності програмного забезпечення є порівнянність усіх показників у часі, за цінами й іншими нормами, використовування для визначення показників, за змістом і колом елементів витрат.

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

Зарплата 0,6 працівника в рік складає:

(36000,00*12)*0,6 = 259200,00 грн.

Річний економічний ефект розраховується за наступною формулою:

$$\mathrm{E}_{\text{год}} = \Delta\mathrm{C}_n - E_n * k, \quad (4.3)$$

де $\Delta\mathrm{C}_n$– вивільнені кошти після впровадження ПЗ (259200,00 грн) мінус експлуатаційні витрати (28376,7) = 230823,3 грн.;

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

k – одноразові витрати на впровадження продукту (349428,14).

$$\mathrm{E}_{\text{год}} = 230823{,}3 - 349428{,}14 * 0{,}6 = 21166{,}41 \text{ грн.}$$

Строк окупності системи розраховується по формулі:

$$\mathrm{T} = k/\Delta\mathrm{C}_n = 349428{,}14/230823{,}3 \approx 1{,}51 \text{ (року)}$$

Отже, строк окупності ПЗ складає приблизно 18 місяців.

5 ОХОРОНА ПРАЦІ

Охорона праці – система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів і засобів, направлених на збереження і працездатності людини в процесі трудової діяльності.

Дана система включає в себе:

– аналіз шкідливих та небезпечних факторів, які впливають на людину;

– розробка заходів щодо усунення шкідливого впливу на людину та створення нормальних умов праці;

– правила з техніки безпеки та виробничої санітарії;

– норми з охорони праці жінок, неповнолітніх та осіб з низькою працездатністю;

– спеціальні норми охорони праці осіб, які працюють в тяжких, шкідливих та небезпечних виробничих умовах.

Безпека людини на виробництві залежить від багатьох факторів. Здебільшого вона визначається кваліфікацією та відповідальністю робітників управління.

Велике значення для підвищення безпеки праці у виробництві має система стандартів безпеки праці, яка включає значну кількість взаємопов’язаних стандартів, розроблених для попередження ушкоджень і захворювань працюючих.

Закон України “Про охорону праці” визначає положення щодо реалізації права на охорону їх життя і здоров’я в процесі трудової діяльності, регулює за участю відповідних органів відносини між власником підприємства, установи і організації і робітником з питань безпеки, гігієни праці і виробничої санітарії, встановлює єдиний порядок організації охорони праці на Україні [20].

Поліпшення умов праці, підвищення їх безпеки нешкідливості має велике економічне значення. Воно впливає на економічні результати виробництва, на продуктивність праці, якість і собівартість продукції, що виготовляється.

Підвищення умов праці призводить до таких соціальних результатів, як покращення здоров’я працюючих, ріст ступеня задоволення працею, зміцнення трудової дисципліни, підвищення престижу ряду професій, ріст виробничої активності і поліпшення ряду інших показників, які характеризують більш високий ступінь розвитку працюючих.

Охорона природи, оскільки очищення і знешкодження стічних вод і викидів в атмосферу, боротьба із шумом і вібрацією, захист від електромагнітних полів та іонізуючих випромінювань служать не тільки цілям охорони праці, але й одночасно сприяють охороні середовища мешкання людини.

5.1 Аналіз небезпечних і шкідливих факторів у відділі розробки програмного забезпечення

При роботі співробітники відділу зазнають впливу небезпечних і шкідливих факторів, як-то:

– недостатня освітленість робочого місця;

– підвищений рівень шуму і вібрації від роботи вентиляторів, принтерів;

– висока ймовірність виникнення пожеж в зв’язку з наявністю великої кількості паперу;

– підвищена напруга в електричній мережі, замикання якої може відбутися через тіло людини;

– вплив неіонізуючих електромагнітних випромінювань, електростатичних і магнітних полів, які виникають при роботі комп’ютерів;

– недостатній рівень параметрів мікроклімату і температури, відносної вологості повітря, швидкості руху повітря.

Робота співробітника вимагає від нього значної зорової уваги, і як наслідок, у приміщенні повинно бути правильно спроектоване і розташоване освітлення.

В залежності від часу доби і джерела світла на підприємстві використовується освітлення трьох видів: природне, штучне і комбіноване.

У денний час і ясну погоду використовується природне освітлення через світлові віконні пройми. Для нього характерна висока розсіяність і це є сприятливим фактором для здорової діяльності.

У передвечірній час і за несприятливих погодних умовах використовується комбіноване освітлення. Оскільки денне світло не забезпечує достатню освітленість робочого місця, то використовуються настільні лампи місцевого освітлення.

У вечірній час взимку використовується штучне освітлення. В якості джерел використовуються лампи накалювання.

Рекомендована і дійсна освітленість у приміщенні відділу товарно-матеріальних цінностей складає 750лк.

Одним із найшкідливіших факторів, що чинить негативний вплив на працездатність робітників, є шум. Джерелами шуму у відділі є вентилятори, ПК.

Для забезпечення нормативного рівня шуму у виробничому приміщенні відділу і на робочих місцях використовуються шумопоглинаючі засоби: важкогорючі спеціальні перфоровані плити, мінеральна вата з максимальним коефіцієнтом звукопоглинання в межах частот 31,5-8000Гц.

Рівень вібрації при виконанні робіт з використанням ПК не повинен перевищувати допустимих значень. Оскільки в бюро використовуються 2 ПК з одним струйним принтером, то рівень вібрації, який створюють один принтер і два процесори, знаходиться в межах допустимих норм.

У відділі є меблі, які являються швидкозагораючим предметом. Існує ризик, у випадку виникнення пожежі, швидкого її поширення по приміщенню.

Електричний струм, проходячи через тіло людини справляє тепловий, хімічний, біологічний вплив, порушуючи нормальну життєдіяльність організму. Будь-який з цих впливів може привести до електротравм.

У відділі є холодильник, електрочайник, кондиціонер. Електровмикання здійснюється через трьохштирьові штепсельні вилки, які заземлені на штатну систему заземлення.

Робота комп'ютера супроводжується електромагнітним і електростатичним випромінюванням. Навколо електростатично-зарядженого монітору підвищується концентрація пилу. Такий електризований пил може викликати запалення шкіри, а електромагнітне випромінювання може призвести до зниження загальної продуктивності співробітника.

Мікрокліматичні параметри (температура, відносна вологість повітря, швидкість руху повітря) впливають у першу чергу на продуктивність праці і самопочуття людини, а також на надійність роботи обчислювальної техніки. Забезпечення достатньої роботи системи вентиляції і опалення також підтримують працездатність співробітника.

Приміщення відділу обладнане системою опалення, кондиціонування повітря. Параметри мікроклімату, які діють у відділу, відповідають встановленим нормам.

5.2 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. 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) 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 1м, між собою – на відстані не менше 1,5 м. Відносно вікон робочі місця доцільно розташувати таким чином, щоб природне світло падало на нього збоку, переважно зліва.

4) 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 антивідблискові 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.

5) 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.

6) 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.

7) 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.

8) 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.

9) 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 вимкнутому 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).

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

Вимоги безпеки при закінчення 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. 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 Аналіз існуючої проблеми необхідності охорони навколишнього середовища

Охорона навколишнього природного середовища, раціональне використання природних ресурсів, забезпечення екологічної безпеки життєдіяльності людини – невід'ємна умова сталого економічного та соціального розвитку України.

З цією метою Україна здійснює на своїй території екологічну політику, спрямовану на збереження безпечного для існування живої і неживої природи навколишнього середовища, захисту життя і здоров'я населення від негативного впливу, зумовленого забрудненням навколишнього природного середовища, досягнення гармонійної взаємодії суспільства і природи, охорону, раціональне використання і відтворення природних ресурсів.

Закон про охорону навколишнього середовища визначає правові, економічні та соціальні основи організації охорони навколишнього природного середовища в інтересах нинішнього і майбутніх поколінь.

Завданням законодавства про охорону навколишнього природного середовища є регулювання відносин у галузі охорони, використання і відтворення природних ресурсів, забезпечення екологічної безпеки, запобігання і ліквідації негативного впливу господарської та іншої діяльності на навколишнє природне середовище, збереження природних ресурсів, генетичного фонду живої природи, ландшафтів та інших природних комплексів, унікальних територій та природних об'єктів, пов'язаних з історико-культурною спадщиною [21].

Основними принципами охорони навколишнього природного середовища є:

а) пріоритетність вимог екологічної безпеки, обов'язковість додержання екологічних стандартів, нормативів та лімітів використання природних ресурсів при здійсненні господарської, управлінської та іншої діяльності;

б) гарантування екологічно безпечного середовища для життя і здоров'я людей;

в) запобіжний характер заходів щодо охорони навколишнього природного середовища;

г) екологізація матеріального виробництва на основі комплексності рішень у питаннях охорони навколишнього природного середовища, використання та відтворення відновлюваних природних ресурсів, широкого впровадження новітніх технологій;

д) збереження просторової та видової різноманітності і цілісності природних об'єктів і комплексів;

е) науково обгрунтоване узгодження екологічних, економічних та соціальних інтересів суспільства на основі поєднання міждисциплінарних знань екологічних, соціальних, природничих і технічних наук та прогнозування стану навколишнього природного середовища;

є) обов'язковість надання висновків державної екологічної експертизи;

ж) гласність і демократизм при прийнятті рішень, реалізація яких впливає на стан навколишнього природного середовища, формування у населення екологічного світогляду;

з) науково обгрунтоване нормування впливу господарської та іншої діяльності на навколишнє природне середовище;

и) безоплатність загального та платність спеціального використання природних ресурсів для господарської діяльності;

і) компенсація шкоди, заподіяної порушенням законодавства про охорону навколишнього природного середовища;

ї) вирішення питань охорони навколишнього природного середовища та використання природних ресурсів з урахуванням ступеня антропогенної зміненості територій, сукупної дії факторів, що негативно впливають на екологічну обстановку;

й) поєднання заходів стимулювання і відповідальності у справі охорони навколишнього природного середовища;

к) вирішення проблем охорони навколишнього природного середовища на основі широкого міждержавного співробітництва;

л) встановлення екологічного податку, збору за спеціальне використання води, збору за спеціальне використання лісових ресурсів, плати за користування надрами відповідно до Податкового кодексу України.

6.2 Забруднення навколишнього середовища фірмою з розробки програмного забезпечення

Основними чинниками забруднення навколишнього середовища фірмою є лише використання автотранспорту. Бо підприємство окрім забруднення повітряного середовища не забруднює ні водне, ні ґрунтове.

В сучасних умовах автомобільний транспорт стає найбільш значним джерелом забруднення атмосферного повітря, особливо великих міст.

Відомо, що атмосферне повітря міст постійно забруднюється і за всіма параметрами докорінно відрізняється від повноцінного природного повітря. Міські поселення характеризуються найвищими рівнями антропогенних навантажень на навколишнє середовище, в результаті чого воно деформується, набуває якісно нових рис, аж до зміни мікрокліматичних факторів і фізико-хімічних властивостей середовища, зокрема повітряного басейну.

Метеорологічні спостереження свідчать, що температура повітря в межах міських територій у середньому на декілька градусів вища, ніж у приміських районах. Над містами, особливо великими, частіше випадають атмосферні опади, бувають густі тумани, а також смоги – густі тумани, змішані з димом, кіптявою та вихлопними газами. Прозорість атмосфери в містах менша, ніж за її межами і в сільській місцевості. Тумани, а також запиленість повітря помітно зменшують проникнення до земної поверхні сонячних променів.

Однак за рахунок автомобільних викидів якість атмосферного повітря в містах погіршилась.

Прогресуючому забрудненню атмосфери в містах сприяє висока питома вага автомобілів, адже зростання кількості автотранспортних засобів супроводжується збільшенням об'ємів забруднюючих речовин із вихлопних труб. Останнім часом у міському повітрі виріс об'єм оксидів вуглецю, вуглеводнів, оксидів азоту, сажі. Але найбільшу небезпеку, окрім оксидів азоту, складають сірчані та свинцеві сполуки. Їх вміст у міському повітрі значною мірою виріс. Місто не пристосоване до такої кількості автотранспорту. Довжина пробігу без зупинок між світлофорами становить лише 400–600 м., внаслідок цього середня швидкість руху вдень у центрі міста і на великих автошляхах знижується до 12–20 км/год, а це збільшує витрати палива в 3–4 рази. Відповідно збільшуються й викиди забруднюючих речовин. Автотранспорт також призводить до специфічних форм забруднення повітря. При русі стираються шини і тисячі тон гуми у вигляді пилу потрапляє в повітря. Автомобільний транспорт не тільки забруднює повітря продуктами згорання палива, він сприяє зростаючому надходженню свинцю в навколишнє середовище. В Україні поки ще використовують бензин із вмістом свинцю 0,36 г/л, тоді як в Англії, Німеччині та США – 0,013–0,15.

6.3 Розробка заходів щодо зменшення забруднення

Збір відходів часто є найбільш дорогим компонентом усього процесу утилізації. Тому правильна організація збору відходів може заощадити значну кількість грошових витрат. Існуюча в Україні система збору ТПВ повинна залишатися стандартизованою з погляду економічності. У той же час додаткове планування необхідно для того, щоб вирішити нові проблеми (наприклад, відходи комерційних кіосків, на збір яких часто не вистачає ресурсів).

У густонаселених територіях нерідко доводиться транспортувати відходи на великі відстані. Рішенням у цьому випадку може бути станція тимчасового зберігання відходів, від якої сміття може вивозитися великими по вантажопідйомності машинами або по залізниці. При цьому треба відзначити, що станції проміжного зберігання являють собою об'єкти підвищеної екологічної небезпеки й можуть при неправильному розташуванні й експлуатації викликати не менше дорікань місцевих жителів і громадських організацій, ніж смітники.

Вторинна переробка

Досить багато компонентів ТПВ можуть бути перероблені в корисні продукти. Скло звичайно перероблюють шляхом здрібнювання й переплавлення (бажано, щоб вихідне скло було одного кольору). Скляний бій низької якості після здрібнювання використовується як наповнювач для будівельних матеріалів (наприклад, т.зв. «глассфальт»). У багатьох містах існують підприємства по відмиванню й повторному використанню скляного посуду. Така ж, безумовно, позитивна практика існує, наприклад, у Данії.

Сталеві й алюмінієві банки переплавляються з метою одержання відповідного металу. При цьому виплавка алюмінію з баночок для прохолодних напоїв вимагає тільки 5% від енергії, необхідної для виготовлення тієї ж кількості алюмінію з руди, і є одним з найбільш вигідних видів «ресайклінга».

Паперові відходи різного типу вже багато десятків років застосовують поряд зі звичайною целюлозою для виготовлення пульпи – сировини для паперу. Зі змішаних або низькоякісних паперових відходів можна виготовляти туалетний або обгортковий папір і картон. На жаль, в Україні тільки в невеликих масштабах присутня технологія виробництва високоякісного паперу з високоякісних відходів (обрізків друкарень, використаного паперу для ксероксів і лазерних принтерів і т.п.). Паперові відходи можуть також використовуватися в будівництві для виробництва теплоізоляційних матеріалів і в сільському господарстві – замість соломи на фермах.

Пластик. Переробка пластику досить дорогий та складний процес. З деяких видів пластику можна одержувати високоякісний пластик тих же властивостей, інші види пластику (наприклад ПВХ) після переробки можуть бути використані тільки як будівельні матеріали.

Компостування

Компостування – це технологія переробки відходів, заснована на їхньому природному біорозкладанні. Найбільш широко компостування застосовується для переробки відходів органічного, насамперед рослинного походження, таких як листи, вітки й скошена трава. Існують технології компостування харчових відходів та ТПВ.

У Україні компостування за допомогою компостних ям часто застосовується населенням в індивідуальних будинках або на садових ділянках. У той же час процес компостування може бути централізований і проводитися на спеціальних площадках. Існує кілька технологій компостування, що розрізняються за вартістю й складністю. Більше прості й дешеві технології вимагають більше місця й процес компостування займає більше часу.

Кінцевим продуктом компостування є компост, що може знайти різні застосування в міському й сільському господарстві.

Компостування, застосовуване в Україні на механізованих сміттєпереробних заводах, наприклад, у Львові, представляє собою процес зброджування в біореакторах усього обсягу ТПВ, а не тільки його органічної складової. Хоча характеристики кінцевого продукту можуть бути значно поліпшені шляхом добування з відходів металу, пластику й т.п., все ж таки він представляє собою досить небезпечний продукт і знаходить дуже обмежене застосування (на Заході такий «компост» застосовують тільки для покриття смітників).

Таким чином, проблема побутових відходів є досить важливою для нашого суспільства. Існують розроблені методи для високотехнологічної утилізації сміття, але для того щоб вони були ефективними, треба, щоб кожна людина перейнялася відповідальністю за екологічний стан навколишнього середовища.

ВИСНОВКИ

Мета кваліфікаційної роботи, яка полягала в підвищенні достовірності оцінювання тривалості розробки застосунків на мові C# для платформи PC, була повністю досягнута.

Під час виконання кваліфікаційної роботи була отримана така наукова новизна: було удосконалено нелінійну регресійну модель для оцінювання тривалості розробки застосунків на мові C# для платформи PC за рахунок застосування нормалізуючого перетворення десяткового логарифму і викидів регресії, що дозволило підвищити достовірність оцінювання тривалості розробки застосунків на мові C# для платформи PC в порівнянні з існуючими моделями COCOMO та ISBSG.

Для досягнення мети було вирішено такі завдання:

– проаналізувано основні метрики та існуючі моделі, які використовуються для оцінювання тривалості розробки програмних застосунків;

– обґрунтувано необхідність подальшого дослідження за обраною темою;

– проаналізувано математичну модель для оцінювання тривалості розробки програмних застосунків;

– побудувано нелінійну регресійну модель для оцінювання тривалості розробки застосунків на мові C# для платформи PC;

– розроблено проект програми для оцінювання тривалості розробки застосунків на мові C# для платформи PC.

Також було виконано розрахунок економічної ефективності від розробки та впровадження програми для оцінювання, та розглянуто питання з охорони праці і охорони навколишнього середовища.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Приходько С.Б. Метод побудови нелінійних рівнянь регресії на основі нормалізуючих перетворень. Тези доповідей міждерж. наук.-методич. конф. “Проблеми математичного моделювання” (13-15 червня 2012, Дніпродзержинськ). Дніпродзержинськ: ДДТУ. С. 31-33.

2. Oligny S., Bourque P., Abran A., Fournier, B. Exploring the relation between effort and duration in software engineering projects. Proc. of the 16th IFIP World Computer Congress (Aug, 2000, Beijing, China). P. 175-178.

3. Приходько С.Б., Пухалевич А.В., Халанчук І.П. Нелінійні регресійні моделі для оцінювання тривалості розробки програмних застосунків, що створюються мовами Java та Visual Basic для ПК. Інформаційні технології: моделі, алгоритми, системи (ITMAS – 2021): Матеріали ІІ Всеукраїнської науково-практичної інтернет конференції (26-28 жовтня 2021 р., м. Миколаїв). Миколаїв: НУК. С. 6-7.

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

5. Макарова Л.М., Литвин М.В. Математична модель для оцінювання тривалості розробки інтернет-магазинів, створених за допомогою платформи Shopify. *Матеріали XIII-ої Міжнародної науково-практичної конференції «Free and Open Source Software»*. (Харків, 16-18 листопада 2021 р.). Харків: Харківський національний університет будівництва та архітектури, 2021. С.22-23.

6. Пухалевич А.В., Кольцов Є.В. Нелінійна регресійна модель для оцінювання тривалості розробки застосунків на мові C# для платформи PC. *Матеріали V Всеукраїнської науково-практичної інтернет конференції*

“Інформаційні технології: моделі, алгоритми, системи (ITMAS – 2024)” (30-31 жовтня 2024 р., м. Миколаїв). Миколаїв, НУК імені адмірала Макарова, 2024. С. 73–75.

7. Software Measurement and Metrics. URL: https://www.geeksforgeeks.org/software-measurement-and-metrics/ (дата звернення: 14.09.2024).

8. Lines of Code (LOC) in Software Engineering. URL: https://www.geeksforgeeks.org/lines-of-code-loc-in-software-engineering/ (дата звернення: 14.09.2024).

9. Функціональні точки (Function Points - FP). URL: https://www.maxzosim.com/funktsionalni-tochki/ (дата звернення: 14.09.2024).

10. Software Metrics. URL: https://www.javatpoint.com/software-engineering-software-metrics (дата звернення: 14.09.2024).

11. Метод Дельфі (Delphi method). URL: https://www.maxzosim.com/delphi-method/ (дата звернення: 14.09.2024).

12. Boehm B.W. Software engineering economics. Englewood Cliffs, NJ: Prentice Hall, 1981. 768 p.

13. International Software Benchmarking Standards Group. URL: https://www.isbsg.org/ (дата звернення: 14.09.2024).

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

15. Приходько С.Б., Макарова Л.М., Приходько Н.В., Пухалевич А.В. Методичні вказівки та завдання до виконання лабораторних робіт з дисципліни "Емпіричні методи програмної інженерії". Миколаїв: НУК, 2023. 56 с.

16. Multivariate Normality Testing (Mardia). URL: https://real-statistics.com/multivariate-statistics/multivariate-normal-distribution/multivariate-normality-testing/ (дата звернення: 05.10.2024).

17. Mardia K.V. Measures of multivariate skewness and kurtosis with applications. *Biometrika*. 57. 1970. P. 519–530. DOI: 10.1093/biomet/57.3.519

18. Prykhodko, S., Prykhodko, N., Makarova, L., Pukhalevych, A. Application of the Squared Mahalanobis Distance for Detecting Outliers in Multivariate NonGaussian Data. *Proceedings of 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)* (Lviv-Slavske, Ukraine, February 20-24, 2018). P.962-965. DOI: 10.1109/TCSET. 2018.8336353

19. Prykhodko S., Prykhodko N., Makarova L., Pukhalevych A. Outlier Detection in NonLinear Regression Analysis Based on the Normalizing Transformations. *Proceedings of the 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*, IEEE. (Lviv-Slavske, 2020). P.407-410.

20. Про охорону праці: Закон України від 15.11.2024 № 2694-XII. URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text (дата звернення: 10.11.2024).

21. Про охорону навколишнього природного середовища: Закон України від 15.11.2024 № 1264-XII. URL: https://zakon.rada.gov.ua/laws/show/1264-12#Text (дата звернення: 11.11.2024).

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ РОЗРОБКИ ЗАСТОСУНКІВ НА МОВІ C# ДЛЯ ПЛАТФОРМИ PC

Вступ

Повна назва розроблюваного проекту: «Програмне забезпечення для оцінювання тривалості розробки застосунків на мові C# для платформи PC».

Коротка назва: «Estimation».

Галузь застосування: Розробка ПЗ, управління програмними проектами.

1 Підстави для розробки

Підставою для розробки ПЗ є наказ ректора на затвердження тем кваліфікаційних (магістерських) робіт зі спеціальності 121 «Інженерія програмного забезпечення» в Національному університеті кораблебудування ім. адмірала Макарова.

2 Призначення розробки

2.1 Функціональне призначення

Дане програмне забезпечення призначене для оцінювання тривалості розробки застосунків на мові C# для платформи PC.

2.2 Експлуатаційне призначення

Дане програмне забезпечення призначене для автоматизації та скорочення часу відповідних розрахунків.

3 Вимоги до програмного забезпечення

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу виконуваних функцій

Програмне забезпечення повинно мати можливість виконання наступних функцій:

– Програмне забезпечення повинно мати можливість виконання наступних функцій:

– Внесення параметрів моделі (внесення параметрів N, b0, b1, AVGe, SSE, Se розробленої моделі).

– Внесення довірчої ймовірності (внесення Conf).

– Внесення інформації про застосунок (назви та трудомісткості застосунку, що розробляється).

– Оцінювання тривалості розробки застосунку.

– Збереження результатів оцінювання в базу даних.

– Перегляд збережених в базі даних результатів оцінювання.

3.1.1 Вимоги до організації вхідних та вихідних даних

Вимоги до організації вхідних даних

Вхідні дані заносяться в відповідні поля форми відповідно до типів даних.

Вхідними даними є параметри моделі (параметри N, b0, b1, AVGe, SSE, Se розробленої моделі), значення довірчої ймовірності (Conf), інформація про застосунок (назва та трудомісткість застосунку, що розробляється).

Вимоги до організації вихідних даних

Вихідні дані виводяться в відповідні місця екранних форм та зберігаються в базі даних.

Вихідними даними є назва застосунку, трудомісткість застосунку, розрахована тривалість розробки застосунку, довірчий інтервал тривалості, інтервал передбачення тривалості.

3.2 Вимоги до надійності

3.2.1 Вимоги до забезпечення надійності функціонування програмного забезпечення

Програмне забезпечення має функціонувати неперервно. У випадку виникнення помилок у роботі комп'ютера, відновлення нормального функціонування повинно відбуватися після перезавантаження програмного продукту.

3.2.2 Час відновлення після відмови

Вимоги до часу відновлення після відмови не пред’являються.

3.2.3 Вимоги до контролю вхідної та вихідної інформації

У випадку введення користувачем неправильних даних, програмне забезпечення повинно повідомити про помилку і надати можливість її виправити.

3.3 Вимоги до складу та параметрів технічних засобів

Для функціонування ПЗ рекомендується наступний склад технічних засобів з мінімальними характеристиками:

– ПК з процесором Core i3;

– Оперативна пам'ять –1 Гб;

– Обсяг на жорсткому диску –500 Мб.

3.4 Вимоги до інформаційної та програмної сумісності

Операційна система Windows 7 чи вище, або інша, в якій встановлена версія JRE не нижче версії Java 1.8.

3.5 Вимоги до захисту інформації та програм

Вимоги до захисту інформації та програм не висуваються.

3.6 Вимоги до транспортування та пакування

Вимоги до транспортування та пакування не висуваються.

4 Вимоги до програмної документації

До складу програмної документації входять:

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

5 Техніко-економічні показники

Економічна ефективність впровадження програмного забезпечення розрахована в розділі 4. Згідно з отриманими результатами впровадження даного ПЗ є доцільним.

6 Стадії та етапи розробки програмного забезпечення

Стадії та етапи розробки програмного забезпечення для оцінювання тривалості розробки застосунків на мові C# для платформи PC представлені в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки програмного забезпечення

Стадії розробки	Етапи розробки	Термін виконання
1. Технічне завдання	Розробка та затвердження технічного завдання	21.09.2024
2. Ескізний проект	Розробка ескізного проекту	18.10.2024
	Затвердження ескізного проекту	21.10.2024
3. Технічний проект	Розробка технічного проекту	03.11.2024
	Затвердження технічного проекту	06.11.2024
4. Робочий проект	Розробка програми	16.11.2024
	Розробка програмної документації	26.11.2024
	Випробування програми	01.12.2024

7 Порядок контролю і приймання

Порядок контролю та приймання програми здійснюється представниками замовника у присутності представника розробника. Для приймання програми надається програмна документація, яка була розроблена відповідно до технічного завдання. Представники замовника установлюють відповідність програми технічному завданню. За результатами приймання програми представниками замовника складається акт приймання програми.

ДОДАТОК Б – КОД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ РОЗРОБКИ ЗАСТОСУНКІВ НА МОВІ C# ДЛЯ ПЛАТФОРМИ PC

database.sql

```
DROP TABLE IF EXISTS estimations;
CREATE TABLE estimations (
        id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        app_name TEXT NOT NULL UNIQUE,
        effort  NUMERIC NOT NULL,
        time NUMERIC NOT NULL,
        time_confidence_interval_bottom NUMERIC NOT NULL,
        time_confidence_interval_top NUMERIC NOT NULL,
        time_estimation_interval_bottom NUMERIC NOT NULL,
        time_estimation_interval_top NUMERIC NOT NULL,
        date DATE NOT NULL
);
```

EstimationApplication.java

```
package CSharpDevEstimate;

import CSharpDevEstimate.forms.EstimationForm;
import CSharpDevEstimate.forms.StoredResultsForm;
import oracle.toplink.essentials.config.TopLinkProperties;

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

public class EstimationApplication {
   // Main application frame for estimations
   static JFrame mainApplicationFrame;
   // Form for entering estimation details
   static EstimationForm estimationInputForm;
   // Dialog for displaying saved estimations
   static JDialog savedEstimationsDialog;
   // Form to display saved estimations
```

```
    static StoredResultsForm savedEstimationsForm;
    // Entity manager for database operations
    static public EntityManager entityManager;

    public static void main(String[] args) throws UnsupportedLookAndFeelException,
ClassNotFoundException, InstantiationException, IllegalAccessException {
        // Initialize the entity manager for database operations
        initializeEntityManager();

        // Create the main application frame
        mainApplicationFrame = new JFrame("Оцінювання тривалості розробки застосунків на
мові C# для платформи PC");


        // Initialize the estimation form and set up UI components
        estimationInputForm = new EstimationForm();
        estimationInputForm.setupUIComponents();

        // Set up the menu bar for the application
        initializeMenu();

        // Set the content pane of the frame
        mainApplicationFrame.setContentPane(estimationInputForm.$$$getRootComponent$$$());
        mainApplicationFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

        // Pack and display the frame
        mainApplicationFrame.pack();
        mainApplicationFrame.setVisible(true);
        mainApplicationFrame.setLocationRelativeTo(null); // Center the frame on screen
    }

    private static void initializeEntityManager() {
        // Create properties for the database connection
        Properties databaseProperties = new Properties();
        databaseProperties.put(TopLinkProperties.JDBC_URL, getDatabaseJDBCUrl());

        // Create the entity manager for database operations
        entityManager = Persistence.createEntityManagerFactory("csharp_estimations.db",
databaseProperties).createEntityManager();
    }

    private static void initializeMenu() {
        // Create a menu bar
        JMenuBar menuBar = new JMenuBar();

        // Menu item for viewing saved estimations
        JMenu viewEstimationsMenu = new JMenu("Збереженні результати");
        viewEstimationsMenu.addItemListener(new ItemListener() {
            @Override
```

```
            public void itemStateChanged(ItemEvent event) {
                // Show saved estimations dialog if the menu item is selected
                if (ItemEvent.SELECTED != event.getStateChange())
                    return;
                ((JMenu) event.getSource()).setSelected(false); // Deselect the menu item
                displaySavedEstimationsDialog();
            }
        });
        menuBar.add(viewEstimationsMenu);

        // Menu item for exiting the application
        JMenu exitMenu = new JMenu("Вихід");
        exitMenu.addItemListener(new ItemListener() {
            @Override
            public void itemStateChanged(ItemEvent event) {
                // Exit the application if the menu item is selected
                if (ItemEvent.SELECTED != event.getStateChange())
                    return;
                terminateApplication();
            }
        });
        menuBar.add(exitMenu);

        // Set the menu bar to the main frame
        mainApplicationFrame.setJMenuBar(menuBar);
    }

    public static String getDatabaseJDBCUrl() {
        try {
            // Create a URL for the SQLite database file
            File databaseFile = new File("csharp_estimations.db");
            System.out.println("file:jdbc:sqlite:" + databaseFile.getAbsolutePath());
            return new URL("file:jdbc:sqlite:" + databaseFile.getAbsolutePath()).getPath();
        } catch (MalformedURLException exception) {
            terminateApplication(exception); // Exit if there is an error with the URL
        }
        return null;
    }

    public static void displaySavedEstimationsDialog() {
        // Create a dialog to show saved estimations
        savedEstimationsDialog = new JDialog(mainApplicationFrame, "Збереженні результати
оцінювань тривалості", true);
        savedEstimationsDialog.setResizable(true);

        // Initialize the saved estimations form and set up UI components
        savedEstimationsForm = new StoredResultsForm();
        savedEstimationsForm.setupUIComponents();
```

```
        // Set the content pane of the dialog
        savedEstimationsDialog.setContentPane(savedEstimationsForm.$$$getRootComponent$$$());

savedEstimationsDialog.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);

        // Pack and display the dialog
        savedEstimationsDialog.pack();
        savedEstimationsDialog.setVisible(true);
    }

    public static void terminateApplication(Exception exception) {
        // Print the stack trace of the exception and exit the application
        exception.printStackTrace(System.err);
        terminateApplication();
    }

    public static void terminateApplication() {
        // Hide and dispose the main application frame
        mainApplicationFrame.setVisible(false);
        mainApplicationFrame.dispose();
    }
}
```

EstimationForm.java

```
package CSharpDevEstimate.forms;

import CSharpDevEstimate.*;
import CSharpDevEstimate.entities.*;
import Util.MathUtil;
import com.intellij.uiDesigner.core.*;
import org.apache.commons.math3.distribution.TDistribution;

import javax.swing.*;
import javax.swing.plaf.*;
import javax.swing.table.DefaultTableModel;
import javax.swing.text.*;
import java.awt.*;
import java.awt.event.*;
import java.text.SimpleDateFormat;
import java.util.*;


public class EstimationForm {
        private JPanel Panel;
        private JTextField effortTextField;
        private JTable estimationResults;
        public JButton ButtonRecalculate;
```

```
	public JTextField AVGeValue;
	private JTextField AppNameValue;
	private JButton StoreResultsBtn;
	public JLabel SxxLabel;
	private JTextField ConfValue;
	private JLabel EffortLabel;
	private JLabel AppNameLabel;
	private JLabel ProbabilityLabel;
	private JLabel B0Label;
	private JTextField B0Value;
	private JLabel B1Label;
	private JTextField B1Value;
	private JTextField NValue;
	private JTextField SeValue;
	private JTextField SSEValue;
	private Estimation estimate;

	public void setupUIComponents() {
		//estimationResults.setVisible(false);
		StoreResultsBtn.setVisible(false);

		ButtonRecalculate.addActionListener(new ActionListener() {
			@Override
			public void actionPerformed(ActionEvent e) {
				Double confidenceLevel = Double.parseDouble(ConfValue.getText())
/ 100;

				estimate = new Estimation();
				estimate.app_name = AppNameValue.getText();
				estimate.effort = Integer.parseInt(effortTextField.getText());

				int n = Integer.parseInt(NValue.getText());
				double b0 = Double.parseDouble(B0Value.getText());
				double b1 = Double.parseDouble(B1Value.getText());
				double sse = Double.parseDouble(SSEValue.getText());
				double e_avg = Double.parseDouble(AVGeValue.getText());
				double Se = Double.parseDouble(SeValue.getText());
				int v = n - 2; //Degrees of freedom
				double tStudent = (new
TDistribution(v)).inverseCumulativeProbability(1 - (1 - confidenceLevel) / 2);

				double e_z = Math.log10(estimate.effort.doubleValue());
				double t_z_estimate = estimateY(e_z, b0, b1);
				double t_z_estimate_l1 = estimateYConfidence(e_z, b0, b1, false,
false, sse, e_avg, Se, n, tStudent);
				double t_z_estimate_r1 = estimateYConfidence(e_z, b0, b1, true, false,
sse, e_avg, Se, n, tStudent);
				double t_z_estimate_l2 = estimateYConfidence(e_z, b0, b1, false, true,
sse, e_avg, Se, n, tStudent);
```

```
                                double t_z_estimate_r2 = estimateYConfidence(e_z, b0, b1, true, true,
sse, e_avg, Se, n, tStudent);

                                estimate.time = MathUtil.round(Math.pow(10, t_z_estimate), 1);
                                estimate.time_estimation_interval_top =
MathUtil.round(Math.pow(10, t_z_estimate_r2), 1);
                                estimate.time_estimation_interval_bottom =
MathUtil.round(Math.pow(10, t_z_estimate_l2), 1);
                                estimate.time_confidence_interval_top =
MathUtil.round(Math.pow(10, t_z_estimate_r1), 1);
                                estimate.time_confidence_interval_bottom =
MathUtil.round(Math.pow(10, t_z_estimate_l1), 1);

                                final DefaultTableModel dataModel = new DefaultTableModel(0, 2);

                                dataModel.addRow(new String[]{"Назва застосунку",
estimate.app_name});
                                dataModel.addRow(new String[]{"Трудомісткість",
estimate.effort.toString() + " люд.-год."});

                                dataModel.addRow(new String[]{"", ""});
                                dataModel.addRow(new String[]{"Результати:", ""});
                                dataModel.addRow(new String[]{"Оцінка тривалості",
estimate.time.toString() + " міс."});
                                dataModel.addRow(new String[]{"Довірчий інтервал тривалості",
"[" + estimate.time_confidence_interval_bottom + ", " + estimate.time_confidence_interval_top + "]
міс."});
                                dataModel.addRow(new String[]{"Інтервал передбачення
тривалості", "[" + estimate.time_estimation_interval_bottom + ", " +
estimate.time_estimation_interval_top + "] міс."});
                                estimationResults.setModel(dataModel);
                                estimationResults.setVisible(true);
                                estimationResults.setTableHeader(null);

                                StoreResultsBtn.setVisible(true);
                        }
                });

                StoreResultsBtn.addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                                estimate.date = new SimpleDateFormat("yyyy-MM-dd").format(new
Date());
                                EstimationApplication.entityManager.getTransaction().begin();
                                EstimationApplication.entityManager.persist(estimate);
                                EstimationApplication.entityManager.getTransaction().commit();
                        }
                });
        }
```

```
	protected double estimateY(double x, double b0, double b1) {
		return b0 + b1 * x;
	}

	protected double estimateYConfidence(double x, double b0, double b1, boolean top, boolean
for_values, double sse, double x_avg, double Sxx, int n, double tStudent) {
		double sign = top ? +1 : -1;
		double k1 = for_values ? 1 : 0;

		return estimateY(x, b0, b1) + sign * tStudent * Math.sqrt(sse / (n - 2)) * Math.sqrt(k1
+ 1. / n + Math.pow(x - x_avg, 2) / Sxx);
	}
```

ДОДАТОК В – ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ РОЗРОБКИ ЗАСТОСУНКІВ НА МОВІ C# ДЛЯ ПЛАТФОРМИ PC

1 Загальні відомості

Повна назва розроблюваного проекту: «Програмне забезпечення для оцінювання тривалості розробки застосунків на мові C# для платформи PC».

Коротка назва: «Estimation».

Для роботи програмного забезпечення необхідні такі програмні продукти: операційна система Windows 7 чи вище, або інша, в якій встановлена версія JRE не нижче версії Java 1.8.

Розроблюване програмне забезпечення було написано з використанням мови програмування Java.

2 Функціональне призначення розробки

Дане програмне забезпечення призначене для оцінювання тривалості розробки застосунків на мові C# для платформи PC.

Програмне забезпечення повинно мати можливість виконання наступних функцій:

– Внесення параметрів моделі (внесення параметрів N, b_0, b_1, AVGe, SSE, Se розробленої моделі).

– Внесення довірчої ймовірності (Conf).

– Внесення інформації про застосунок (назви та трудомісткості застосунку, що розробляється).

– Оцінювання тривалості розробки застосунку.

– Збереження результатів оцінювання в базу даних.

– Перегляд збережених результатів оцінювання.

3 Технічні засоби

Для функціонування ПЗ рекомендується наступний склад технічних засобів з мінімальними характеристиками:

– ПК з процесором Core i3;

– Оперативна пам'ять –1 Гб;

– Обсяг на жорсткому диску –500 Мб.

4 Вхідні дані

Вхідні дані заносяться в відповідні поля форми відповідно до типів даних.

Вхідними даними є параметри моделі (параметри N, b0, b1, AVGe, SSE, Se розробленої моделі), значення довірчої ймовірності (Conf), інформація про застосунок (назва та трудомісткість застосунку, що розробляється).

5 Вихідні дані

Вихідні дані виводяться в відповідні місця екранних форм та зберігаються в базі даних.

Вихідними даними є назва застосунку, трудомісткість застосунку, розрахована тривалість розробки застосунку, довірчий інтервал тривалості, інтервал передбачення тривалості.

ДОДАТОК Г – ІНСТРУКЦІЯ КОРИСТУВАЧА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ РОЗРОБКИ ЗАСТОСУНКІВ НА МОВІ C# ДЛЯ ПЛАТФОРМИ PC

Завантажуємо програму

В вікні, що з’явилося, є можливість переглянути збережені результати («Збережені результати»), завершити роботу програми («Вихід»), а також ввести параметри моделі, довірчу ймовірність, інформацію про застосунок та виконати оцінювання тривалості розробки застосунку (рис.Г.1).

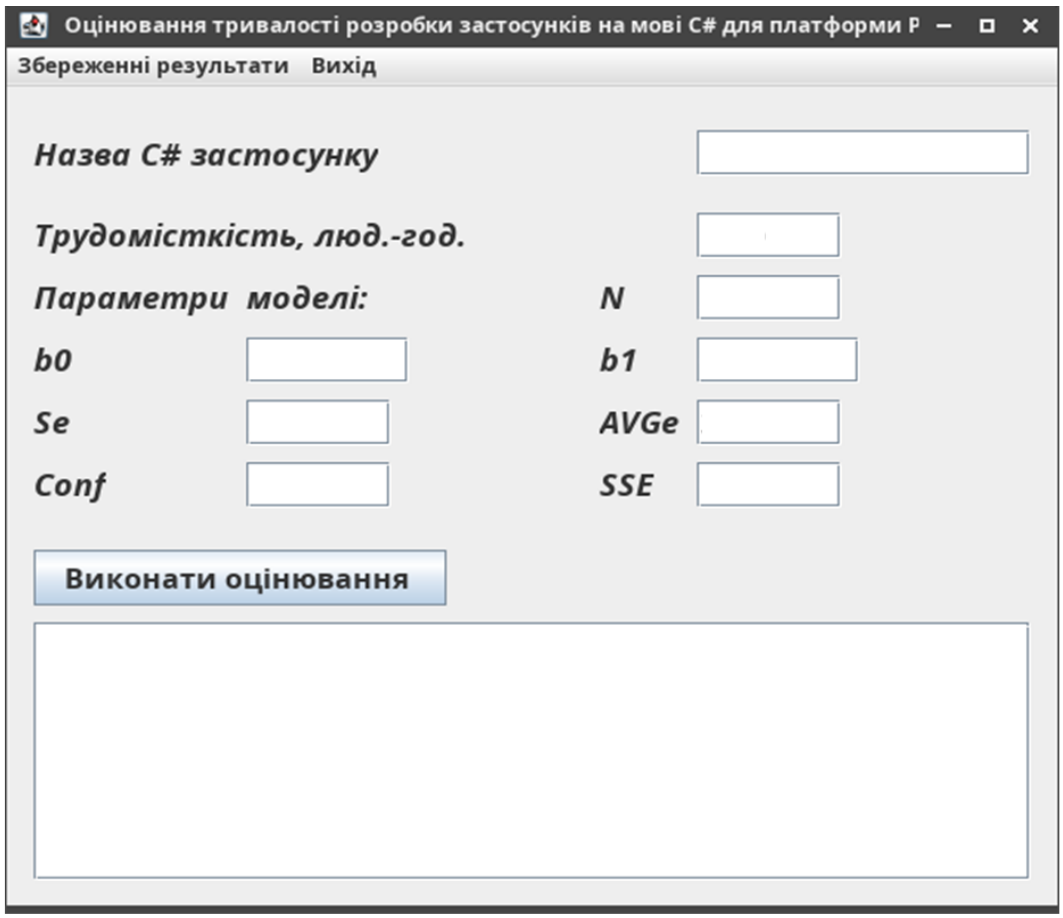

Рисунок Г.1 – Початкове вікно програми

Для оцінювання тривалості розробки застосунку необхідно ввести наступне:

–параметри моделі (N, b0, b1, AVGe, SSE, Se),

– довірчу ймовірність (Conf),

– інформацію про застосунок (назву C# застосунку, трудомісткість)

Вікно з введеними даними представлене на рисунку Г.2.

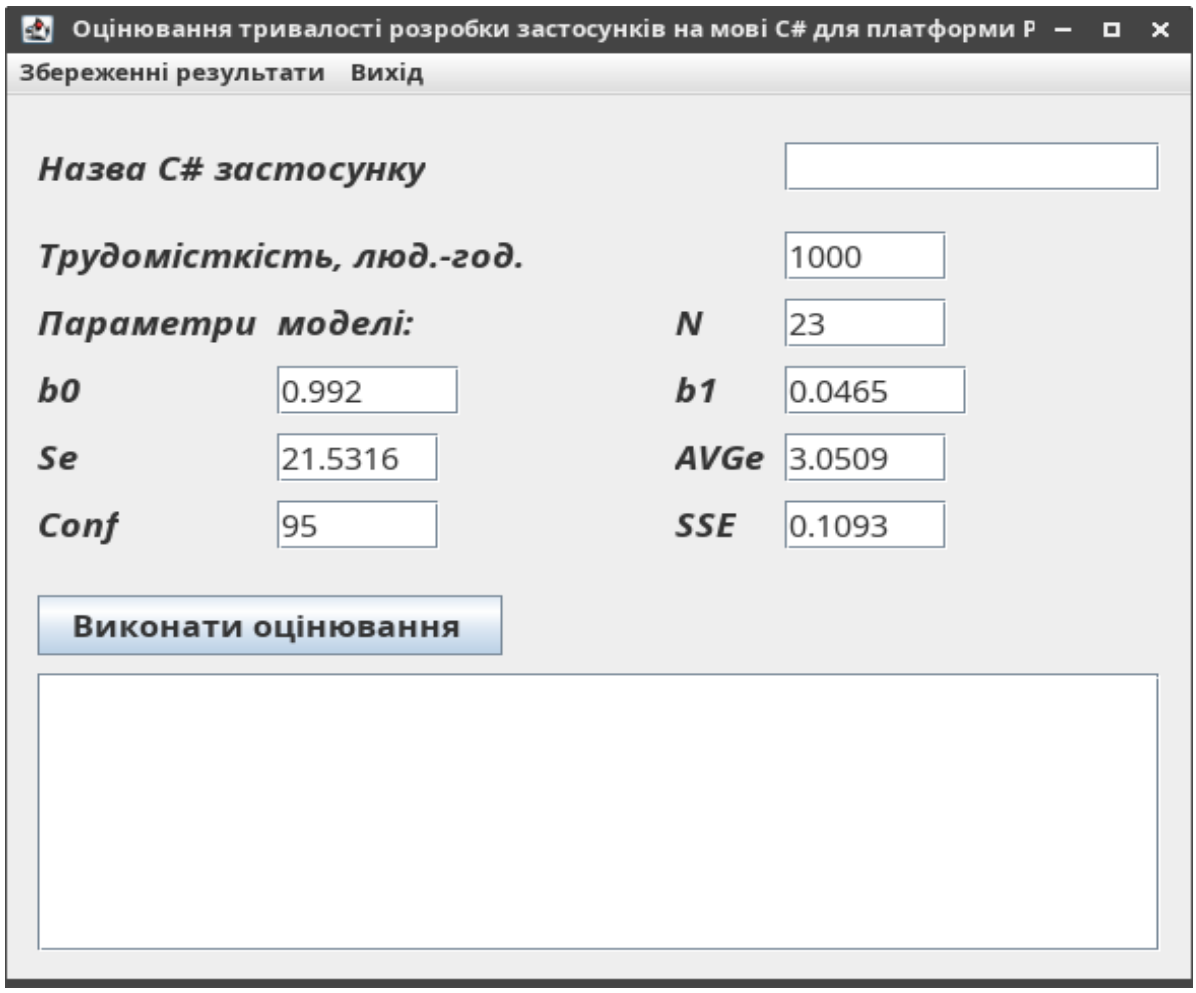

Рисунок Г.2 – Вікно з введеними даними

Після введення даних необхідно натиснути на кнопку «Виконати оцінювання» для отримання оцінки тривалості розробки застосунків на мові C# згідно розробленої моделі. Отримаємо вікно з результатами оцінювання (рис. Г.3)

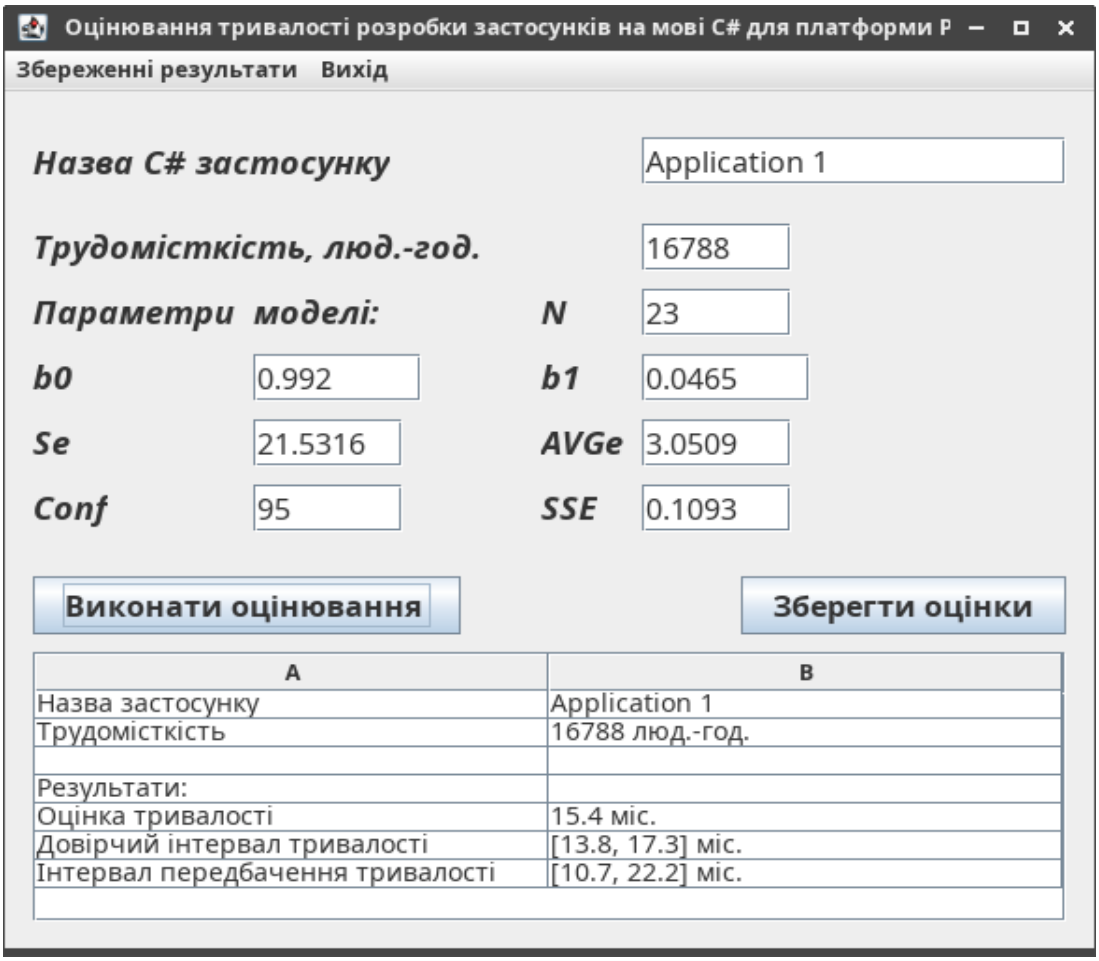

Рисунок Г.3 – Вікно з результатами оцінювання

Вікно містить

1. Вхідні дані:

– назву застосунку

– трудомісткість

2. Результати оцінювання:

– оцінку тривалості розробки в місяцях,

– довірчий інтервал тривалості в місяцях,

– інтервал передбачення тривалості в місяцях.

В формі з результатами оцінювання доступне збереження результатів в базу даних (кнопка «Зберегти оцінки»).

При виборі пункту меню «Збережені результати» відобразиться вікно з наявними в базі даних результатами оцінювання (рис. Г.4).

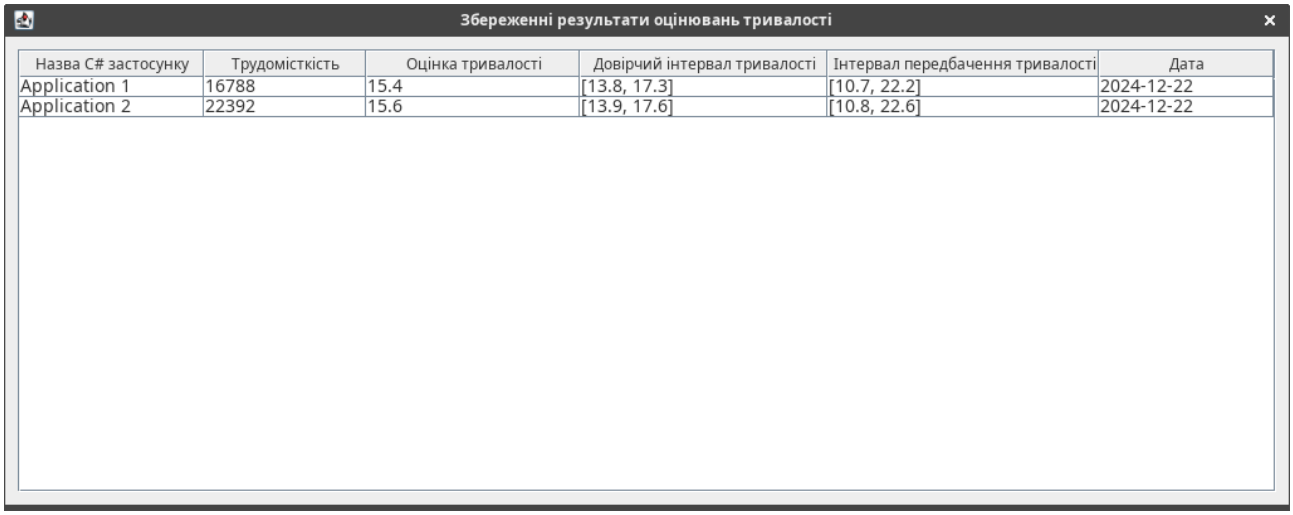

Рисунок Г.4 – Вікно з збереженими в базі даних результатами оцінювання

ДОДАТОК Д – ПРОГРАМА І МЕТОДИКА ВИПРОБУВАНЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ РОЗРОБКИ ЗАСТОСУНКІВ НА МОВІ C# ДЛЯ ПЛАТФОРМИ PC

1 Об’єкт випробовування

Об’єкт випробувань: Програмне забезпечення для оцінювання тривалості розробки застосунків на мові C# для платформи PC.

Галузь застосування: Розробка ПЗ, управління програмними проектами.

Позначення випробуваної програми: «Estimation».

2 Мета випробовування

Метою випробувань є необхідність переконання у відповідності функціоналу розробленого програмного забезпечення технічному завданню, яке наведено в додатку А, виявлення можливих помилок.

3 Вимоги до програми

Під час випробування програмного забезпечення потрібно перевірити, чи відповідають її функціональні характеристики вимогам, які визначені та описані у розділі «Вимоги до функціональних характеристик» технічного завдання.

4 Вимоги до програмної документації

Програмна документація повинна містити:

- технічне завдання;
- текст програми;
- опис програми;
- програму та методика випробувань;
- інструкцію користувача.

5 Засоби та порядок випробувань

Програмні засоби випробувань

До програмних засобів, необхідних для випробувань, належать: операційна система Windows 7 чи вище, або інша, в якій встановлена версія JRE не нижче версії Java 1.8.

Апаратні засоби випробувань

Для випробувань необхідний наступний склад апаратних засобів з мінімальними характеристиками:

– ПК з процесором Core i3;

– Оперативна пам'ять –1 Гб;

– Обсяг на жорсткому диску –500 Мб.

Порядок проведення випробувань

Проведення випробування програмного забезпечення повинно відбуватися в наступному порядку:

- виконуються інструкції до кожної функції;

- робиться аналіз отриманих результатів і встановлюється відповідність програмного продукту вимогам і системним документам.

При виявлені помилок у роботі системи складається їх перелік і обговорюється термін їх виправлення розробником. Після цього замовник проводить повторне випробування в повному обсязі.

6 Методика випробувань

Випробування будемо проводити за стратегією «чорного ящика». В таблиці Д.1 наведено перелік випробувань, що пропонуються до виконання:

Таблиця Д.1 – Перелік випробувань, що пропонуються до виконання:

№	Дія	Очікуваний результат
1	Внесення в форму параметрів моделі	Відображення результату введення параметрів моделі
2	Внесення в форму довірчої ймовірності	Відображення результату введення довірчої ймовірності
3	Внесення в форму інформації про застосунок (назви та трудомісткості)	Відображення результату введення назви та трудомісткості застосунку
4	Оцінювання тривалості розробки застосунку	Відображення форми з вхідними даними та результатом оцінювання тривалості розробки
5	Збереження результатів оцінювання в базу даних	Запис в базу даних вхідних даних та результату оцінювання
6	Перегляд збережених результатів оцінювання	Відображення форми зі збереженими в базі даних результатами

По результатам випробувань робиться висновок про відповідність функціоналу програми вимогам, які сформульовані в розділі «Вимоги до функціональних характеристик» технічного завдання.