МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

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

Програмного забезпечення автоматизованих систем
(повна назва кафедри)

Пояснювальна записка

до кваліфікаційної (магістерської) роботи

за темою Удосконалення моделі ISBSG для оцінювання тривалості програмних проектів для mainframe та розробка програмного забезпечення для її реалізації

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

Мишенін О.І.
(підпис, прізвище та ініціали)

Керівник Пухалевич А.В.
(підпис, прізвище та ініціали)

Рецензент _____
(підпис, прізвище та ініціали)

Завідувач кафедри Приходько С.Б
(підпис, прізвище та ініціали)

м. Миколаїв – 2020

РЕФЕРАТ

Мишенін Олег Ігорович

«Удосконалення моделі ISBSG для оцінювання тривалості програмних проектів для mainframe та розробка програмного забезпечення для її реалізації

(тема кваліфікаційної роботи)

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

Обсяг роботи: 98 стор., 8 табл., 10 рис., 34 використаних джерел, 5 додатків.

Актуальність теми роботи: необхідність підвищення достовірності оцінювання тривалості програмних проектів для mainframe.

Мета та завдання дослідження. Метою роботи є підвищення достовірності оцінювання тривалості програмних проектів для mainframe.

Об'єкт дослідження: процес оцінювання тривалості програмних проектів для mainframe.

Предмет дослідження: математичні моделі для оцінювання тривалості програмних проектів для mainframe.

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

Наукова новизна одержаних результатів: удосконалено нелінійну регресійну модель ISBSG для оцінювання тривалості програмних проектів для mainframe в залежності від трудомісткості розробки цих програмних проектів за рахунок побудови рівнянь границь довірчого інтервалу та рівнянь границь інтервалу прогнозування нелінійних регресій, що дозволяє підвищити достовірність оцінювання тривалості програмних проектів для mainframe.

Практичне значення одержаних результатів. Розроблене програмне забезпечення для оцінювання тривалості програмних проектів для mainframe дозволить скоротити час відповідного оцінювання, забезпечить зберігання результатів оцінювання, а також надасть швидкий доступ до результатів попередніх оцінювань.

Апробація результатів досліджень. Основні положення і результати досліджень, викладені в роботі, пройшли апробацію на III Всеукраїнській науково-практичній інтернет-конференції студентів, аспірантів та молодих вчених за тематикою «Сучасні комп'ютерні системи та мережі в управлінні» (м. Херсон, 30 листопада 2020 р.).

Публікації. Результати роботи опубліковано в 1 матеріалі конференції.

Ключові слова: оцінювання тривалості, програмні проекти, регресійні моделі, нормалізуючі перетворення, нелінійна регресія.

ABSTRACT

Myshenin Oleh Ihorovych

«Improving the ISBSG model for estimating the mainframe software project duration and developing the software for its implementation»

(тема кваліфікаційної роботи)

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

Volume: 98 p., 8 tables, 10 figures, 34 sources, 5 appendixes.

Topic Relevance: Demand of improvement of mathematical models for estimating the duration of software applications development and developing software for its implementation.

Research goal and objectives: increasing the reliability of estimation of the duration of the mainframe software application development.

Object of research: the process of the estimation the duration of the mainframe software applications development.

Subject of research: mathematical models for estimating the duration of the mainframe software applications development.

Methods of research: methods of probability theory, mathematical statistics, regression analysis.

Scientific contribution: nonlinear regression model ISBSG for estimating the mainframe software project duration depending on the effort was improved by developing the equations of the ranges of confidence interval and prediction interval of the nonlinear regression that enabled increasing the reliability of estimation of the duration of software application development.

Practical value of obtained results. Developed software allows to automate and shortening the time of the estimation the duration of software applications development.

Approbation of the thesis results. III All-Ukrainian scientific-practical Internet conference of students, graduate students and young scientists on the topic "Modern computer systems and networks in management" (Kherson, November 30, 2020).

Key words: duration estimation, software projects, regression models, normalizing transformations, non-linear regression.

ЗМІСТ

ВСТУП

Актуальність теми.

Програмні проекти (або проекти з розробки програмного забезпечення) для платформи mainframe характеризуються своїм великим розміром, великою кількістю компонентів та підвищеними вимогами до надійності. Вказані характеристики призводять до того, що достовірне оцінювання тривалості розробки програмних проектів для цієї платформи є особливо проблематичним завданням. В зв'язку з цим відбувається постійне перевищення очікуваної тривалості проектів з розробки програмного забезпечення для платформи mainframe [1-4]. Через це не зупиняються дослідження по вдосконаленню моделей тривалості програмних проектів (включаючи побудову моделей тривалості програмних проектів із застосуванням методів теорії ймовірностей, математичної статистики, регресійного аналізу), та зі створення програмного забезпечення для їх реалізації.

Оцінюванню тривалості програмних проектів приділено велику кількість досліджень та наукових праць [1-28 і інші]. Більшість існуючих моделей для оцінювання тривалості програмних проектів є нелінійними регресійними моделями, які базуються на використанні нормалізуючих перетворень (в тому числі моделі COCOMO [6], моделі на основі нормалізуючого перетворення Джонсона [7], моделі ISBSG [8, 9]).

Емпіричні дані тривалості програмних проектів та трудомісткості програмних проектів мають закон розподілу, який відрізняється від Гаусівського закону розподілу [6-9, 25-28]. Тому моделі COCOMO [6] і більш нові моделі ISBSG [8, 9] побудовані з застосовуванням логарифмічного перетворення для нормалізації емпіричних даних. Ці моделі дозволяють отримати точкову оцінку тривалості в залежності від трудомісткості програмного проекту для платформ PC, mid-range, mainframe. Проте, достовірність отриманих оцінок зазвичай є

невисокою. Крім того, для цих моделей немає побудованих рівнянь довірчих інтервалів та рівнянь інтервалів прогнозування тривалості, хоча інтервальні оцінки є кращими, в порівняні з точковими [7]. Враховуючи розмір програмних проектів для платформи mainframe, навіть незначне підвищення достовірності оцінювання тривалості таких проектів може значно зменшити можливі втрати від недостовірного оцінювання.

Отже, підвищення достовірності оцінювання тривалості програмних проектів для mainframe, яке буде досягнуте при удосконалені моделі ISBSG для mainframe із використанням нормалізуючих перетворень [29, 30] за рахунок побудови рівнянь границь довірчого інтервалу та побудови рівняння границь інтервалу прогнозування нелінійних регресій, а також розробка програмного забезпечення для її реалізації є актуальним та має практичну цінність.

Мета роботи. Метою роботи є підвищення достовірності оцінювання тривалості програмних проектів для mainframe.

Завдання дослідження. Щоб досягнути поставленої мети, в дослідженні потрібно вирішити наступні завдання:

1. Провести аналіз існуючих методів та моделей оцінювання тривалості програмних проектів.
2. Удосконалити нелінійну регресійну модель ISBSG для оцінювання тривалості програмних проектів для mainframe в залежності від трудомісткості цих програмних проектів, побудувавши рівняння нелінійної регресії, рівняння границь довірчого інтервалу та рівняння границь інтервалу прогнозування нелінійної регресії тривалості програмних проектів в залежності від трудомісткості програмних проектів.
3. На основі удосконалених моделей створити програмне забезпечення для оцінювання тривалості програмних проектів для mainframe.

Об’єкт дослідження: процес оцінювання тривалості програмних проектів для mainframe.

Предмет дослідження: математичні моделі для оцінювання тривалості програмних проектів для mainframe.

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

Методи теорії ймовірностей, математичної статистики та регресійного аналізу використовувалися для побудови вдосконаленої нелінійної регресійної моделі для оцінювання тривалості програмних проектів для mainframe.

Наукова новизна одержаних результатів. *Удосконалено* нелінійну регресійну модель ISBSG для оцінювання тривалості програмних проектів для mainframe в залежності від трудомісткості розробки цих програмних проектів *за рахунок* побудови рівнянь границь довірчого інтервалу та рівнянь границь інтервалу прогнозування нелінійної регресії, *що дозволяє* підвищити достовірність оцінювання тривалості програмних проектів для mainframe.

Практичне значення одержаних результатів. Розроблене програмне забезпечення для оцінювання тривалості програмних проектів для mainframe дозволить скоротити час відповідного оцінювання, забезпечить зберігання результатів оцінювання, а також надасть швидкий доступ до результатів попередніх оцінювань.

Апробація результатів досліджень. Основні положення і результати досліджень, викладені в роботі, пройшли апробацію на III Всеукраїнській науково-практичній інтернет-конференції студентів, аспірантів та молодих вчених за тематикою «Сучасні комп’ютерні системи та мережі в управлінні» (м. Херсон, 30 листопада 2020 р.).

Публікації. Результати роботи опубліковано в 1 матеріалі конференції [31].

.

1 АНАЛІЗ ІСНУЮЧИХ МАТЕМАТИЧНИХ МОДЕЛЕЙ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ ПРОГРАМНИХ ПРОЕКТІВ

1.1 Дослідження існуючих методів і моделей для оцінювання тривалості програмних проектів

Моделям для оцінювання тривалості програмних проектів приділено велику кількість досліджень і наукових праць. Перелік моделей для оцінювання тривалості програмних проектів весь час збільшується, створюються все нові моделі і методи.

Головними методами для оцінювання тривалості робіт, які використовуються в тому числі для оцінювання тривалості програмних проектів є [5, 7]:

1. Експертне оцінювання.
2. Оцінювання за аналогами.
3. Параметричне оцінювання.

Окрім вказаних методів використовуються також методи, що беруть до уваги специфіку програмних проектів. Проаналізуємо всі ці методи та моделі, щоб виявити їх основні переваги та недоліки.

Методи, які основані на проведені експертного оцінювання

Методи, які основані на проведені експертного оцінювання, базуються на суб'єктивному рішенні людини-експерта або групи експертів, є найбільш широко використовуваними методами для оцінювання тривалості програмних проектів [10, 12].

Методи, що базуються на експертній оцінці мають свої недоліки, серед яких [7, 14]:

- підстави для отримання оцінок не є явними;

- складно знаходити висококваліфікованих експертів для кожного нового проекту;
- зв'язок між розміром системи і оцінюваними характеристиками – нелінійний [6]. Тому адекватна експертна оцінка може бути отримана лише в тому випадку, коли поточний проект і попередні є приблизно однакового трудомісткості.

Оцінювання за аналогами [7]

Оцінювання за аналогами передбачає використання таких параметрів як тривалість, бюджет, розмір, вага і складність попередніх подібних проектів в якості основи для оцінювання тих же параметрів або вимірювань майбутнього проекту [5]. При оцінюванні тривалості даний метод спирається на фактичну тривалість попередніх подібних проектів в якості основи для оцінювання тривалості поточного проекту. Цей підхід, що дозволяє оцінювати загальну величину, іноді адаптується в залежності від відомих відмінностей у складності проекту.

Найчастіше оцінювання тривалості за аналогами використовується для оцінювання тривалості проекту, коли обсяг детальної інформації про проект обмежений, наприклад на його ранніх фазах. При оцінюванні за аналогами використовується історична інформація та експертне оцінювання.

Як правило, оцінювання за аналогами обходиться дешевше і займає менше часу, ніж інші методи, але при цьому отримані оцінки зазвичай виявляються і менш точними. Оцінювання за аналогами може застосовуватися до всього проекту або до його частин, а також може використовуватися разом з іншими методами оцінювання. Оцінювання за аналогам виявляється найбільш надійним в тих випадках, коли попередні операції схожі по суті, а не тільки за формою, а члени команди проекту, що готують оцінки, мають необхідний досвід [5].

Алгоритмічні методи [7]

Алгоритмічні методи використовують ітераційний підхід, який базується на математичних моделях та формулах [10]. При цьому вхідними даними є трудомісткість розробки програмного забезпечення проекту (розрахована у

людино-годинах), та такі параметри, як апаратне забезпечення, програмне забезпечення, рівень командного досвіду, рівень досвіду менеджерів і метод, що використовується для розробки програмного забезпечення. Алгоритм оцінювання дозволяє за вхідними даними визначити оцінку тривалості та її точність. Алгоритмічні методи ітеративно запускаються кілька разів, щоб уточнити значення вхідних даних та щоб підвищити точність оцінювання. Обмеження цього класу методів проявляється, коли алгоритм використовується з неточними чи з неперевіреними даними. Більшість алгоритмічних методів дозволяють оцінювати тривалість, тривалість і навіть загальну вартість проектів [10]. До алгоритмічних методів відносяться COCOMO [6] і COCOMO 2.0 [11], нейроні мережі, метод критичного шляху, метод критичного ланцюга.

Методи, що базуються на математично-статистичних моделях тривалості [7]

Методи, що базуються на математично-статистичних моделях тривалості є особливо корисними, коли вони базуються на великій кількості історичних даних для аналізу [10]. До цих моделей відносяться лінійна і множинна регресії та їх модифікації.

Модель Путнема SLIM [15] (Software Life-cycle Model, модель життєвого циклу програмного забезпечення) була запропонована в 1978 р. Лоуренсом Путнемом (Lawrence Putnam) і стала першою нелінійною моделлю, що використовує емпіричні дані й знайшла практичне застосування при оцінюванні програмного забезпечення. Модель SLIM створена для програмного забезпечення розміром більш ніж 70 тисяч рядків коду і базується на припущенні, що витрати на розробку програмного забезпечення розподіляються згідно кривих Нордена-Рейлі, які є графіками функції, що показує розподіл робочої сили в часі [16]

Модель COCOMO [6] (Constructive Cost Model, модель витрат розробки) – це алгоритмічний метод оцінювання вартості та тривалості програмних проектів, який базується на регресійних моделях. Параметри регресійних моделей були

визначені за даними, зібраними за низкою проектів. Регресійні моделі COCOMO були побудовані для різних типів проектів:

- органічні (organic) – 2-50 тисяч рядків програмного коду (програми, призначені для проміжних розрахунків, для потреб програміста, для наукових обчислень);
- напів-розділені (semi-detached) – 50-300 тисяч рядків програмного коду (прикладні системи, компілятори);
- вбудовані (embedded) – більше 300 тисяч рядків коду (системи контролю повітряного руху, мережі АТМ, військові системи).

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

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

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

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

де D – тривалість проекту (в місяцях);

E – трудомісткість проекту (в людино-годинах).

Модель ISBSG [8] – нелінійна регресійна модель для оцінювання тривалості програмних проектів. Як фактор в ній було використано трудомісткість проектів з розробки програмного забезпечення:

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

де D – тривалість проекту (в місяцях);

E – трудомісткість проекту (в людино-годинах).

Крім того, моделі ISBSG було побудовано для різних платформ: [9]

$$D = 1{,}936 \cdot E^{0{,}201} \text{ (для платформи PC)}, \quad (1.5)$$

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

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

де D – тривалість проекту (в місяцях);

E – трудомісткість проекту (в людино-годинах).

Емпіричні дані тривалості програмних проектів та трудомісткості програмних проектів мають закон розподілу, який відрізняється від Гаусівського закону розподілу [6-9, 25-28]. Тому моделі COCOMO [6] і більш нові моделі ISBSG [8, 9] побудовані з застосовуванням нормалізуючих перетворень. Для нормалізації було використано перетворення у вигляді десяткового логарифму. Ці моделі дозволяють отримати точкову оцінку тривалості в залежності від трудомісткості програмного проекту. Проте, достовірність отриманих оцінок зазвичай є невисокою. Крім того, для цих моделей немає побудованих рівнянь довірчих інтервалів та рівнянь інтервалів прогнозування тривалості, хоча інтервальні оцінки є кращими, в порівняні з точковими [7]. Враховуючи розмір програмних проектів для платформи mainframe, навіть незначне підвищення достовірності оцінювання тривалості таких проектів може значно зменшити можливі втрати від недостовірного оцінювання.

Змішані методи [7]

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

1.2 Обґрунтування необхідності проведення досліджень

Дослідження існуючих методів і моделей для оцінювання тривалості програмних проектів показує, що для оцінювання тривалості програмних проектів найчастіше використовуються нелінійні регресійні моделі тривалості програмних проектів в залежності від трудомісткості розробки програмних проектів, такі як COCOMO [6] і більш нові моделі ISBSG [8, 9]. Проте, достовірність отриманих оцінок зазвичай є невисокою.

Емпіричні дані тривалості програмних проектів та емпіричні дані трудомісткості програмних проектів мають закон розподілу, який відрізняється від Гаусівського закону розподілу [6-9, 25-28]. Тому регресійні моделі COCOMO і ISBSG не дозволяють виконувати оцінювання довірчих інтервалів тривалості програмних проектів.

Часто в таких випадках закон розподілу вважається нормальним, або застосовується непараметричний підхід. Застосування непараметричного підходу для оцінювання довірчих інтервалів чи виконання інтервального оцінювання в припущенні про нормальний закон розподілу емпіричних даних буде знижувати достовірність оцінювання тривалості програмних проектів [7, 32].

Щоб підвищити достовірність оцінювання тривалості програмних проектів, можна побудувати математичні моделі для оцінювання тривалості, які дадуть можливість робити оцінювання довірчого інтервалу тривалості програмних проектів та інтервалу прогнозування тривалості програмних проектів, тому що інтервальні оцінки дадуть можливість брати до уваги невизначеності точкових оцінок тривалості програмних проектів.

Через те, що емпіричні дані тривалості програмних проектів та емпіричні дані трудомісткості програмних проектів мають закон розподілу, який відрізняється від Гаусівського закону розподілу, з'являється проблема побудови рівнянь довірчого інтервалу тривалості програмних проектів та рівнянь інтервалу прогнозування тривалості програмних проектів. Для вдосконалення нелінійної

регресійної моделі ISBSG для mainframe можна застосувати метод побудови рівнянь нелінійної регресії та метод побудови довірчого інтервалу регресії на основі нормалізуючих перетворень [29, 30].

Застосування цих методів на основі нормалізуючих перетворень дозволить видалити викиди з емпіричних даних та побудувати рівняння довірчого інтервалу тривалості програмних проектів та рівнянь інтервалу прогнозування тривалості програмних проектів.

При побудові моделей ISBSG для оцінювання тривалості програмних проектів використовувався набір емпіричних даних по програмних з репозитарію ISBSG (International Software Benchmarking Standards Group). Цей же набір потрібно використати для вдосконалення моделі ISBSG для оцінювання тривалості програмних проектів для mainframe в залежності від трудомісткості розробки цих програмних проектів Використання цього репозитарію пов'язане з великою кількістю програмних проектів (790 проектів), присутністю даних щодо якості статистичних даних для кожного проекту, а також з присутністю даних щодо платформи, тривалості та трудомісткості кожного програмного проекту. До репозитарію ISBSG включено проекти зі створення нового та вдосконалення існуючого програмного забезпечення на мовах JAVA, C, C++, Visual Basic, Access, Powerbuilder, COBOL/COBOL II, ПЛ/1, Focus.

Щоб підвищити достовірність оцінювання тривалості програмних проектів для mainframe, потрібно вирішити наступні завдання:

1. Вдосконалити нелінійну регресійну модель ISBSG для оцінювання тривалості програмних проектів для mainframe в залежності від трудомісткості цих програмних проектів, видаливши викиди з емпіричних даних та побудувавши рівняння границь довірчого інтервалу та рівняння границь інтервалу прогнозування нелінійної регресії тривалості програмних проектів в залежності від трудомісткості програмних проектів.
2. На основі вдосконалених моделей розробити програмне забезпечення для оцінювання тривалості програмних проектів для mainframe.

2 ПОБУДОВА НЕЛІНІЙНОЇ РЕГРЕСІЙНОЇ МОДЕЛІ ТРИВАЛОСТІ ПРОГРАМНИХ ПРОЕКТІВ ДЛЯ MAINFRAME

Одними з моделей, як в даний час часто використовуються для оцінювання тривалості програмних проектів є COCOMO [6] і ISBSG [8, 9]: нелінійні регресійні моделі для оцінювання тривалості програмних проектів в залежності від трудомісткості. Для побудови цих моделей виконувалась нормалізація емпіричних даних тривалості програмних проектів та емпіричних даних трудомісткості програмних проектів. Використання нормалізуючих перетворень при побудові регресійних моделей є необхідним, тому що емпіричні дані тривалості програмних проектів та трудомісткості програмних проектів мають закон розподілу, який відрізняється від Гаусівського закону розподілу [6-9, 25-28].

Для побудови нелінійних регресійних моделей ISBSG, в тому числі для платформи mainframe, було використано десятковий логарифм для нормалізації вказаних емпіричних даних тривалості програмних проектів та емпіричних даних трудомісткості програмних проектів. Щоб вдосконалити нелінійну регресійну моделі ISBSG для оцінювання тривалості програмних проектів для mainframe необхідно видалити викиди з емпіричних даних та побудувати рівняння довірчого інтервалу тривалості програмних проектів та рівняння інтервалу прогнозування тривалості програмних проектів з використанням нормалізуючих перетворень.

2.1 Метод побудови нелінійних регресійних моделей з використанням нормалізуючих перетворень

Побудова нелінійних регресійних моделей з використанням нормалізуючих перетворень проходить в декілька етапів: виконується нормалізація емпіричних

даних з використанням нормалізуючого перетворення; за нормалізованими даними будується лінійна регресія; за побудованим рівнянням лінійної регресії з використанням перетворення, зворотного до нормалізуючого, будується нелінійна регресійна модель, а також рівняння нижньої та верхньої границь довірчого інтервалу нелінійної регресії, рівняння нижньої та верхньої границь інтервалу прогнозування нелінійної регресії [7].

Спочатку приведемо формули, що використовуються для знаходження статистичних значень випадкових величин [7].

Знаходження середнього значення випадкової величини x виконується як

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i , \tag{2.1}$$

де n - це кількість емпіричних значень випадкової величини x;

$x_i - i$-значення випадкової величини x.

Знаходження незміщеної вибіркової дисперсії випадкової величини x виконується як

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2 , \tag{2.2}$$

де n - це кількість емпіричних значень випадкової величини x;

$x_i - i$-значення випадкової величини x;

$\bar{x}$ – середнє значення випадкової величини x, що знаходиться за (2.1).

Знаходження коефіцієнту асиметрії випадкової величини x виконується як

$$A = \frac{n}{(n-1)(n-2)}\sum_{i=1}^{n}(x_i - \bar{x})^3 \frac{1}{\left(S^2\right)^{3/2}} , \tag{2.3}$$

де n - це кількість емпіричних значень випадкової величини x;

$x_i - i$-значення випадкової величини x;

$\bar{x}$ – середнє значення випадкової величини x, що знаходиться за (2.1);

S^2 – значення незміщеної вибіркової дисперсії величини x, що знаходиться за (2.2).

Знаходження коефіцієнту ексцесу випадкової величини x виконується як

$$\varepsilon = \frac{(n^2-2n+3)\sum_{i=1}^{n}(x_i-\bar{x})^4 + (2n-3)\frac{3}{n}\left[\sum_{i=1}^{n}(x_i-\bar{x})^2\right]^2}{(n-1)(n-2)(n-3)(S^2)^2}, \quad (2.4)$$

де n - це кількість емпіричних значень випадкової величини x;

x_i – i-значення випадкової величини x;

$\bar{x}$ – середнє значення випадкової величини x, що знаходиться за (2.1);

S^2 – значення незміщеної вибіркової дисперсії величини x, що виконується за (2.2).

Щоб вдосконалити нелінійну регресійну моделі ISBSG для оцінювання тривалості програмних проектів для mainframe в залежності від трудомісткості розробки цих програмних використаємо дані по тривалості та трудомісткості 215 програмних проектів з репозитарію International Software Benchmarking Standards Group.

Емпіричні дані тривалості програмних проектів та трудомісткості програмних проектів мають закон розподілу, який відрізняється від Гаусівського закону розподілу [6-9, 25-28]. Тому для побудови нелінійних регресійних моделей буде використано нормалізовані значення тривалості програмних проектів та нормалізовані значення трудомісткості програмних проектів.

Розглянемо, як виконується побудова нелінійних регресійних моделей з використанням нормалізуючих перетворень [7].

Нормалізацію випадкової величини виконуємо за перетворенням у вигляді десяткового логарифму:

$$z = \log_{10} x \tag{2.5}$$

де z – нормально розподілена випадкова величина;

x – випадкова величина, яка нормалізується.

Нехай випадкова величина D – це емпіричні значення тривалості програмних проектів (в місяцях), а випадкова величина E – це емпіричні значення трудомісткості програмних проектів (в людино-годинах).

З випадкових величин D та E, використовуючи нормалізуюче перетворення, буде отримано нормалізовані випадкові величини z_D та z_E. Це дозволить побудувати рівняння лінійної регресії для нормалізованих значень:

$$\hat{z}_D(z_E) = b_0 + b_1 z_E, \tag{2.6}$$

де b_0 і b_1 - коефіцієнти рівняння лінійної регресії.

Для рівняння лінійної регресії коефіцієнти b_0 і b_1 (2.6) обчислюються за допомогою методу найменших квадратів [34]:

$$b_1 = \frac{\overline{z_E z_D} - \overline{z_E} \cdot \overline{z_D}}{\overline{z_E^2} - \overline{z_E}^2};\ b_0 = \overline{z_D} - b_1 \cdot \overline{z_E}, \tag{2.7}$$

де $\overline{z_D} = \frac{1}{n}\sum_{i=1}^{n} z_{Di}$; $\overline{z_E} = \frac{1}{n}\sum_{i=1}^{n} z_{Ei}$; $\overline{z_E z_D} = \frac{1}{n}\sum_{i=1}^{n} z_{Ei} z_{Di}$; $\overline{z_E^2} = \frac{1}{n}\sum_{i=1}^{n} z_{Ei}^2$.

Знаходження довірчого інтервалу лінійної регресії (2.6) виконується як [34]

$$[\bar{z}_D(z_E)] = \hat{z}_D(z_E) \pm t_{\alpha/2,n-2} \cdot \sqrt{s_{z_D}^2} \sqrt{\frac{1}{n} + \frac{(z_E - \bar{z}_E)^2}{S_{z_E}}}, \tag{2.8}$$

де n - це кількість емпіричних значень;

$s_{z_D}^2 = \frac{1}{n-2}\sum_{i=1}^{n}\left(z_{D_i} - \hat{z}_D\left(z_{E_i}\right)\right)^2$; $S_{z_E} = \sum_{i=1}^{n}(z_{E_i} - \bar{z}_E)^2$;

$t_{\alpha/2,n-2}$ - квантіль t-розподілу Стьюдента, визначається за таблицею верхніх

$100\alpha\%$ точок t-розподілу Стьюдента за рівнем значимості $\alpha/2$ та кількістю ступенів вільності $n-2$.

Знаходження інтервалу прогнозування лінійної регресії (2.6) виконується як [34]

$$\left[z_D(z_E)\right] = \hat{z}_D(z_E) \pm t_{\alpha/2,n-2} \cdot \sqrt{s^2_{z_D}} \sqrt{1 + \frac{1}{n} + \frac{(z_E - \bar{z}_E)^2}{S_{z_E}}}, \qquad (2.9)$$

де n - це кількість емпіричних значень;

$s^2_{z_D} = \frac{1}{n-2}\sum_{i=1}^{n}\left(z_{D_i} - \hat{z}_D\left(z_{E_i}\right)\right)^2$; $S_{z_E} = \sum_{i=1}^{n}(z_{E_i} - \bar{z}_E)^2$;

$t_{\alpha/2,n-2}$ - квантіль t-розподілу Стьюдента, визначається за таблицею верхніх $100\alpha\%$ точок t-розподілу Стьюдента за рівнем значимості $\alpha/2$ та кількістю ступенів вільності $n-2$.

Зважаючи на (2.6) і те, що нормалізація значень тривалості програмних проектів та трудомісткості проектів з розробки програмних проектів виконується з використанням нормалізуючого перетворення, то рівняння нелінійної регресії тривалості програмних проектів в залежності від програмних проектів визначається як

$$\hat{D}(E) = z'_D\left(\hat{z}_D(z_E)\right), \qquad (2.10)$$

де $\hat{z}_D(z_E) = b_0 + b_1 z_E$; $z_E = z(E)$;

z'_D - перетворення, зворотнє до вибраного перетворення для нормалізації випадкової величини D

b_0 і b_1 - коефіцієнти рівняння лінійної регресії (2.6) для нормалізованих значень.

Зважаючи на (2.10), нелінійна регресійна модель для оцінювання тривалості програмних проектів в залежності від трудомісткості задається як

$$D(E) = \hat{D}(E) + \varepsilon, \tag{2.11}$$

де ε – це випадкова похибка.

Зважаючи на (2.8), довірчий інтервал [$\overline{\mathrm{D}}$(E)] нелінійного регресійного рівняння (2.10) задається як

$$\left[\overline{D}(E)\right] = z'_D\left(\left[\bar{z}_D(z_E)\right]\right), \tag{2.12}$$

де $\left[\bar{z}_D(z_E)\right]$ задається як (2.8).

Зважаючи на (2.9), інтервал прогнозування [D(E)] нелінійного регресійного рівняння (2.10) задається як

$$\left[D(E)\right] = z'_D\left(\left[z_D(z_E)\right]\right), \tag{2.13}$$

де $\left[\mathrm{z_D(z_E)}\right]$ задається як (2.9).

Щоб порівняти існуючу і вдосконалені моделі можна використати суму квадратів відхилень між оцінками тривалості за рівнянням регресії та емпіричними даними, коефіцієнт детермінації, відносну похибку за нелінійною регресійною моделлю [7].

Знаходження суми квадратів відхилень між оцінками тривалості за рівнянням регресії та емпіричними даними:

$$SSE = \sum_{i=1}^{n}\left(D_i - \hat{D}(E_i)\right)^2, \tag{2.14}$$

де n - це кількість емпіричних значень;

D_i - фактичне значення тривалості, а $\hat{D}(E_i)$ - оцінка випадкової величини D.

Рівняння регресії є кращим, якщо для нього значення SSE є меншим.

Знаходження значення коефіцієнту детермінації R^2:

$$R^2 = 1 - \frac{\sum_{i=1}^{n} \left(D_i - \hat{D}(E_i)\right)^2}{\sum_{i=1}^{n} \left(D_i - \overline{D}\right)^2}, \qquad (2.15)$$

де n - це кількість емпіричних значень;

D_i - фактичне значення, а $\hat{D}(E_i)$ - оцінка випадкової величини D;

$\overline{D}$ - середнє значення випадкової величини D, $\overline{D} = \frac{1}{n}\sum_{i=1}^{n} D_i$.

Рівняння регресії є кращим, якщо для нього значення коефіцієнту детермінації R^2 є більшим.

Знаходження відносної похибки за нелінійною регресійною моделлю задається як

$$\delta = \frac{1}{2}\left|\frac{\overline{D}(E_i)_{\max} - \overline{D}(E_i)_{\min}}{\hat{D}(E_i)}\right| \cdot 100\%,$$

де $\overline{D}(E_i)_{\min}$ – це нижнє значення довірчого інтервалу (2.12) нелінійної регресії;

$\overline{D}(E_i)_{\max}$ – верхнє значення довірчого інтервалу (2.12) нелінійної регресії;

$\hat{D}(E_i)$ - оцінка значення випадкової величини D за рівнянням регресії.

Модель є достовірнішою, якщо для неї значення δ є меншим.

2.2 Побудова нелінійної регресійної моделі для оцінювання тривалості програмних проектів для mainframe

Нехай випадкова величина D – це емпіричні значення тривалості програмних проектів, а випадкова величина E – це емпіричні значення

трудомісткості програмних проектів для платформи mainframe з бази даних ISBSG (215 проектів).

Після того як було виконано нормалізацію випадкових величин D та E, за отриманими нормалізованими випадковими величинами z_D та z_E будуємо лінійну регресійну модель тривалості для mainframe. Обчислені за (2.7) коефіцієнти регресійної моделі мають наступні значення: $b_0 = -0{,}245$ і $b_1 = 0{,}354$. При побудові довірчого інтервалу (2.8) використовуване значення t-розподілу Стьюдента було $t_{(0{,}05/2),(205-2)} = 1{,}972$. Рівняння лінійної регресії (2.6) з вказаними коефіцієнтами та рівняння 95% інтервалу прогнозування регресії наведено на рисунку 2.1.

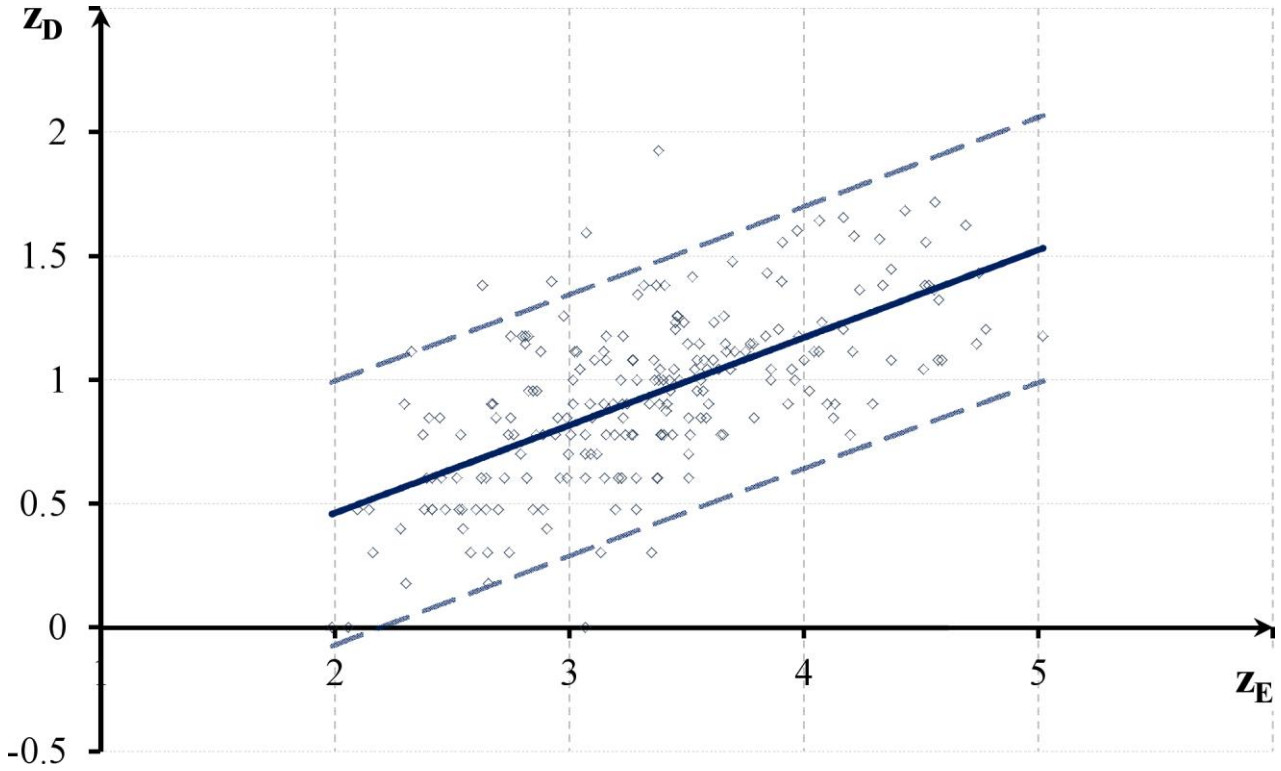

Рисунок 2.1 – Рівняння лінійної регресії та рівняння 95% інтервалу прогнозування лінійної регресії для нормалізованих значень тривалості програмних проектів для mainframe

Після побудови частину емпіричних значень було вилучено через те, що вони виходили за межі інтервалу прогнозування лінійної регресії для нормалізованих значень. Далі знову ітеративно будуємо лінійну регресійну

модель і вилучаємо точки, які виходять за межі інтервалу прогнозування, допоки після побудови лінійної регресійної моделі не будуть відсутні викиди.

Для цієї моделі обчислені за (2.7) коефіцієнти регресійної моделі мають наступні значення: b_0 = -0,131 і b_1 = 0,311. При побудові довірчого інтервалу (2.8) використовуване значення t-розподілу Стьюдента було $t_{(0,05/2),(205-2)}$ = 1,98. Рівняння лінійної регресії (2.6) з вказаними коефіцієнтами та рівняння 95% інтервалу прогнозування регресії наведено на рисунку 2.2.

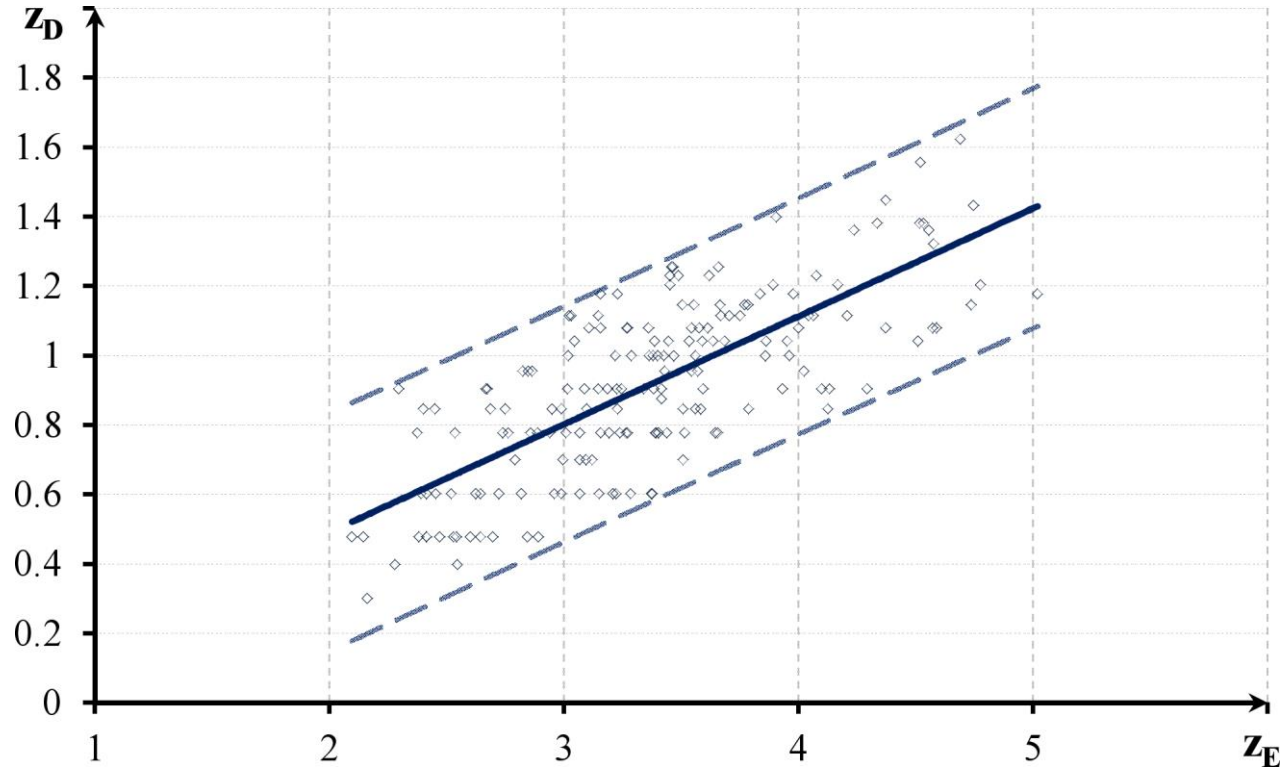

Рисунок 2.2 – Рівняння лінійної регресії та рівняння 95% інтервалу прогнозування лінійної регресії для нормалізованих значень тривалості програмних проектів для mainframe після видалення викидів

На наступному кроці, за (2.10), використовуючи значення обчислених коефіцієнтів рівняння лінійної регресії, виконуємо побудову нелінійної регресійної моделі для оцінювання тривалості програмних проектів для mainframe в залежності від трудомісткості розробки цих проектів. За (2.12), використовуючи визначене раніше значення t-розподілу Стьюдента, будуємо рівняння 95% довірчого інтервалу нелінійної регресії тривалості програмних проектів для mainframe в залежності від трудомісткості. За (2.13), використовуючи визначене

раніше значення t-розподілу Стьюдента, будуємо рівняння 95% інтервалу прогнозування нелінійної регресії тривалості програмних проектів для mainframe в залежності від трудомісткості розробки цих програмних проектів.

Побудована нелінійна регресійна модель для оцінювання тривалості програмних проектів для mainframe в залежності від трудомісткості розробки програмних проектів та рівняння інтервалу прогнозування нелінійної регресії наведено на рисунку 2.3. По цьому рисунку видно, що емпіричні значення знаходяться в середині інтервалу прогнозування нелінійної регресії.

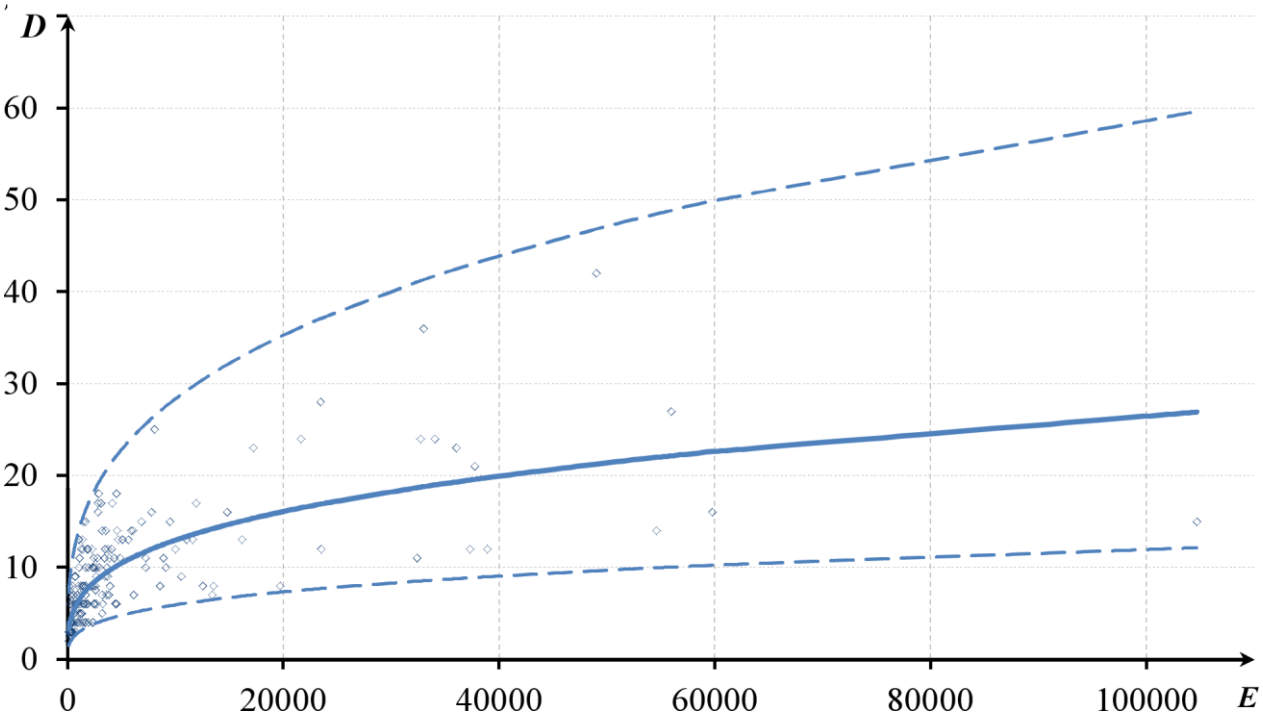

Рисунок 2.3 – Нелінійна регресійна модель для оцінювання тривалості програмних проектів для mainframe:

◦ - емпіричні дані (E – трудомісткість проекту; D – тривалість робіт)

— - нелінійна регресія;

— - 95% довірчий інтервал нелінійної регресії;

– – - 95% інтервал прогнозування нелінійної регресії

Побудовані рівняння нелінійної регресії тривалості програмних проектів для mainframe мається наступні характеристики: значення суми квадратів відхилень між оцінками тривалості за рівнянням регресії та емпіричними даними

тривалості (2.14) дорівнює $SSE = 3363$; коефіцієнт детермінації (2.15) дорівнює $R^2 = 0{,}4887$; відносна похибка за нелінійною регресійною моделлю $\delta = 34\%$.

Характеристики моделі ISBSG для оцінювання тривалості програмних проектів для mainframe є наступними: значення суми квадратів відхилень між оцінками тривалості за рівнянням регресії та емпіричними даними тривалості (2.14) дорівнює $SSE = 3592$; коефіцієнт детермінації (2.15) дорівнює $R^2 = 0{,}4539$; відносна похибка за нелінійною регресійною моделлю $\delta = 35\%$.

За цими характеристиками, побудована модель для оцінювання тривалості програмних проектів для mainframe є кращою, ніж модель ISBSG (на 6,38% за SSE, на 7.67% за R^2 та на 2.83% за δ).

3 РОЗРОБКА ПРОЕКТУ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ ПРОГРАМНИХ ПРОЕКТІВ

3.1 Розробка ескізного проекту програми для оцінювання тривалості програмних проектів

Моделювання варіантів використання

Даний проект передбачає наявність лише одного актора – користувача програми, який буде повноцінно використовувати всі функції програмного продукту.

Виявлені варіанти використання знаходяться в таблиці 3.1.

Таблиця 3.1 – Виявлені варіанти використання

Основні актори	**Найменування**	**Формулювання**
Користувач	Внести назву проекту	Вносить інформацію про назву проекту
Користувач	Внести трудомісткість проекту	Вносить інформацію про трудомісткість проекту
Користувач	Внести платформу проекту	Вносить інформацію платформу проекту
Користувач	Виконати оцінювання тривалості	Виконує оцінювання тривалості на основі нелінійної регресії та її інтервалу передбачення
Користувач	Зберегти результати оцінювання тривалості	Виконує збереження результатів оцінювання тривалості в базу даних

Користувач	Переглянути результати попередніх оцінювань тривалості програмних проектів	Перегляд збережених результатів оцінювання тривалості програмних проектів

Конкретизуємо основні варіанти використання:

1. *Найменування:* Внести дані про проект

Основна дійова особа: Користувач

Зв'язок з іншими варіантами використання: Узагальнює прецеденти «Внести назву проекту», «Внести трудомісткість проекту», «Внести платформу проекту».

Короткий опис:

Варіант використання «Внести дані про проекти» відображає три варіанти операцій і дає можливість внести інформацію про проекти у відповідні поля. Для варіанту «Внести назву проекту» – це назву проекту, для варіанту «Внести трудомісткість проекту» – це трудомісткість проекту, для варіанту «Внести платформу проекту» – це вибір платформи проекту.

Потоки подій: Прецедент починається, коли у вікні програми користувач обирає комірку, яку необхідно заповнити даними.

Базовий потік – Внесення даних про проекти:

1) Користувач запускає програму;

2) Користувач заповнює поле «Назва проекту» назвою проекту;

3) Користувач заповнює поле «Трудомісткість» значенням трудомісткості проекту;

4) Користувач вибирає платформу проекту з випадаючого списку.

Альтернативі потоки:

1) Якщо у п. 2 внести значення, яке є меншим за 0 чи дорівнює 0, то буде виведено повідомлення про помилку та пропозиція повторити внесення даних. Внесене число не буде збережене.

Спеціальні вимоги: значення трудомісткості проекту повинно завжди бути цілим додатнім числом.

2. *Найменування:* Виконати оцінювання тривалості

Основна дійова особа: Користувач

Зв'язок з іншими варіантами використання: відсутній.

Короткий опис:

Варіант використання «Виконати оцінювання тривалості» дозволяє отримати Оцінку тривалості, Довірчий інтервал нелінійної регресії, Інтервал прогнозування тривалості програмних проектів

Потоки подій: Прецедент починається, коли користувач натискає кнопку «Оцінити тривалість».

Базовий потік – Виконати оцінювання тривалості:

1) Користувач натискає кнопку «Оцінити тривалість»;

2) Результати оцінювання тривалості програмних проектів (Оцінка тривалості, Довірчий інтервал нелінійної регресії, Інтервал прогнозування) будуть показані в таблиці.

3. *Найменування:* Зберегти результати оцінювання

Основна дійова особа: Користувач

Зв'язок з іншими варіантами використання: відсутній.

Короткий опис:

Варіант використання «Зберегти результати оцінювання» дозволяє збереження результатів оцінювання тривалості в базу даних.

Потоки подій: Прецедент починається, коли користувач натискає кнопку «Зберегти оцінку».

Базовий потік – Збереження результатів оцінювання тривалості:

1) Користувач натискає кнопку «Зберегти оцінку»;

2) Результати оцінювання тривалості (Оцінка тривалості, Довірчий інтервал нелінійної регресії, Інтервал прогнозування) зберігаються в базу даних.

4. *Найменування:* Переглянути результати попередніх оцінювань

Основна дійова особа: Користувач

Зв'язок з іншими варіантами використання: відсутній.

Короткий опис:

Варіант використання «Переглянути результати попередніх оцінювань» дозволяє переглянути результати попередніх оцінювань тривалості програмних проектів (Оцінка тривалості, Довірчий інтервал нелінійної регресії, Інтервал прогнозування).

Діаграма варіантів використання зображена на рисунку 3.1.

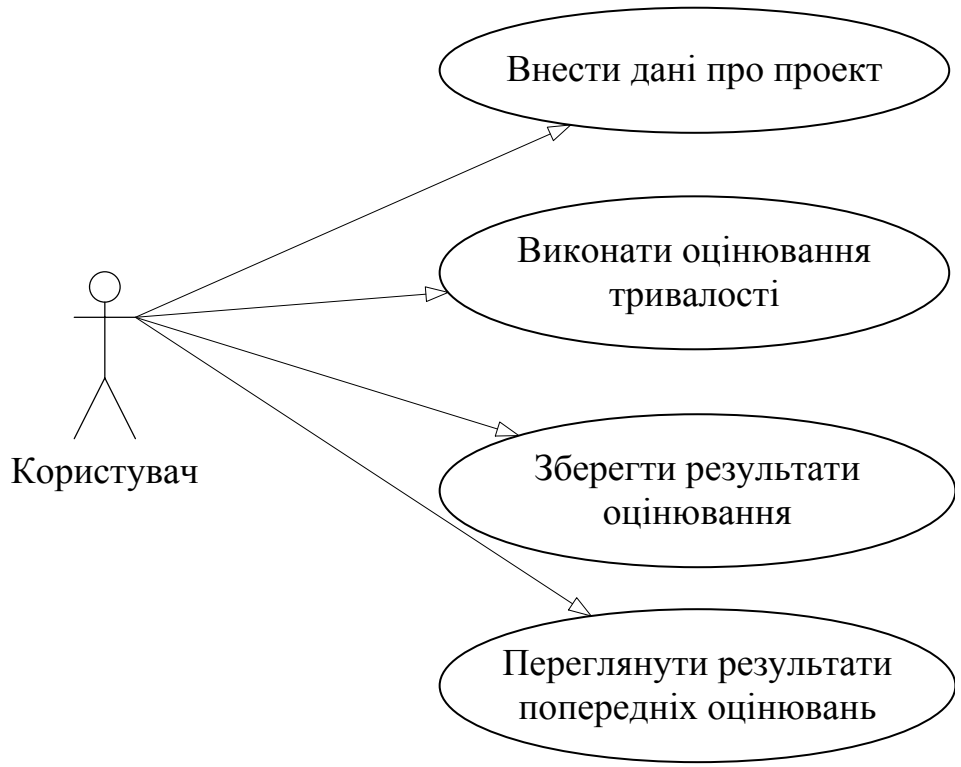

Рисунок 3.1 – Діаграма варіантів використання програми для оцінювання тривалості програмних проектів

Діаграма станів показана на рисунку 3.2. Основним станом даної комп`ютерної програми є відображення головного вікна, під час якого відбувається очікування дій користувача. Перехід до інших станів відбувається в залежності від виконаних дій користувача, що відображено на рисунку 3.2.

Рисунок 3.2 – Діаграма станів програми для оцінювання тривалості програмних проектів

Моделювання потоків даних

Діаграма потоків даних в нотації Гейна-Сарсона показана на рисунку 3.3.

Складні потоки даних діаграми 3.3. мають наступні атрибути:

параметри нелінійної регресійної моделі – параметри b_0, b_1, $s^2_{z_D}$, $\bar{z}_E$, S_{z_E}, n моделей (2.12), (2.13);

- довірчий інтервал тривалості проекту з розробки програмного забезпечення – нижня границя довірчого інтервалу тривалості $[\overline{D}(E)]$ (2.12), верхня границя довірчого інтервалу тривалості $[\overline{D}(E)]$ (2.12);
- інтервал прогнозування тривалості проекту з розробки програмного забезпечення – нижня границя інтервалу прогнозування тривалості [D(E)] (2.13), верхня границя довірчого інтервалу тривалості [D(E)] (2.13).

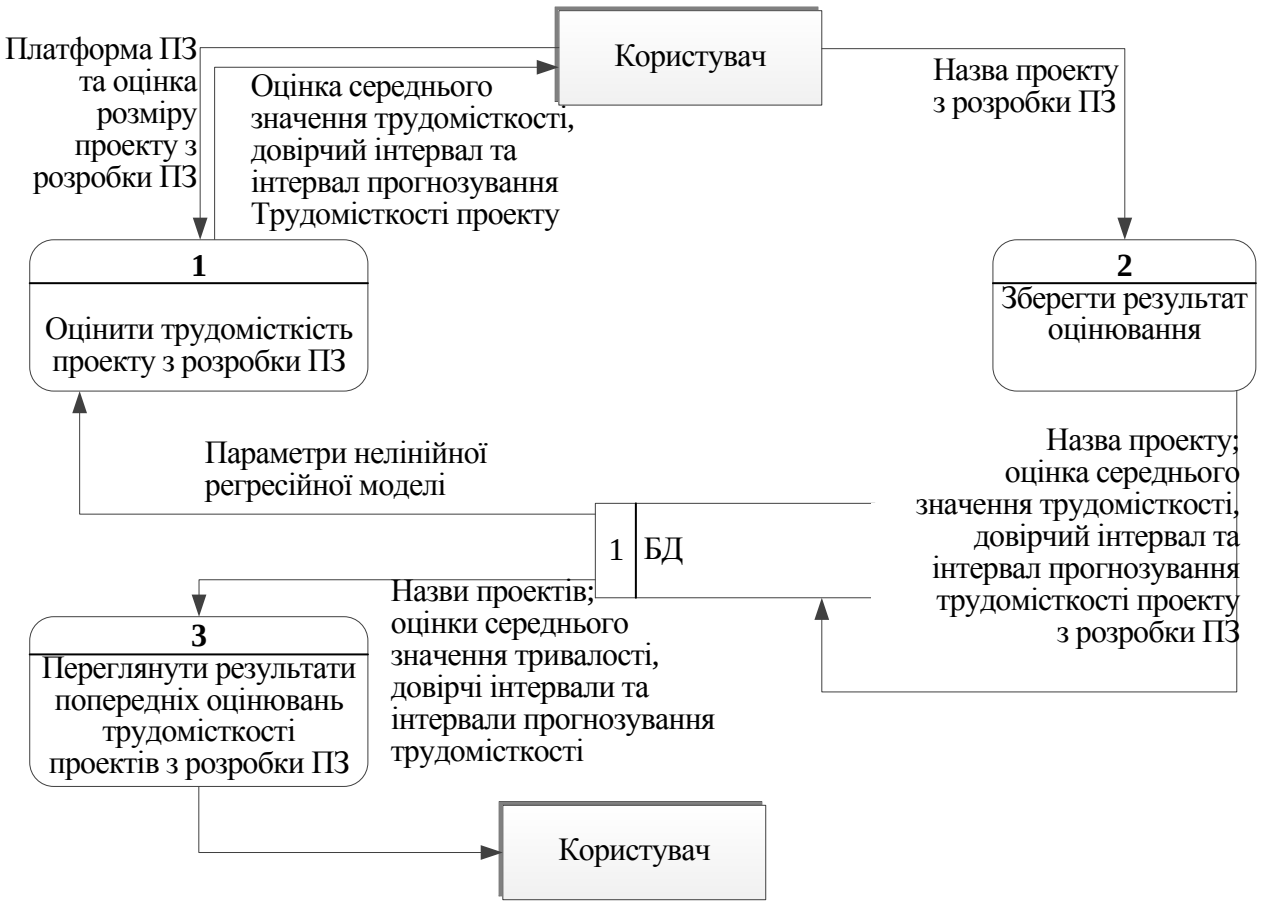

Рисунок 3.3 – Діаграма потоків даних програми для оцінювання тривалості програмних проектів

3.2 Розробка технічного проекту програми для оцінювання тривалості програмних проектів

Логічна модель бази даних

Логічна модель таблиці параметрів математичних моделей тривалості програмних проектів для різних платформ:

- ідентифікатор запису – числовий тип даних, первинний ключ;
- назва платформи – строковий тип даних, заборонені дублювання;;
- код платформи – строковий тип даних, заборонені дублювання;
- параметр b_0 моделей (2.10), (2.12), (2.13) – числовий тип даних;
- параметр b_1 моделей (2.10), (2.12), (2.13) – числовий тип даних;
- параметр n моделей (2.12), (2.13) – числовий тип даних;
- параметр $s_{z_D}^2$ моделей (2.12), (2.13) – числовий тип даних;
- параметр $\bar{z}_E$ моделей (2.12), (2.13) – числовий тип даних;
- параметр S_{z_E} моделей (2.12), (2.13) – числовий тип даних.

Логічна модель таблиці результатів оцінювань та моделювань тривалості програмних проектів для різних платформ:

- ідентифікатор запису – числовий тип даних, первинний ключ;
- назва проекту – строковий тип даних, заборонені дублювання;
- код платформи – строковий тип даних;
- трудомісткість – числовий тип даних;
- тривалість – числовий тип даних;
- нижня границя інтервалу прогнозування тривалості – числовий тип даних;
- верхня границя інтервалу прогнозування тривалості – числовий тип даних;
- нижня границя довірчого інтервалу тривалості – числовий тип даних;
- верхня границя довірчого інтервалу тривалості – числовий тип даних;
- дата оцінювання – дата.

3.3 Розробка робочого проекту програми для оцінювання тривалості програмних проектів

Вибір мови та середовища програмування

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

Критерії, які використовувалися для вибору мови програмування:

- можливість роботи програми для оцінювання тривалості програмних проектів в різних операційних системах;
- доступність компонентів для формування ергономічного візуального інтерфейсу;
- існування математичних бібліотек, необхідних для виконання оцінювання тривалості програмних проектів з використанням побудованої моделі;
- існування драйверів для підключення до СКБД SQLite;
- можливість доступу до СКБД, використовуючи одну з реалізацій Java Persistence API – специфікації прикладного програмного інтерфейсу, яка описує роботу з реляційними СКБД в Java-застосунках;
- можливість переключення програми для оцінювання тривалості програмних проектів для роботи з іншими СКБД завдяки використанню для доступу до бази даних мови запитів Java Persistence Query Language – платформо-незалежної об'єктно-орієнтованої мови запитів, що є частиною специфікації Java Persistence API і дозволяє не змінювати запити при зміні системи управління базами даних;
- доступність кількох безкоштовних IDE (Integrated Development Environment – інтегрованих середовищ розробки прикладних програм), що дозволить скоротити час розробки програмного забезпечення;
- можливість безкоштовного використання і розповсюдження Java разом з програмою для комерційних проектів.

Фізична модель бази даних

Систему управління базами даних SQLite було обрано для організації доступу до бази даних. Всі данні СУБД SQLite зберігаються в одному бінарному файлі. Використання цієї СУБД є найбільш оптимальним, тому що в базі даних не потрібно зберігати велику кількість даних. СУБД SQLite не потребує запуску серверу, що найкраще підходить при використанні бази даних на одному комп'ютері. При необхідності доступу з різних комп'ютерів, СУБД SQLite можна буде змінити на серверну СУБД (MySQL чи PostgreSQL) не вносячи зміни в запити, так як СУБД SQLite використовує мову запитів SQL.

Фізична модель таблиці параметрів нелінійних регресійних моделей тривалості програмних проектів ключає наступні поля:

- id – INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT;
- title – TEXT NOT NULL UNIQUE;
- platform – TEXT NOT NULL UNIQUE;
- b0 – NUMERIC NOT NULL;
- b1 – NUMERIC NOT NULL;
- n – INTEGER NOT NULL;
- zd_sse – NUMERIC NOT NULL;
- ze_avg – NUMERIC NOT NULL;
- ze_S – NUMERIC NOT NULL.

Фізична модель таблиці результатів оцінювань та моделювань тривалості програмних проектів ключає наступні поля:

- id – INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT;
- title – TEXT NOT NULL UNIQUE;
- platform – TEXT NOT NULL;
- size – NUMERIC NOT NULL;
- effort – NUMERIC NOT NULL;
- effort_estimation_interval_bottom – NUMERIC NOT NULL;
- effort_estimation_interval_top – NUMERIC NOT NULL;
- effort_confidence_interval_bottom – NUMERIC NOT NULL;
- effort_confidence_interval_top – NUMERIC NOT NULL;

– created – DATE NOT NULL.

Використаємо UML (Unified Modeling Language – стандартну мову для написання моделі аналізу, проектування і реалізації об'єктно-орієнтованих програмних систем) для відображення структури класів програми.

Діаграма класів програмного забезпечення «Оцінювання тривалості програмних проектів» показана на рисунку 3.4.

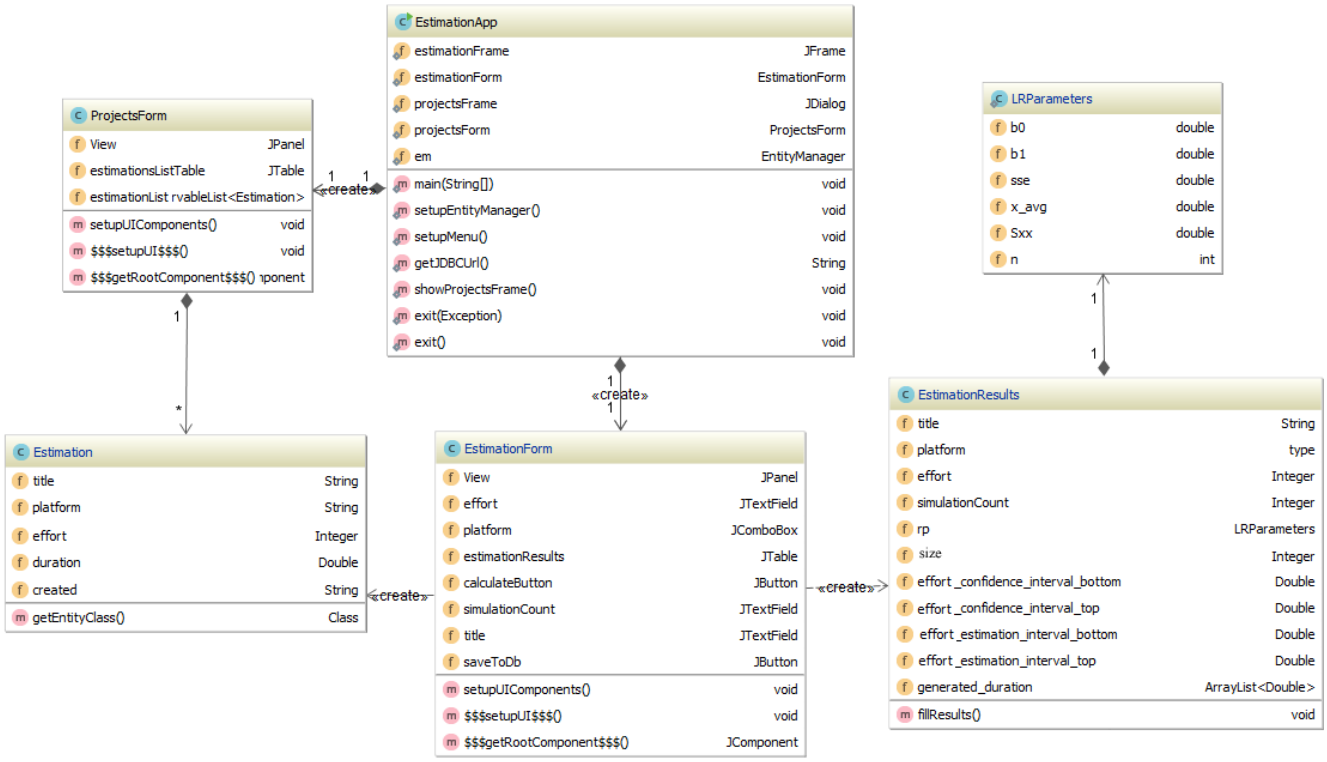

Рисунок 3.4 – Діаграма класів програмного забезпечення «Оцінювання тривалості програмних проектів»

На початку роботи програми виконується метод EstimationApp:: main(), в якому створюється підключення до СУБД, а потім створюється і відображається екземпляр головного вікна програми. Також до головного вікна програми додається пункт меню для створення і відображення вікна з таблицею збережених попередніх результатів оцінювання тривалості програмних проектів. Екземпляр класу EstimationResults, який створюється в головному вікні, реалізує оцінювання тривалості програмних проектів.

Код класу EstimationResults є наступним:

```
package EstimationApp.forms;
```

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

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

public class EstimationResults extends DefaultTableModel {

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

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

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

     public void fillResults() {
          setRowCount(0);

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

          double e_z = Math.log10(size.doubleValue());
          double t_z_estimate = LinearRegression.estimateY(e_z,
rp.b0, rp.b1);
          double t_z_estimate_l1 =
LinearRegression.estimateYConfidence(e_z, rp.b0, rp.b1, false,
false, rp.sse, rp.x_avg, rp.Sxx, rp.n, tStudent);
          double t_z_estimate_r1 =
LinearRegression.estimateYConfidence(e_z, rp.b0, rp.b1, true, false,
rp.sse, rp.x_avg, rp.Sxx, rp.n, tStudent);
          double t_z_estimate_l2 =
LinearRegression.estimateYConfidence(e_z, rp.b0, rp.b1, false, true,
rp.sse, rp.x_avg, rp.Sxx, rp.n, tStudent);
          double t_z_estimate_r2 =
LinearRegression.estimateYConfidence(e_z, rp.b0, rp.b1, true, true,
rp.sse, rp.x_avg, rp.Sxx, rp.n, tStudent);
```

```
            effort = MathUtil.round(Math.pow(10, t_z_estimate), 1);
            effort_confidence_interval_bottom =
MathUtil.round(Math.pow(10, t_z_estimate_l1), 1);
            effort_confidence_interval_top =
MathUtil.round(Math.pow(10, t_z_estimate_r1), 1);
            effort_estimation_interval_bottom =
MathUtil.round(Math.pow(10, t_z_estimate_l2), 1);
            effort_estimation_interval_top =
MathUtil.round(Math.pow(10, t_z_estimate_r2), 1);


            addRow(new String[]{"Назва", this.title});
            addRow(new String[]{"Платформа",
this.platform.toString()});
            addRow(new String[]{"Трудомісткість", size.toString() + "
функціональних точок"});

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

Інструкція цієї комп`ютерної програми приведена в додатку В.

Випробування програми для оцінювання тривалості програмних проектів

Тестування програми – це процес виконання ПЗ з метою виявлення помилок. Кроки процесу задаються тестами.

Існує два принципи тестування програми:

– функціональне тестування (тестування "чорного ящика");

– структурне тестування (тестування "білого ящика").

При тестуванні "чорного ящика" розглядаються системні характеристики ПЗ, ігнорується їх внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Об’єктом тестування "білого ящика" є не зовнішня, а внутрішня поведінка програмного забезпечення. Перевіряється коректність побудови всіх елементів ПЗ та правильність їх взаємодії. Зазвичай аналізуються управляючі зв’язки елементів, інколи – інформаційні зв’язки. Тестування за цим принципом характеризується ступінню, в якій тести виконують або покривають логіку (вхідний текст) програми. Вичерпне тестування також викликає певні труднощі.

У результаті опрацювання специфікації до програмного забезпечення, що підлягає тестуванню було отримано ряд тестів «чорної скриньки». Результати тестування за принципом "чорного ящика" наведені в таблиці 3.2.

Таблиця 3.2 – Результати тестування за принципом "чорного ящика"

Тест	**Вхідні значення**	**Результат**
Ведення трудомісткості програмного проекту	Значення введено невірно	При намаганні виконати оцінювання тривалості виводиться помилка
	Значення введено вірно	Стає можливим побачити результати оцінювання тривалості
	Значення введено невірно	З’являється вікно з попередженням про неправильно введені дані
Ведення назви програмного проекту	Значення введено вірно	Стає можливим зберегти результати оцінювання тривалості під даною назвою програмного проекту
	Значення не введено	При намаганні зберегти результати оцінювання тривалості виводиться помилка

В результаті тестування за принципом "білого ящика" була перевірена коректність побудови всіх елементів програмного забезпечення та правильність їх

взаємодії. Випробування програмного забезпечення було виконано згідно із програмою і методикою випробувань. Випробування показали, що розроблене програмне забезпечення повністю відповідає поставленим задачам.

4 РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ

В результаті дослідження було *удосконалено* нелінійну регресійну модель ISBSG для оцінювання тривалості програмних проектів для mainframe в залежності від трудомісткості розробки цих програмних проектів *за рахунок* побудови рівнянь границь довірчого інтервалу та рівнянь границь інтервалу прогнозування нелінійної регресії, *що дозволяє* підвищити достовірність оцінювання тривалості програмних проектів для mainframe.

На основі побудованої нелінійної регресійної моделі в ході виконання роботи було розроблено програмне забезпечення «Оцінювання тривалості програмних проектів» для оцінювання тривалості програмних проектів. Вигляд головного вікна програми показаний на рисунку 4.1.

Розроблене програмне забезпечення має наведені нижче функції:

- Оцінювання середнього значення тривалості програмного проекту з використанням нелінійної регресійної моделі в залежності від трудомісткості розробки;
- Оцінювання нижньої границі 95% інтервалу прогнозування тривалості програмного проекту;
- Оцінювання верхньої границі 95% інтервалу прогнозування тривалості програмного проекту;
- Оцінювання нижньої границі 95% довірчого інтервалу середнього значення тривалості програмного проекту;
- Оцінювання верхньої границі 95% довірчого інтервалу середнього значення тривалості програмного проекту;
- Збереження результатів оцінювання програмного проекту в базу даних;
- Перегляд результатів попередніх оцінювань тривалості програмних проектів.

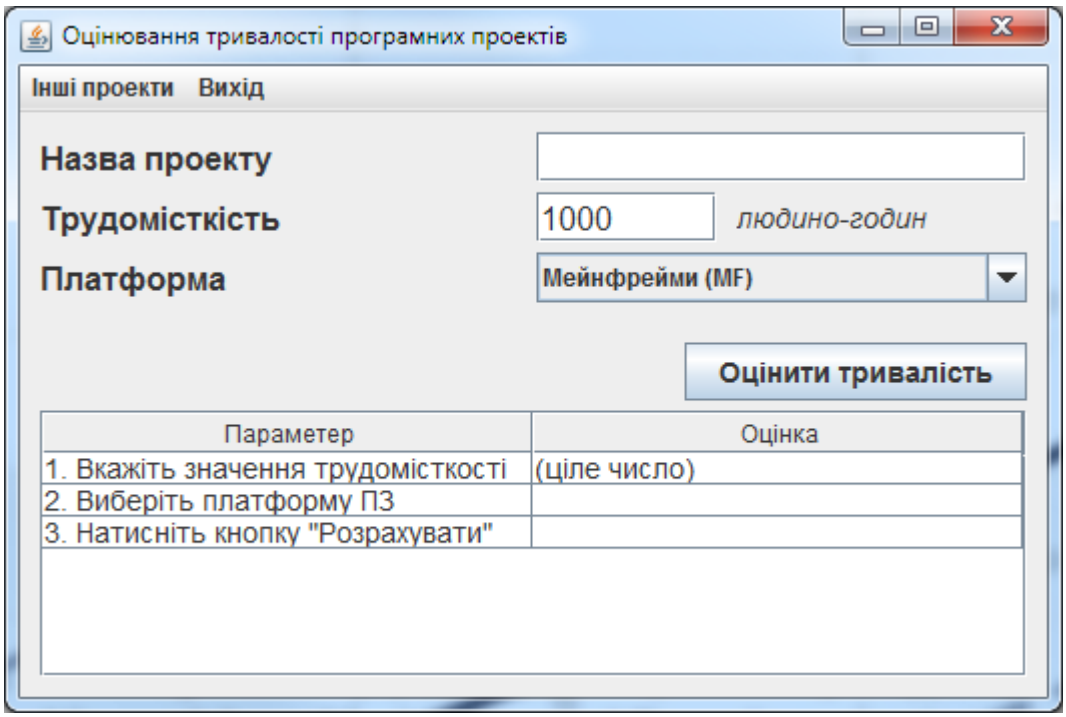

Рисунок 4.1 – Головне вікно програмного забезпечення "Оцінювання тривалості програмних проектів"

Програмне забезпечення реалізовано з використанням Netbeans та мови програмування Java.

Розроблене програмне забезпечення може працювати на будь-якому комп’ютері з операційною системою, в якій встановлена Java Runtime Environment (JRE) версії не нижче Java 1.8.

В процесі виконання програми відбуваються наступні дії:

1. Параметри нелінійної регресійної моделі для оцінювання тривалості програмних проектів в залежності від трудомісткості розробки (2.10) для вибраної платформи завантажуються з бази даних.
2. Параметри рівнянь нижньої та верхньої границь довірчого інтервалу (2.12) нелінійної регресії тривалості програмних проектів в залежності від трудомісткості розробки для вибраної платформи завантажуються з бази даних.
3. Параметри рівнянь нижньої та верхньої границь інтервалу прогнозування (2.13) нелінійної регресії тривалості програмних проектів в залежності від

трудомісткості розробки для вибраної платформи завантажуються з бази даних.

4. Використовуючи нелінійну регресійну модель для оцінювання тривалості програмних проектів в залежності від трудомісткості, розраховується та відображається на екрані оцінка середнього значення тривалості програмного забезпечення.
5. Використовуючи рівняння нижньої та верхньої границь довірчого інтервалу (2.12) нелінійної регресії тривалості програмних проектів в залежності від трудомісткості розробки, обчислюються та виводяться на екран значення нижньої і верхньої границь 95% довірчого інтервалу отриманої оцінки середнього значення тривалості.
6. Використовуючи рівняння нижньої та верхньої границь інтервалу прогнозування (2.13) нелінійної регресії тривалості програмних проектів в залежності від трудомісткості розробки, обчислюються та виводяться на екран значення нижньої і верхньої границь 95% інтервалу прогнозування.
7. Якщо результати оцінювання тривалості необхідно зберегти в базу даних, користувач має натиснути на кнопку «Зберегти оцінку», тоді результати оцінювання тривалості програмного проекту буде збережено в базу даних.
8. Вибравши пункт меню «Вихід» вверху головного вікна програми або натиснувши кнопку закриття головного вікна програми, яка знаходиться серед елементів управління вікном операційної системи, користувач може закрити програму.

При перегляді результатів оцінювання виконується наступна послідовність дій:

1. Завантажуються з бази даних збережені результати оцінювання тривалості програмних проектів.
2. Відображається вікно перегляду збережених результатів оцінювання тривалості з завантаженими даними (рисунок 4.3).
3. У відкритому вікні програми користувач переглядає збережені результати оцінювання тривалості програмних проектів.

Тестування даного програмного продукту показало, що програмне забезпечення успішно виконує поставлені перед ним завдання.

Була розроблена наступна програмна документація: технічне завдання (додаток А); опис програми (додаток Б); інструкція (додаток В); програма та методика випробувань (додаток Г).

Результати випробувань програми наведені на рисунках 4.2-4.3

Після введення назви проєкту, тривалості, вибору платформи та натискання кнопки «Оцінити тривалість», результати оцінювання тривалості програмних проектів (Оцінка тривалості, Довірчий інтервал нелінійної регресії, Інтервал прогнозування) були відображені в таблиці в нижній частині вікна програми (рисунок 4.2).

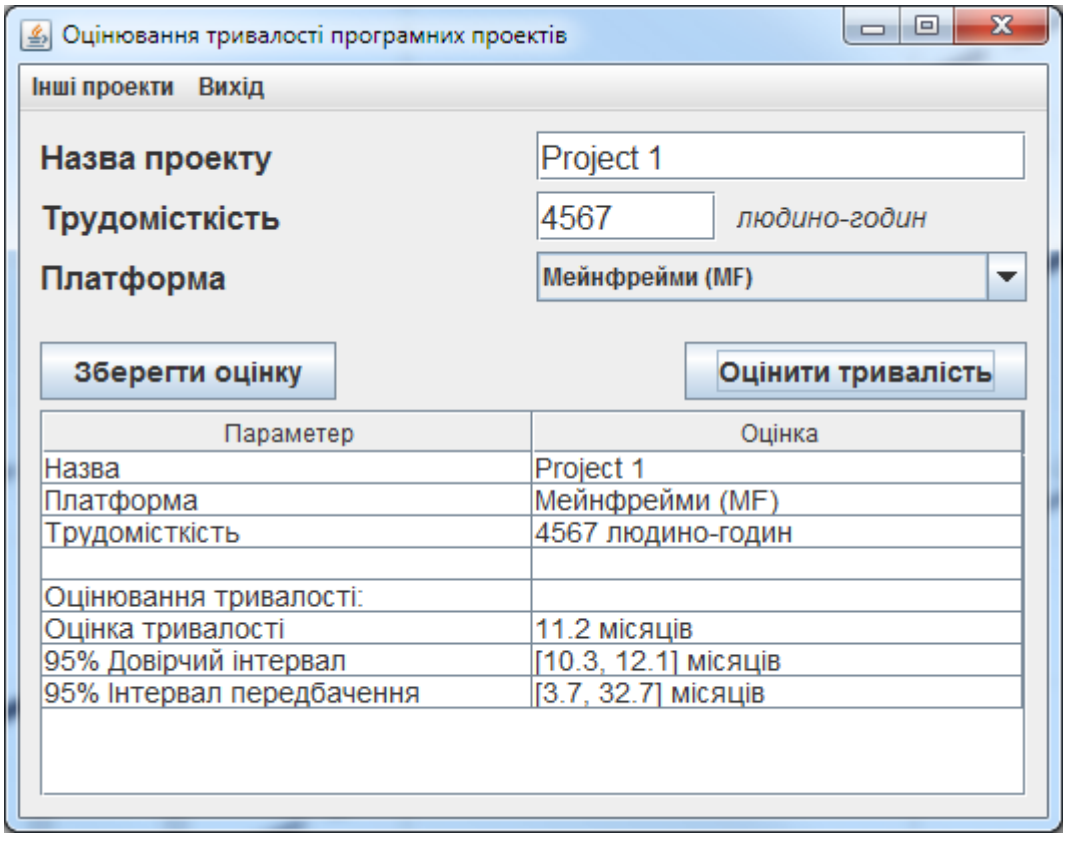

Рисунок 4.2 – Результати оцінювання тривалості програмного проекту

Результати оцінювання декількох проектів зберігаються в базу даних натисканням на кнопку «Зберегти оцінку». Для перегляду раніше збережених

результатів оцінювання вибрано пункт меню «Інші проекти» в головному меню програми.

Вікно перегляду збережених результатів оцінювання тривалості показано на рисунку 4.3.

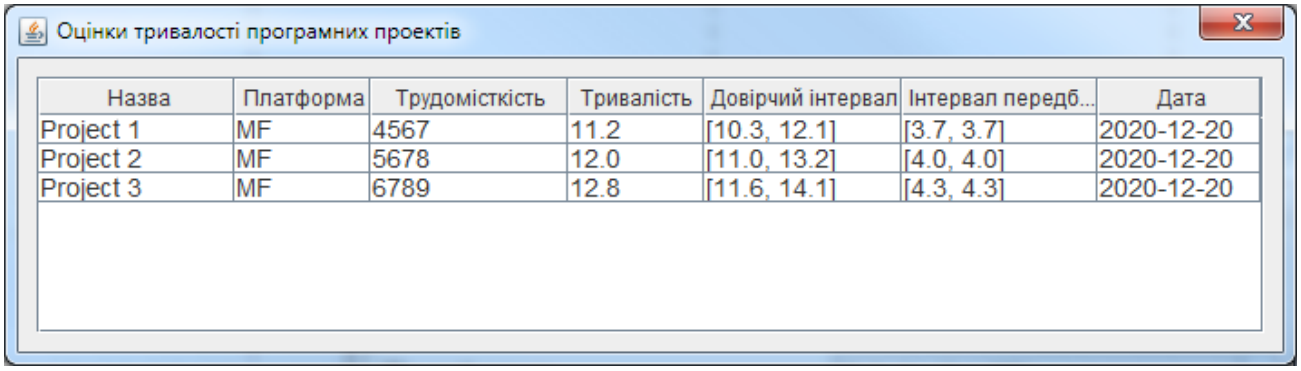

Рисунок 4.3 – Вікно перегляду збережених результатів оцінювання тривалості програмних проектів

5 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВІД РОЗРОБКИ ТА ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Вступ

Дана робота присвячена розробці програмного забезпечення для оцінювання тривалості. Його можна використовувати в галузі розробки та забезпечення якості ПЗ, а також управління програмними проектами.

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

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

5.1 Розрахунок витрат на створення й експлуатацію програмного забезпечення

Витрати на розробку системи складаються з витрат:

- на заробітну платню розробника;

- на амортизацію ЕОМ, на якій виконується розробка;

- на експлуатацію цієї ЕОМ;

- на засоби розробки;

- на матеріали і комплектуючі.

Розробка програмного забезпечення виконується програмістом, місячна заробітна плата якого складає 8000 грн.

Вартість сучасного ПК становить 15000 грн.

При вартості кіловат-години електроенергії 1.29 грн, розраховується вартість розробки програми.

Витрати на допоміжні матеріали наведені в табл. 5.1.

Таблиця 5.1 – Витрати на допоміжні матеріали

№	Пункти витрат	Сума, грн
1	Флеш-накопичувач даних	250
2	Папір для принтера	80
3	Картридж та тонер для принтера	200
	Разом	530

Вартість ПЗ знаходиться за формулою:

$$C_{пр} = (З_{зп} + З_{сз} + З_{зг} + З_{е}) * Т + З_{м} \quad (5.1)$$

де Т – тривалість розробки, міс.

Ззп – основна і додаткова заробітна плата обслуговуючого персоналу, грн.

Зсз – відрахування на соціальні заходи (15% від основної і додаткової заробітної плати), грн.

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

Зе – витрати на електроенергію, грн.

Зм – витрати на основні і допоміжні матеріали, грн.

Зе при споживанні потужності 0.5 кВт, тривалості роботи на місяць, рівної 22*8 = 176 годин, вартості кіловат-години електроенергії 1.29 грн становить:

$$З_е = 176 * 1.29 * 0.5 = 113.52 \text{ грн}$$

Витрати на розробку ПЗ наведені в табл. 5.2.

Таблиця 5.2 – Витрати на розробку програмного забезпечення

№	Найменування витрат	Одиниця	Кількість
1	Тривалість розробки (Т)	Міс.	3
2	Основна і додаткова заробітна плата (Ззп)	Грн.	12000.00
3	Відрахування на соціальні заходи (Зсз)	Грн.	1200.00
4	Загальногосподарські витрати (Ззг)	Грн.	800.00
5	Витрати на допоміжні матеріали (Зм)	Грн.	530.00
6	Витрати на електроенергію (Зе)	Грн.	113.52

За формулою 5.1 розрахуємо вартість програми:

$$С_{пр} = (12000 + 1200 + 800 + 113.52) * 3 + 530 = 42870.56 \text{ грн}$$

Амортизаційні відрахування складають 25% балансової вартості на рік:

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

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

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

Річний обсяг робіт ПК у годинах визначається в такий спосіб:

$$Ф_м = 264.5 * ТЗ \qquad (5.2)$$

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

Таким чином, річний обсяг роботи ПК складає:

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

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

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

Експлуатаційні витрати на ПК на рік складуть:

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

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

$$З_{р} = 42870.56 + 5523.62 = 48394.18 \text{ грн}$$

5.2 Економічна ефективність розробки і впровадження програмного забезпечення для оцінювання тривалості

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

Визначимо пряму економічну ефективність, ґрунтуючись на тому, що впровадження програмного продукту вивільняє 1 працівника (за експертною оцінкою фахівців підприємства).

Заробітна плата 1 працівника в рік складає:

$$З = 12000 * 12 * 1 = 144000 \text{ грн}$$

Річний економічний ефект розраховується за формулою:

$$E_{рік} = \Delta C_n – E_n * k \qquad (5.3)$$

де ΔC_n – вивільнені кошти після впровадження продукту (144000 грн) мінус експлуатаційні витрати (48394.18 грн) = 95605.82 грн;

E_n – коефіцієнт ефективності (дорівнює коефіцієнту амортизації 0.25);

k – одноразові витрати на впровадження продукту (42870.56 грн).

$$E_{рік} = 144000 – 0.25 * 42870.56 = 133282.36 \text{ грн}$$

Термін окупності ПЗ розраховується за формулою:

$$T = k / \Delta C_n = 42870.56 / 95605.82 = 0.45 \text{ року}$$

Отже, термін окупності ПЗ становить приблизно 5-6 місяців.

Висновки

У цьому розділі були здійснені розрахунки витрат, економічної ефективності та терміну окупності програми для оцінювання тривалості програмних проектів. Виходячи з розрахунків, впровадження даного програмного забезпечення окупить себе протягом 5-6 місяців. Таким чином, його розробка є економічно обґрунтованою.

6 ОХОРОНА ПРАЦІ

Вступ

Охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних та лікувально-профілактичних заходів та засобів, спрямованих на збереження здоров'я і працездатності людей в процесі праці (ст. 1 Закону «Про охорону праці»). Як правовий інститут охорона праці – це чималий комплекс правових норм. Можна визначити охорону праці також як систему забезпечення безпеки, збереження здоров'я і працездатності людини в процесі праці, засновану на сукупності законодавчих актів і відповідних їм соціально-економічних, технічних, гігієнічних і організаційних заходів.

Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці.

Власник або уповноважений ним орган повинен впроваджувати сучасні засоби техніки безпеки, які запобігають виробничому травматизму, і забезпечувати санітарно-гігієнічні умови, що запобігають виникненню професійних захворювань працівників.

На власника або уповноважений ним орган покладається систематичне проведення інструктажу (навчання) працівників з питань охорони праці, протипожежної охорони.

Охорона праці – один з центральних інститутів трудового права. Він має виключно практичне значення. Недодержання вимог охорони праці створює небезпеку для здоров'я і життя працівників. У свою чергу і ті, кого законодавець називає власником або уповноваженим ним органом, несуть сувору, у тому числі й кримінальну відповідальність за порушення правил охорони праці [24].

6.1 Аналіз шкідливих та небезпечних факторів у відділі забезпечення якості в офісі

Небезпечним називається виробничий фактор, вплив якого на працюючу людину в певних умовах приведе до травми або іншому раптовому різкому погіршенню здоров'я. Якщо ж виробничий фактор приведе до захворювання або зниження працездатності, то його вважають шкідливим. Залежно від рівня й тривалості впливу шкідливий виробничий фактор може стати небезпечним.

На офісного працівника при роботі впливають дві основні групи шкідливих факторів:

1) Фізичні, до яких відносять фактори, що породжуються безпосереднім впливом оточуючого середовища (електромагнітне випромінювання, освітленість робочого місця, температура повітря, шум).

2) Психофізіологічні, до яких відносять фізичне та нервово-психічне перевантаження організму (зорове напруження, статичні навантаження, нервово-психічне навантаження).

Освітлення – отримання, розподіл та використання світлової енергії для забезпечення нормальних умов праці. Світло, крім забезпечення фізичної можливості розрізняти предмети праці, також впливає на психічний стан людини та перебіг фізіологічних процесів в організмі. Раціонально влаштоване освітлення виробничих приміщень позитивно впливає на працівників, підвищує ефективність та безпеку праці, знижує втому та травматизм, забезпечує високу працездатність.

За вимогами санітарних норм освітлення повинно бути достатньо яскравим, рівномірним, не засліплювати очі та не створювати відблисків на робочій поверхні. За спектральним складом освітлення має наближатися до сонячного світла. Гігієнічними нормами вимагається максимально використовувати природне освітлення, оскільки денне світло краще сприймається органами зору. Штучне освітлення у виробничих та адміністративно-громадських приміщеннях має здійснюватись системою загального рівномірного освітлення,

допускають застосування системи комбінованого освітлення. Значення освітленості на поверхні робочого столу в зоні розміщення документів має становити 300-500лк. Як джерела світла для штучного освітлення мають застосовувати переважно люмінесцентні лампи типу ЛБ [25].

Одним з найнесприятливіших факторів, що зменшує працездатність та уважність робітників, створює передумови до травматизму та захворювань, є шум. Інтенсивний шум та вібрації негативно впливають не лише на слух, а і на нервову, серцево-судинну, та більшість інших систем організму, викликаючи зниження слуху, гіпертонію, психічні розлади тощо.

Гігієнічними нормативами (СН 3222-85, ГОСТ 12.1.003-85, ГР 2411-81) встановлені допустимі рівні звуку різних частот для робітників певних професій. У таблиці 6.1 наведені нормативи для людей, що працюють з ЕОМ.

Таблиця 6.1 – Допустимі рівні шуму

Вид трудової діяльності	Рівні звукового тиску в дБ октавних смугах із середньо геометричними частотами, ГЦ									
	31,5	63	125	250	500	1000	2000	4000	8000	Рівні звуку, еквівалентні рівні звуку, дБа/дБАекв
Програмісти ЕОМ	86	71	61	54	49	45	42	40	38	50
Оператори в залах обробки інформації на ЕОМ ті оператори комп’ютерного набору	96	83	74	68	63	60	57	55	54	65
В приміщеннях для розташування шумних агрегатів ЕОМ	103	91	83	77	73	70	68	66	64	75

Виходячи з даних норм, при проведенні комплексу заходів зі зменшення шуму усувається не будь-яка віброакустична активність обладнання, а тільки та, яка перевищує встановлений граничний рівень та шкідливо впливає на організм

людини. Усунення промислового шуму в першу чергу відбувається шляхом вдосконалення технологічних процесів та обладнання. При цьому в першу чергу усуваються найбільш сильні джерела шуму. Також необхідно знешумлювати усе обладнання, яке цього потребує, оскільки знешумлення окремих одиниць при наявності великої кількості обладнання, що продовжує випромінювати шум, недоцільно.

Самопочуття та працездатність людини також сильно залежать від теплового стану організму, на який впливають такі фізичні фактори виробничого середовища, як температура повітря, вологість, швидкість руху повітря, атмосферний тиск, доля кисню у повітрі тощо. Наприклад, якщо кількість кисню в повітрі зменшиться до 12% то утруднюється дихання. В таких умовах людина напружує дихальний апарат, дихає частіше; такий стан людина витримує до 0,5 години. Перегрівання організму відбувається за умов надлишкового конвективного випромінювання тепла нагрітих поверхонь. При перегріванні вступають в активну реакцію пристосувальні функції організму. При цьому активізується робота серцево-судинної та дихальної систем, відбувається інтенсивне потовиділення, котре сягає 5л за зміну. З потом втрачається велика кількість мінеральних солей та вітамінів. Вологість повітря суттєво впливає на терморегуляцію людського оргазму. Підвищення відносної вологості повітря у виробничому приміщенні ускладнює терморегуляцію, зменшення тепловиділення організмом. Фізіологічно оптимальною є відносна вологість в межах 40..60%.

Сукупність описаних показників стану навколишнього середовища називають мікрокліматом. Для приміщень з ЕОМ встановлені норми мікроклімату приведені у таблиці 6.2.

Таблиця 6.2 – Норми мікроклімату для приміщень з ЕОМ

Пора року	Категорія робіт	Температура повітря, С, не більше	Відносна вологість повітря, %	Швидкість руху повітря, м/с
Холодна	Легка – 1а	22-24	40-60	0,1
	Легка – 1б	21-23	40-60	0,1
Тепла	Легка – 1а	23-25	40-60	0,1
	Легка – 1б	22-24	40-60	0,2

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

Небезпека електричного струму на відміну від інших небезпек збільшується тим, що людина не в змозі без спеціальних приладів виявити напругу дистанційно, а також швидкоплинністю поразки – небезпека виявляється, коли людина вже уражена. Ураження струмом приводить до різних порушень в організмі, викликаючи як місцеву поразку тканин і органів, так і загальне ураження організму.

Людина відчуває дратівну дію змінного струму промислової частоти силою 0,6 -1,5 мА і постійного струму 5-7 мА. Ці струми не представляють серйозної небезпеки для організму людини, тому що при такій силі струму можливе самостійне звільнення людини від контакту зі струмоведучими частинами. Для перемінного струму промислової частоти сила невідпускаючого струму знаходиться в межах 6-20 мА. Постійний струм не викликає невідпускаючого ефекту, але приводить до сильних болючих відчуттів, сила такого струму 15-80 мА і більше [26].

6.2 Розрахунок системи пожежогасіння приміщення відділу забезпечення якості офісу

Група приміщення: 1

Тип зрошувача: вода

Розміри приміщення: довжина a = 7 м, ширина b = 6 м

1. Знайдемо групу приміщення (табл. 6.3) згідно з пожежним навантаженням, які забезпечуються автоматичними установками пожежогасіння (ДВН В.2.5-13-98).

За таблицею 6.3 приміщення офісу ІТ-компанії належить до 1 групи.

2. Параметри для розрахунку спринклерної установки залежно від групи приміщення (1) є наступними:

- Інтенсивність зрошування водою L = 0,08 л/(см^2)
- Площа, яка захищається одним зрошувачем $S_{зр}$ = 12 м^2
- Тривалість роботи установки водного пожежогасіння T = 30 хвилин
- Відстань між зрошувачами D = 4 м

3. Знаходимо площу приміщення:

$$S_{прим} = a * b = 7 * 6 = 42 \text{ м}^2$$

4. Знаходимо необхідну кількість зрошувачів:

$$N = S_{прим}/S_{зр} = 42 / 12 = 3,5$$

5. Розміщуємо зрошувачі на плані приміщення.

6. Знаходимо необхідну інтенсивність води в трубопроводі:

$$L_{тр} = L * S_{прим} = 0,08 * 42 = 3,36 \text{ л/с}$$

7. Знаходимо інтенсивність води крізь один спринклер:

$$L_{др} = L_{тр} / N = 3,36 / 3,5 = 0,97 \text{ л/с}$$

8. Знаходимо необхідний об’єм води:

$$Q_н = L_{тр} * T = 3,36 * 1800 \text{ с} = 6048 \text{ л}$$

Таблиця 6.3 – Групи приміщень

Група	Приміщення	Пожежне навантаження
1	Приміщення книгосховищ, ЕОМ, магазинів, будівель управлінь, готелів, лікарень	до 200 Мдж/м^2
2	Приміщення з використанням рідин, які легко спалахують (ЛСР) і горять (ГР); деревообробні, швейні та інші приміщення; підприємства з обслуговування автомобілів	200...2000 Мдж/м^2
3	Приміщення гумотехнічного виробництва	200...2000 Мдж/м^2

4	Приміщення для виробництва, обробки, переробки різних матеріалів з використанням ЛСР і ГР	>2000 Мдж/м^2
5	Склади негорючих матеріалів в упаковці, що горить	>2000 Мдж/м^2
6	Склади твердих матеріалів, що горять	>2000 Мдж/м^2
7	Склади лаків, фарб, ЛСР, ГР, пластмас та ін.	>2000 Мдж/м^2

6.3 Розробка заходів щодо зменшення впливу шкідливих та небезпечних факторів

Сьогодні фахівці в області ергономіки вже зрозуміли, що не можна знайти ідеальне положення, у якому можна перебувати і працювати протягом усього робочого дня. Для більшості людей комфортабельним робочим місцем повинне бути таке, котре можна пристосувати не менш чим для двох позицій, при цьому положення крісла, монітора повинні щораз відповідати виконуваній роботі і звичкам. Багато хто вважають, що для роботи на комп'ютері більше підходять вертикальне і злегка похиле положення. Можливо, щоб крісло було злегка нахилене вперед.

Схеми розміщення робочих місць із персональним комп'ютером повинні ураховувати відстані між робочими столами з відеомоніторами (в напрямку тилу поверхні одного відеомонітора й екрана іншого відеомонітора), які повинні бути не менш 2 м, а відстань між бічними поверхнями відеомоніторів – не менш 1,2 м.

Робочий стіл повинен мати простір для ніг висотою не менш 600 мм, шириною не менш 500 мм, глибиною на рівні колін не менш 450 мм і на рівні витягнутих ніг не менш 650 мм.

Дисплей. Положення тіла звичайно відповідає напрямку погляду; дисплеї, розташовані занадто низько чи під неправильним кутом, є основними причинами сутулості. Відстань від дисплея до очей може варіюватися в залежності від характеру виконуваної роботи – 40-70 см. При роботі з текстом відстань від екрана до очей повинна лише небагато перевищувати відстань між книгою й очима і складати 40-45 см. Необхідно давати відпочинок очам і час від часу їх

просто закривати. Не можна допускати, щоб очі увесь час були сфокусовані на одній відстані, працюючи з текстом, рекомендується установлювати великий шрифт. Дуже важливо, щоб екран монітора не мерехтів. Частота регенерації зображення повинна бути не менш 72 Гц, тому що при більш низькій частоті помітне мерехтіння, хоча люди з особливо чуттєвим зором можуть помітити його і при більш високій частоті. Їм доведеться підібрати графічну плату і монітор з частотою регенерації 85 Гц і вище.

Необхідно уникати того, щоб монітор був звернений убік вікна, оскільки інтенсивна освітленість області зору може затопити потоками світла очі і розмити зображення на сітківці. Якщо приходиться сидіти поруч з вікном, то треба розташуватися під прямим кутом до нього, причому екран дисплея повинний бути перпендикулярний шибці – цим виключаються відблиски на екрані.

Для зниження впливу низькочастотного електромагнітного випромінювання монітора на ЭЛТ рекомендується вибирати монітор, що задовольняє стандарту MPR II, чи TCO 92-95. У противному випадку необхідно установити на екран захисний фільтр, найбільш сучасні моделі, затримують до 99 % електромагнітного випромінювання.

Форма спинки крісла повинна повторювати форму спини працюючого. Крісло необхідно установити на такій висоті, щоб не почувався тиск на куприк (крісло розташоване занадто низько) на стегна (крісло розташоване занадто високо). Фахівці з ергономіки, вважали, що кут між стегнами і хребтом повинний складати 90°, однак недавно проведені дослідження показали, що більшість людей переважно сидять трохи відкинувшись.

Клавіатуру необхідно установити так, щоб не треба було до неї тягтися і нахилятися вперед. При зміні положення тіла (наприклад, з вертикального на похиле) обов'язково треба перемінити положення клавіатури і дисплея. Може виявитися корисною регульована підставка клавіатури, завдяки якій можна без усякої напруги працювати з маніпулятором "миша". Можна поставити клавіатуру і на коліна. Руки при роботі повинні бути прямими в зап'ястях і зігнуті в ліктях

приблизно під прямим кутом. Пальці також повинні бути злегка зігнуті. Удари по клавішах не повинні бути занадто сильними.

Зручна висота столу особливо важлива в тому випадку, коли на ньому розташовується клавіатура. Якщо в клавіатури немає підставки, а висоту столу не можна змінити (і він занадто високий), то треба вище підняти сидіння крісла, а під ноги підставити лавочку: чи що-небудь інше. Якщо стіл занадто низький, необхідно підкласти що-небудь під його ніжки.

Щогодини необхідно робити перерву в роботі. У цей час рекомендується робити вправи для зап'ясть. Нижче описаний можливий комплекс вправ.

- Покласти руку на край столу долонею вниз. Взявшись, за пальці, іншою рукою відвести кисть назад і утримувати протягом 5 секунд. Повторити для іншої руки.

- Злегка упертися рукою в стіл і на 5 секунд напружити пальці в зап'ястя. Повторити іншою рукою.

- Сильно зжати пальці, а потім розпрямити їх.

Виконання приведених рекомендацій дозволяє скоротити вплив шкідливих виробничих факторів на організм людини.

7 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Вступ

Під навколишнім середовищем розуміють цілісну систему взаємопов'язаних природних і антропогенних об'єктів і явищ, під впливом і при безпосередньому використанні яких відбувається праця, побутова діяльність, відпочинок людей. Поняття «навколишнє середовище» включає соціальні, природні і штучно створені фізичні, хімічні та біологічні фактори, тобто все те, що впливає на життя і діяльність людини. Складовою частиною навколишнього середовища є природне середовище. Перед сучасним суспільством стоїть завдання не тільки зберегти природу, а й запобігти негативним наслідкам господарської діяльності людини в майбутньому.

Економічна діяльність у всіх її проявах здійснює забруднення навколишнього середовища. У процесі цієї діяльності забруднюються і стають дефіцитними ресурси повітря, води, територій, що здавалися нескінченними. Нині рівень забруднення досяг загрозливих розмірів, набувши по суті кризового характеру.

Цей процес посилюється розвитком виробничих сил і збільшенням маси речовин, що залучаються в господарський обіг. Через це в навколишнє середовище надходить все більше й більше різноманітних речовин, які йому чужі, а часом токсичні. Значна частина з них не включається в природний кругообіг, накопичується в біосфері і зумовлює небажані екологічні наслідки. Відомо, що екологія – це наука взаємовідносин між живими організмами і сферою їх перебування, тому наслідки промислової і господарської діяльності людства можуть завдати непоправні збитки біосфері і велику шкоду людині.

Забруднюючі речовини, що потрапляють в природне середовище здатні переміщуватись на досить великі відстані, а закономірність цих процесів вивчена ще недостатньо. Ці речовини мігрують у великих кількостях в контурах окремих

складових біосфери. Так в атмосфері вони переносяться повітряними течіями. Ступінь їх розсіювання формується швидкістю і напрямом переміщення повітряних мас і залежить від метеорологічних умов. потрапивши у воду, забруднюючі речовини окиснюються мікроорганізмами або адсорбуються частинками речовин, які є у воді.

Охорона навколишнього середовища являє собою форму відносин між суспільством і природою. Вона здійснюється різними засобами: економічними, правовими, науково-технічними, санітарно-гігієнічними, біологічними та іншими.

В загальному випадку проблема охорони навколишнього середовища зводиться до вирішення двох завдань:

1) організації раціонального природокористування;

2) забезпечення чистоти природних (екологічних) систем.

При здійсненні різних видів економічної діяльності суб'єкти господарювання використовують різноманітні природні ресурси: землю, воду, корисні копалини тощо. Проте ресурси ці обмежені. Обмеженість природних ресурсів була і залишається головною і дуже жорсткою умовою, що накладається на розвиток економіки і відповідно зростання суспільного добробуту.

Наслідком обмеженості природних ресурсів є конкуренція за їх застосування, тобто суперництво між альтернативними цілями використання ресурсів. Адже майже всі ресурси можуть використовуватися для задоволення найрізноманітніших потреб.

Раціональне природокористування означає розробку та здійснення концепції і конкретних заходів щодо раціонального використання і відтворення природних ресурсів, гармонічну взаємодію суспільства і природи, людини і навколишнього природного середовища.

Завдання організації раціонального природокористування вирішується шляхом:

1) оптимального розподілу ресурсів між різними господарськими цілями;

2) використання технологій, що зберігають ресурси;

3) проведення заходів щодо поповнення природних ресурсів.

Іншим, не менш важливим, завданням охорони навколишнього середовища є забезпечення чистоти природних екологічних систем, тобто водного середовища, повітряного басейну, ґрунтових покровів тощо, з тим, щоб забезпечити населення екологічно чистими продуктами харчування, водою, повітрям і, в остаточному підсумку, зберегти високий рівень здоров'я населення та його активного довголіття.

7.1 Забруднення навколишнього середовища

До джерел забруднення довкілля офісу насамперед, належать:

- дизельна електростанція;
- твердопаливні котли;
- побутове сміття;
- електромагнітне випромінювання.

Одночасно в офісі компанії можуть працювати до 300 комп’ютерів, стаціонарних телефонів та велика кількість серверної техніки, яка обслуговує роботу цього обладнання, аварійне чи планове відключення електроенергії призводить до значних перешкод у роботі персоналу. Для того, щоб уникнути таких ситуацій, на території компанії знаходиться дизельна електростанціє. Вона працює як в аварійному, так і в резервному режимі.

Результатом роботи дизельної електростанції є продукти згоряння дизельного палива та шумове забруднення.

Твердопаливні коти використовуються в холодну пору року в системі опалення офісних приміщень. Для опалення використовуються деревні пелети. Хоча таке джерело тепла має менший відсоток шкідливих викидів в атмосферу, ніж, наприклад, вугілля чи мазут, цей вид опалення не можна вважати екологічно чистим. Порівняльні характеристики продуктів згоряння палива наведені в табл. 7.1.

Таблиця 7.1 – Рівні викидів забруднюючих речовин в атмосферу при спалюванні різних видів палива

Вид палива	Викиди забруднюючих речовин в атмосферне повітря без систем очищення, тонн на 1 тис. тонн нат. палива				
	CO2	NO_2	SO_2	Тверді частинки (пил неорг.)	РАЗОМ
Природний газ	1,18	3,52	0,00	0,00	4,70
Древні брикети, пелети	**4,68**	**9,31**	**0,28**	**4,11**	**17,69**
Деревина дров'яна	4,9	9,4	0,3	4,3	18,9
Тирса деревна	5,0	9,6	0,5	5,0	20,0
Деревні відходи, обрізки	5,2	9,9	0,4	5,2	20,7
Швидкозростаюча деревина	4,8	9,5	0,0	8,4	22,7
Тріска, сучки, кора	5,6	11,4	0,8	13,4	31,3
Мазут	5,20	5,20	35,30	0,30	45,90
Брикет торф'яний	8,04	26,81	3,00	13,02	50,87
Кам'яне вугілля	9,58	63,56	9,20	65,32	147,66

Побутові відходи – це залишки речовин і предметів, які утворюються в результаті побутової та господарської діяльності людини і які не можуть бути використані на місці утворення, а їх накопичення і зберігання порушують санітарний стан навколишнього середовища.

Всі побутові відходи, які утворюються в процесі функціонування компанії, можна розділити на рідкі та тверді. До рідких побутових відходів належать нечистоти з вигребів туалетів, помиї (від миття посуду, підлоги, тощо) і стічні води (побутові, зливові). До твердих побутових відходів можна віднести сміття (побутові відходи) та кухонні відходи. Зокрема, твердими відходами є папір, картон, скло, пластмаса, продукти харчування.

Генератором електромагнітного забруднення є, в першу чергу, велика кількість комп’ютерної техніки на території офісу. Більшість із них працює 8 годин на добу, а деякі не вимикаються цілодобово. Випромінювачами в даному випадку є і процесор, і монітор. Випромінювання останнього значно вищі, особливо його бічні і задні стінки, адже вони не мають спеціального захисного покриття, як у лицьовій частині монітора. Крім того, в офісі є дві кухні, кожна з яких обладнана 5 мікрохвильовими печами.

Електромагнітне випромінювання має несприятливий вплив на організм людини. Його наслідками можуть бути головний біль, порушення сну, перевтома і, навіть, у деяких випадках стенокардія.

7.2 Розробка заходів щодо зменшення забруднення

Для того, щоб зменшити рівень забруднення навколишнього середовища, варто вжити певних заходів, які мінімізують цей вплив, якщо його неможливо усунути цілком. Серед таких заходів:

1) Очищення димових газів у мокрих золоуловлювачах.

Електрофільтри на електростанціях застосовуються для досягнення найбільш глибокого очищення димових газів в основному на великих енергоблоках потужністю 300 МВт та більше. Мокрі золоуловлювачі, які працюють при маленьких питомих витратах води та невеликих перепадах тиску, встановлюються за котлоагрегатами середньої паропродуктивності. У мокрих золоуловлювачах уловлювання часток золи на плівці води, яка стікає по його стінках, здійснюється за рахунок відцентрової сили, яка діє на частки. Ефективність апарата не перевищує 90%.

2) Використання природного газу для опалення офісних приміщень замість деревних пелетів.

Згідно з показниками, наведеними в табл. 7.1, рівень шкідливих викидів від згоряння природного газу є найнижчим серед перерахованих видів палива. Проте зміну джерела енергії слід проводити, враховуючи економічну ефективність, оскільки вартість цих видів палива не є однаковою.

3) Раціональне позбавлення від побутових відходів.

Деякі побутові відходи можна безпечно спалювати для отримання енергії. Вторинну сировину (макулатуру, скло, пластмаси) можна відправляти на переробку, а не вивозити на сміттєзвалища. Крім того, зменшити кількість

побутових відходів допоможе повторне їх використання (наприклад, скляних чи пластикових пляшок та контейнерів для їжї), скорочення споживання та використання меншої кількості упаковки.

4) Зменшити рівень електромагнітного забруднення та захиститися від його надмірного впливу.

Зрозуміло, що неможливо повністю обійтися без електроприладів та комп’ютерів, проте можна дещо зменшити їх негативний вплив. Для цього слід вимикати з електромережі комп’ютери, телефони та обладнання, з якими ніхто не працює. Також варто час від часу виходити на прогулянки і робити перерви в роботі з комп'ютером.

Що стосується мікрохвильових печей, то їх потужність може змінюватись, тому час від часу треба звертатися до майстра, щоб контролювати рівень випромінювання.

ВИСНОВКИ

В кваліфікаційній роботі було досягнуто поставленої мети: підвищення достовірності оцінювання тривалості програмних проектів для mainframe.

В процесі проведення досліджень одержані наступні результати:

1. Проаналізовано особливості, переваги та недоліки існуючих моделей та методів оцінювання тривалості програмних проектів; визначено задачі, які необхідно вирішити для досягнення мети дослідження – підвищення достовірності оцінювання тривалості програмних проектів для mainframe. Аналіз показав, що для отримання достовірних результатів необхідно використовувати нелінійні регресійні моделі та нормалізуюче перетворення для емпіричних даних тривалості програмних проектів.
2. *Удосконалено* нелінійну регресійну модель ISBSG для оцінювання тривалості програмних проектів для mainframe в залежності від трудомісткості розробки цих програмних проектів *за рахунок* побудови рівнянь границь довірчого інтервалу та рівнянь границь інтервалу прогнозування нелінійної регресії, *що дозволяє* підвищити достовірність оцінювання тривалості програмних проектів для mainframe. Удосконалена модель краща ніж ISBSG на 6,38% за SSE, на 7.67% за R^2 та на 2.83% за δ.
3. Розроблене програмне забезпечення для оцінювання тривалості програмних проектів для mainframe дозволить скоротити час відповідного оцінювання, забезпечить зберігання результатів оцінювання, а також надасть швидкий доступ до результатів попередніх оцінювань.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Standish Group 2015 Chaos Report - Q&A with Jennifer Lynch // S. Hastie, S. Wojewoda. – InfoQ, 2015. – Режим доступу : https://www.infoq.com/articles/standish-chaos-2015.
2. Chow, A. W. Engaging a corporate community to manage technology and embrace innovation / A. W. Chow, B. D. Goodman, J. W. Rooney, C. D. Wyble. // IBM Systems Journal. – Vol. 46, No. 4, 2007. – P. 639 - 650.
3. Guide to the Software Engineering Body of Knowledge – Los Alamitos, CA : IEEE Computer Society, 2004. – 204 p.
4. McConnell, S. Rapid Development: Taming Wild Software Schedules / S. McConnell. – Redmond, WA : Microsoft Press, 1996. – 660 p.
5. A guide to the project management body of knowledge. Fifth edition – Project Management Institute, 2013. – 589 p.
6. Boehm, B. W. Software engineering economics / B. W. Boehm. – Englewood Cliffs, NJ : Prentice Hall, 1981. – 768 p.
7. Пухалевич, А. В. Моделі та інформаційна технологія переробки інформації для оцінювання тривалості проектів з розробки програмного забезпечення : Дис. канд. техн. наук: 05.13.06 / Пухалевич Андрій Володимирович, НУК ім. адм. Макарова – Миколаїв, 2017. – 179 с.
8. Oligny, S. An empirical assessment of project duration models in software engineering / S. Oligny, P. Bourque, A. Abran. // In Proc. 8th European Software Control and Metrics Conference ESCOM. – Berlin, 1997.
9. Oligny, S. Exploring the relation between effort and duration in software engineering projects / S. Oligny, P. Bourque, A. Abran, B. Fournier // In proc. of the World Computer Congress, Aug. 2000. – Pp. 175-178.

10. Vetrici, M. Software Project Duration Estimation Using Metrix Model / M. Vetrici. – Informatica Economica Journal. – Vol. 3, No. 47, 2008. – Pp. 87-91.

11. Boehm, B. W. Software Cost Estimation with COCOMO II. / B. W. Boehm. – Upper Saddle River, NJ : Prentice Hall PTR, 2000. – 586 p.

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

13. Bittner, K. Managing Iterative Software Development Projects / K. Bittner, I. Spence. – Addison-Wesley Professional, 2006. – 640 p.

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

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

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

17. Barry, E.J. Software Project Duration and Effort: An Empirical Study / E. J. Barry, T. Mukhopadhyay, S. A. Slaughter // Information Technology and Management. – Vol. 3, Issue 1, January 2002. – Pp. 113-136.

18. McConnell, S. Software Estimation: Demystifying the Black Art / S. McConnell. – Microsoft Press, 2006. – 308 p.

19. Galorath, D. D. Software sizing, estimation, and risk management: when performance is measured performance improves / D. D. Galorath, M. W. Evans. – CRC Press, 2006. – 576 p.

20. Ambler, S., Software development project success survey 2008 / S. Ambler. – Dr. Dobbs Journal, 2009.

21. Buglione, L. Estimation tools and techniques / L. Buglione, C. Ebert. IEEE Software, Vol. 23, No. 3, 2011. – Pp. 15-18.

22. The IFPUG Guide to IT and software measurement / IFPUG. – CRC Press, 2012. – 848 p.

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

24. Димо, О. Б. Підвищення ефективності управління проектами розробки програмного забезпечення з відкритим вихідним кодом : Дис. канд. техн. наук: 05.13.22 / Олександр Борисович Димо, НУК ім. адм. Макарова – Миколаїв, 2007. – 226 с.

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

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

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

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

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

30. Приходько, С. Б. Доверительный интервал нелинейной регрессии времени восстановления работоспособности устройств терминальной сети / С. Б. Приходько, Л. Н. Макарова // Восточно-Европейский журнал передовых технологий. – Харьков : Технологический центр, 2014. – № 3/4 (69). – С. 26-31. – ISSN 1729-3774.

31 Мишенін, О. І. Удосконалення моделей ISBSG для оцінювання тривалості програмних проектів / О. І. Мишенін, Є. В. Трофімчук, А. В. Пухалевич // Матеріали III Всеукраїнської науково-практичної інтернет-конференції студентів, аспірантів та молодих вчених за тематикою «Сучасні комп'ютерні системи та мережі в управлінні»: збірка наукових праць / Під редакцією Г.О. Райко. – Херсон: Видавництво ФОП Вишемирський В. С., 2020. – С. 259-260. ISBN 978-617-7783-98-4.

32. Орлов, А. И. Прикладная статистика / А. И. Орлов. – Москва : "Экзамен", 2004. – 656 с.

33. Хан, Г. Статистические модели в инженерных задачах. Пер. с англ. / Г. Хан, С. Шапиро. – М. : Мир, 1969. – 396 с.

34. Yan, X. Linear regression analysis : theory and computing / X. Yan, X. G. Su. – Singapore : World Scientific Publishing Co. Pte. Ltd., 2009. – 349 p.

ДОДАТОК А
ТЕХНІЧНЕ ЗАВДАННЯ

Вступ

Назва програми: «Оцінювання тривалості програмних проектів».

Позначення програми: «DurationEstimate»

Область застосування:

Область застосування програми обмежена сферою розробки програмного забезпечення.

1 Підстави для розробки

Підставою для розробки програмного продукту є завдання на магістерську роботу «Математичні моделі для оцінювання тривалості програмних проектів та розробка програмного забезпечення для їх реалізації».

2 Призначення програми:

2.1 Функціональне призначення

Даний програмний продукт призначений для оцінювання тривалості програмних проектів.

2.2 Експлуатаційне призначення

Програма може використовуватися для оцінювання тривалості програмних проектів.

3 Вимоги до програми

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу функцій, що виконуються

Програма повинна забезпечувати можливість виконування наведених нижче функцій:

- Оцінювання середнього значення тривалості проекту з розробки програмного забезпечення з використанням нелінійної регресійної моделі в залежності від трудомісткості;
- Оцінювання 95% довірчого інтервалу середнього значення тривалості проекту з розробки програмного забезпечення;
- Оцінювання 95% інтервалу прогнозування тривалості проекту з розробки програмного забезпечення;
- Збереження результатів оцінювання в базу даних;
- Перегляд результатів раніше проведених оцінювань тривалості програмних проектів.

3.1.2 Вимоги до організації вхідних та вихідних даних

Внесення вхідних даних виконується шляхом їх введення у відповідні поля форми програми.

Вихідні дані (результати оцінювання тривалості, довірчий інтервалу та інтервал передбачення) повинні виводитися в таблиці на формі програми.

3.2 Вимоги до надійності

3.2.1 Вимоги до забезпечення надійності функціонування програми

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

3.2.2 Відмова виконання програмного продукту через некоректні дії користувача.

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

3.3 Умови експлуатації

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

3.5 Вимоги до інформаційної та програмної сумісності

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

3.6 Вимоги до захисту інформації та програм

Вимоги до захисту інформації та програм відсутні.

3.7 Вимоги до маркування та упаковки

3.7.1 Вимоги до маркування:

Вимоги до маркування відсутні.

3.7.2 Вимоги до упаковки:

Вимоги до упаковки відсутні.

3.8 Вимоги до транспортування та зберігання

Вимоги до транспортування програми відсутні.

4 Вимоги до програмної документації

Склад програмної документації:

Технічне завдання;

Текст програми;

Опис програми;

Програма і методика випробувань;

Результати проходження тестів;

Інструкція користувача.

5 Техніко-економічні показники

Економічна ефективність впровадження програмного забезпечення розрахована у розділі 5. Згідно з отриманими результатами впровадження даного ПЗ є доцільним, а термін його окупності становить приблизно 6 місяців.

6 Порядок контролю та приймання

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

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

ДОДАТОК Б
ОПИС ПРОГРАМИ «ОЦІНЮВАННЯ ТРИВАЛОСТІ ПРОГРАМНИХ ПРОЕКТІВ»

Програма для точкового та інтервального оцінювання тривалості програмних проектів для платформи main frame в залежності від трудомісткості представляє собою застосунок на мові програмування Java, який дозволяє виконувати наступні дії:

- Оцінювання середнього значення тривалості проекту з розробки програмного забезпечення з використанням нелінійної регресійної моделі в залежності від трудомісткості;
- Оцінювання 95% довірчого інтервалу середнього значення тривалості проекту з розробки програмного забезпечення;
- Оцінювання 95% інтервалу прогнозування тривалості проекту з розробки програмного забезпечення;
- Збереження результатів оцінювання в базу даних;
- Перегляд результатів раніше проведених оцінювань тривалості програмних проектів.

ДОДАТОК В
ІНСТРУКЦІЯ КОМП'ЮТЕРНОЇ ПРОГРАМИ «ОЦІНЮВАННЯ ТРИВАЛОСТІ ПРОГРАМНИХ ПРОЕКТІВ»

1 ВСТУП

Ця настанова призначена для ознайомлення користувача з технічними характеристиками і функціональними можливостями комп'ютерної програми «Оцінювання тривалості програмних проектів».

Дана програма призначена для точкового та інтервального оцінювання тривалості програмних проектів для платформ PC (personal computers), MR (mid-range), MF (main frame) в залежності від трудомісткості. Програма є незалежним програмним продуктом, що має графічний інтерфейс, який дозволяє введення даних, перегляд результатів оцінювання тривалості програмних проектів та збереження отриманих результатів в базу даних.

2 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ПРОГРАМУ

2.1 Позначення і найменування програми

Найменування програми - «Оцінювання тривалості програмних проектів». Позначення програми - "DurationEstimate".

2.2 Мови програмування, на яких написана програма

Програма написана на мові програмування Java.

2.3 Призначення програми

Оцінювання тривалості програмних проектів для платформ PC (personal computers), MR (mid-range), MF (main frame) в залежності від трудомісткості.

2.4 Можливості програми

Програма виконує наступні функції:

- Оцінювання середнього значення тривалості проекту з розробки програмного забезпечення з використанням нелінійної регресійної моделі в залежності від трудомісткості;
- Оцінювання 95% довірчого інтервалу середнього значення тривалості проекту з розробки програмного забезпечення;
- Оцінювання 95% інтервалу прогнозування тривалості проекту з розробки програмного забезпечення;
- Збереження результатів оцінювання в базу даних;
- Перегляд результатів раніше проведених оцінювань тривалості програмних проектів.

2.5 Обмеження області застосування програми

- Область застосування програми обмежена сферою розробки програмного забезпечення.

3 УМОВИ ЗАСТОСУВАННЯ ПРОГРАМИ

3.1 Умови, необхідні для виконання програми

Спеціальні умови для виконання програми відсутні.

3.2 Вимоги до технічних засобів, необхідних для функціонування програми

Рекомендовані вимоги:

- ПК з процесором не нижче Core II Duo;
- Обсяг оперативної пам'яті - не менше 1 Гб;
- Необхідний обсяг на жорсткому диску - не менше 50 Мб.

3.3 Програмне забезпечення, необхідне для функціонування програми

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

4 ОПИС ПРОГРАМИ

4.1 Загальний вигляд стартового вікна програми

Загальний вигляд стартового вікна комп'ютерної програми представлений на рисунку Б.1.

Стартове вікно містить 3 поля для ведення вхідних даних:

- Текстове поле «Назва проекту» – дозволяє вказати назву проекту при необхідності подальшого збереження результатів оцінювання в базу даних;
- Текстове поле «Трудомісткість» – дозволяє вибрати значення трудомісткості, яке буде основою для оцінювання тривалості програмних проектів;
- Випадаючий список «Платформа» – дозволяє вибрати платформу, для якої розроблюється програмне забезпечення: "Персональні комп’ютери (PC)", "Платформи середнього трудомісткості (MR)", "Мейнфрейми (MF)".

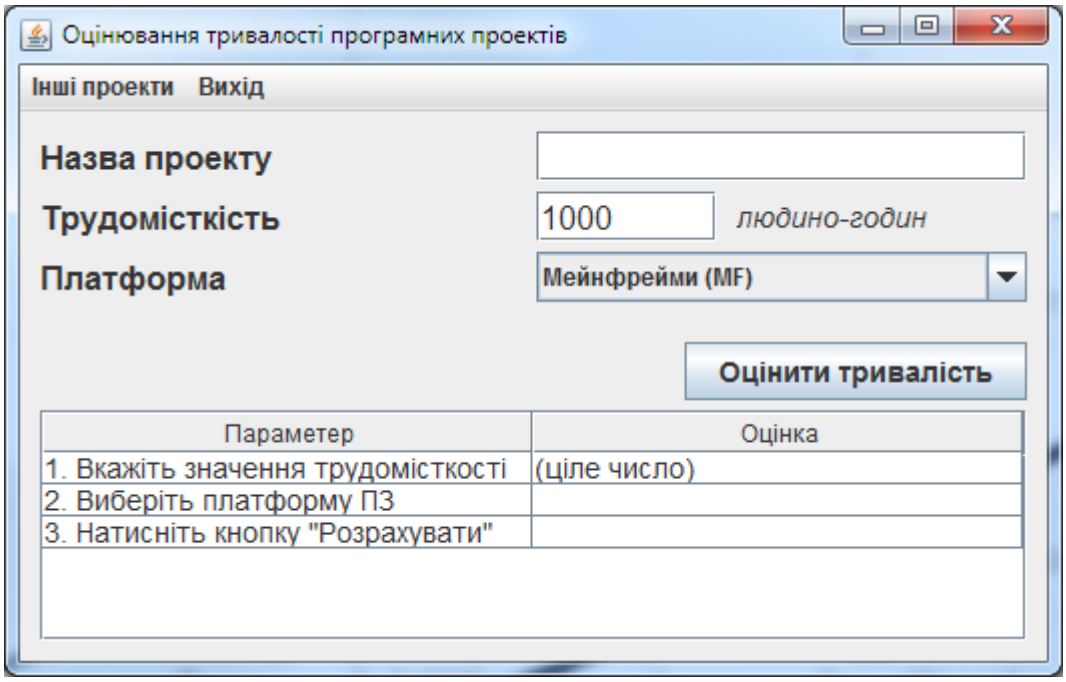

Рисунок Б.1 – Загальний вид стартового вікна комп'ютерної програми

4.2 Результати оцінювання тривалості програмних проектів

Після введення вхідних даних у вказані поля стартового вікна користувачу потрібно натиснути кнопку «Оцінити тривалість». Результати оцінювання тривалості програмних проектів (Оцінка тривалості, Довірчий інтервал нелінійної регресії, Інтервал прогнозування) будуть показані в таблиці в нижній частині вікна програми (рисунку Б.2).

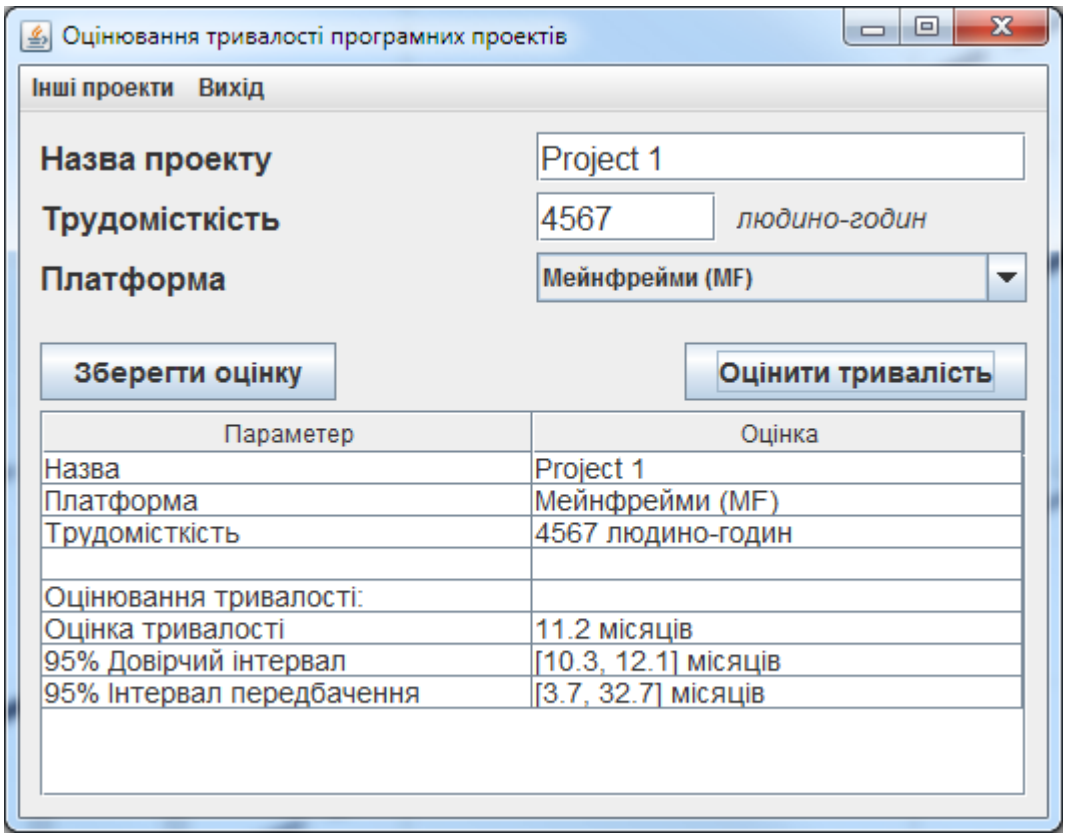

Рисунок Б.2 – Результати оцінювання тривалості програмних проектів

При необхідності результати оцінювання можна зберегти в базу даних, натиснувши на кнопку «Зберегти оцінку». Для перегляду раніше збережених результатів оцінювання (рисунку Б.3) потрібно вибрати пункт меню «Інші проекти» вверху програми.

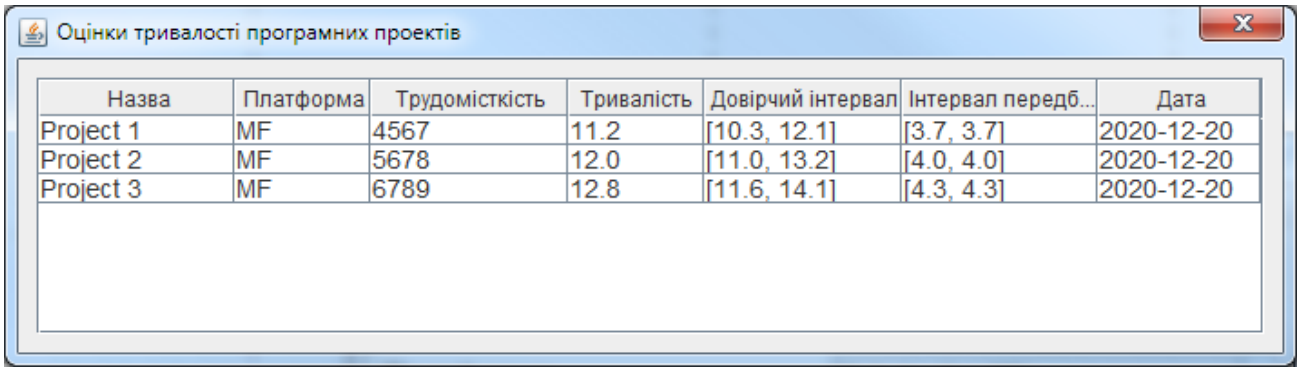

Рисунок Б.3 – Перегляд збережених результатів оцінювання тривалості програмних проектів

ДОДАТОК Г
ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ

Об'єкт випробувань

Повне найменування програми: Оцінювання тривалості програмних проектів.

Коротка характеристика галузі застосування: сфера розробки програмного забезпечення.

Мета випробувань

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

Вимоги до програми

Програма повинна забезпечувати можливість виконання нижче перерахованих функцій:

- Оцінювання середнього значення тривалості проекту з розробки програмного забезпечення з використанням нелінійної регресійної моделі в залежності від трудомісткості;
- Оцінювання 95% довірчого інтервалу середнього значення тривалості проекту з розробки програмного забезпечення;
- Оцінювання 95% інтервалу прогнозування тривалості проекту з розробки програмного забезпечення;
- Збереження результатів оцінювання в базу даних;
- Перегляд результатів раніше проведених оцінювань тривалості програмних проектів.

Вимоги до програмної документації

Програмна документація містить:

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

Засоби і порядок випробувань

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

Середовище виконання:

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

Порядок випробувань:

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

Методи випробувань

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

Результати випробувань

Підтверджені можливості та функції програмного забезпечення «Оцінювання тривалості програмних проектів».

Відомості про відмови, збої і аварійні ситуації

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

Відомості про коригування параметрів об’єкта випробувань та технічної документації

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

ДОДАТОК Д
ТЕКСТ ПРОГРАМИ

Клас EstimationApp

```
package EstimationApp;

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

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

public class EstimationApp {

     static JFrame estimationFrame;
     static EstimationForm estimationForm;

     static JDialog projectsFrame;
     static ProjectsForm projectsForm;

     static public EntityManager em;

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

     estimationFrame.setContentPane(estimationForm.$$$getRootCompone
nt$$$());

     estimationFrame.setDefaultCloseOperation(WindowConstants.EXIT_O
N_CLOSE);
          estimationFrame.pack();
          estimationFrame.setVisible(true);
          estimationFrame.setLocationRelativeTo(null);
     }
```

```
    private static void setupEntityManager() {
        Properties database_properties = new Properties();
        database_properties.put(TopLinkProperties.JDBC_URL,
EstimationApp.getJDBCUrl());
        em = Persistence.createEntityManagerFactory("dur_est.db",
database_properties).createEntityManager();
    }

    private static void setupMenu() {
        JMenuBar menuBar = new JMenuBar();
        JMenu menuItem;

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

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

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

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

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

    public static void showProjectsFrame() {
```

```
          projectsFrame = new JDialog(estimationFrame, "Оцінки
тривалості програмних проектів", true);
          projectsFrame.setResizable(true);
          projectsForm = new ProjectsForm();
          projectsForm.setupUIComponents();

     projectsFrame.setContentPane(projectsForm.$$$getRootComponent$$
$());

     projectsFrame.setDefaultCloseOperation(WindowConstants.DISPOSE_
ON_CLOSE);
          projectsFrame.pack();
          projectsFrame.setVisible(true);
     }

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

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

Клас ProjectsForm

```
package EstimationApp.forms;

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

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

public class ProjectsForm {

     private JPanel View;
     private JTable estimationsListTable;

     public ObservableList<Estimation> estimationList;

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

```
    }

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

    estimationsListTable.getColumnModel().getColumn(1).setMaxWidth(
80);

    estimationsListTable.getColumnModel().getColumn(3).setMaxWidth(
80);

    estimationsListTable.getColumnModel().getColumn(6).setMaxWidth(
90);

    estimationsListTable.getColumnModel().getColumn(6).setPreferred
Width(90);
    }

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

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

    /**
     * Method generated by IntelliJ IDEA GUI Designer
```

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

    estimationsListTable.setFont(estimationsListTableFont);
        }
        estimationsListTable.setShowHorizontalLines(true);
        scrollPane1.setViewportView(estimationsListTable);
    }

    /**
     * @noinspection ALL
     */
    public JComponent $$$getRootComponent$$$() { return View; }
}
```

Клас EstimationResults

```
package EstimationApp.forms;

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

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

public class EstimationResults extends DefaultTableModel {

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

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

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

     public void fillResults() {
          setRowCount(0);

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

          double e_z = Math.log10(size.doubleValue());
          double t_z_estimate = LinearRegression.estimateY(e_z,
rp.b0, rp.b1);
          double t_z_estimate_l1 =
LinearRegression.estimateYConfidence(e_z, rp.b0, rp.b1, false,
false, rp.sse, rp.x_avg, rp.Sxx, rp.n, tStudent);
          double t_z_estimate_r1 =
LinearRegression.estimateYConfidence(e_z, rp.b0, rp.b1, true, false,
rp.sse, rp.x_avg, rp.Sxx, rp.n, tStudent);
          double t_z_estimate_l2 =
LinearRegression.estimateYConfidence(e_z, rp.b0, rp.b1, false, true,
rp.sse, rp.x_avg, rp.Sxx, rp.n, tStudent);
          double t_z_estimate_r2 =
LinearRegression.estimateYConfidence(e_z, rp.b0, rp.b1, true, true,
rp.sse, rp.x_avg, rp.Sxx, rp.n, tStudent);

          effort = MathUtil.round(Math.pow(10, t_z_estimate), 1);
          effort_confidence_interval_bottom =
MathUtil.round(Math.pow(10, t_z_estimate_l1), 1);
          effort_confidence_interval_top =
MathUtil.round(Math.pow(10, t_z_estimate_r1), 1);
```

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


          addRow(new String[]{"Назва", this.title});
          addRow(new String[]{"Платформа",
this.platform.toString()});
          addRow(new String[]{"Трудомісткість", size.toString() + "
функціональних точок"});

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

Клас LinearRegression

```
package DataProcess;

import Util.MathUtil;

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

public class LinearRegression {

     List<Double> x;
     public DataAnalyzeGauss ds_x;

     List<Double> y;
     public DataAnalyzeGauss ds_y;

     List<Double> predicted_y;

     List<Double> residuals;
     DataStat ds_residuals;

     public Double b0;
     public Double b1;

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

```
    public double tStudent;

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

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

    protected void setVariables(List<Double> x, List<Double> y) {
        this.x = x;
        ds_x = new DataAnalyzeGauss(x);

        this.y = y;
        ds_y = new DataAnalyzeGauss(y);

        n = x.size();

        tStudent = DataAnalyze.getTStudent(n - k, 0.025); //p =
0.95
    }

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

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


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


        b0 = y_avg - b1 * x_avg;
```

```
    }

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

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

        return S;
    }

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

     public double getCriticalExternallyStudentizedResidual() {
return 2.; }

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

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

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

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

     public void printIsolatedEntries () {
```

```
          double criticalLeverage = getCriticalLeverage();
          System.out.println("Critical Leverage="+ criticalLeverage);
          for (int i = 0; i < n; i++) {
               final double leverage = leverage(i);
               if (leverage < criticalLeverage )
                    continue;

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

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

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

     public List<Integer> getIsolatedIndexes() {
          double criticalLeverage = getCriticalLeverage();
          double criticalExternallyStudentizedResidual =
getCriticalExternallyStudentizedResidual();

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

               isolated.add(i);
               if (true)
                    continue;

               System.out.print(i);
               if (!(leverage < criticalLeverage))
                    System.out.print(" High leverage");
               if (!(stud_residual <
criticalExternallyStudentizedResidual))
                    System.out.print(" High residual");
               System.out.println();
```

```
            }

            return isolated;
      }
}
```

Клас Estimation

```
package EstimationApp.models;

import javax.persistence.*;

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

      public String title;
      public String platform;
      public Integer size;
      public Double effort;
      public Double effort_confidence_interval_bottom;
      public Double effort_confidence_interval_top;
      public Double effort_estimation_interval_bottom;
      public Double effort_estimation_interval_top;
      public String created;


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

Клас AppModel

```
package EstimationApp.models;

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

@MappedSuperclass
public abstract class AppModel implements Serializable {

      @Id
      public Integer id;

      public AppModel() {
      }
```

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

     public Integer getId() {
          return id;
     }

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

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

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

          AppModel other = (AppModel) object;
          return !((this.id == null && other.id != null) || (this.id
!= null && !this.id.equals(other.id)));
     }

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

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

Клас LRParameters

```
    public class LRParameters {
          public double b0;
          public double b1;
          public double sse;
          public double x_avg;
          public double Sxx;
          public int n;
```

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