МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

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

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

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «магістр»

на тему: Нелінійна регресійна модель для оцінювання тривалості розробки Java-застосунків для ПК та розробка програми для її реалізації

Виконав: студент 6151мз групи
А.В. Пухалевич
(підпис, ПІБ)

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

Миколаїв – 2021 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

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

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

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

Студенту Пухалевичу Андрію Володимировичу _____
(Прізвище, ім'я, по батькові)

1. Тема роботи: Нелінійна регресійна модель для оцінювання тривалості розробки Java-застосунків для ПК та розробка програми для її реалізації _____

Керівник роботи Приходько Сергій Борисович _____

Затверджені наказом ректора № 1239уч від «13» _____10_____ 2021 р.

2. Термін подання роботи: 06.12.2021 р. _____

3. Вихідні дані по роботі: _____

_____

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

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

5. Перелік презентаційних матеріалів _____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

6. Консультанти розділів роботи

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

7. Дата видачі завдання 13.10.2021 р.

К А Л Е Н Д А Р Н И Й П Л А Н

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

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

Студент _____ А.В. Пухалевич
(підпис) (ПІБ)

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

РЕФЕРАТ

Пухалевич Андрій Володимирович

«Нелінійна регресійна модель для оцінювання тривалості розробки Java-застосунків для ПК та розробка програми 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, 2021 р.

Обсяг роботи: 104 стор., 6 табл., 11 рис., 29 використаних джерел, 5 додатків

Актуальність теми: Визначається низькою достовірністю оцінювання тривалості розробки Java-застосунків для ПК з використанням існуючих моделей.

Мета та завдання дослідження. Метою роботи є підвищення достовірності оцінювання тривалості розробки Java-застосунків для ПК.

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

Предмет дослідження: нелінійна регресійна модель для оцінювання тривалості розробки Java-застосунків для ПК та програмне забезпечення для її реалізації.

Методи дослідження: методи математичної статистики та теорії ймовірностей, регресійного аналізу, об`єктно-орієнтованого програмування.

Наукова новизна одержаних результатів: *отримала подальший розвиток* модель ISBSG для оцінювання тривалості розробки застосунків для платформи ПК в залежності від трудомісткості розробки *за рахунок* побудови моделі тільки для Java-застосунків та додавання в модель випадкової змінної, *що дозволяє* підвищити достовірність оцінювання тривалості розробки Java-застосунків для платформи ПК.

Практичне значення одержаних результатів. Розроблена програма, що реалізує побудовану нелінійну модель для оцінювання тривалості розробки

Java-застосунків для платформи ПК, може використовуватися в ІТ компаніях для оцінювання тривалості розробки Java-застосунків для платформи ПК.

Апробація результатів роботи. Основні положення й результати кваліфікаційної роботи доповідалися на ІІ Всеукраїнській науково-практичній інтернет-конференції «Інформаційні технології: моделі, алгоритми, системи (ITMAS-2021)» (м. Миколаїв, 26-28 жовтня 2021 року).

Ключові слова: Java-застосунки, оцінювання тривалості, розробка програмних застосунків, нелінійні регресійні моделі, нормалізуючі перетворення.

ABSTRACT

Pukhalevych Andrii Volodymyrovych

«The nonlinear regression model for estimating the development duration of the PC Java applications and developing software for its implementation»

(тема магістерської роботи)

Qualification work for obtaining a master's degree in specialty 121 - "Software Engineering". Admiral Makarov National University of Shipbuilding. Mykolaiv, 2021

Volume: 104 p., 6 tables, 11 figures, 29 references, 5 appendices.

Topic Relevance: It is determined by the low reliability of estimation of the development duration of Java-applications for PCs using existing models.

Research goal and objectives: increasing the reliability of the estimation of the development duration of the PC Java applications.

Object of research: the process of the estimation of the development duration of the PC Java applications.

Subject of research: The nonlinear regression model for estimating the development duration of the PC Java applications.

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

Scientific contribution: the nonlinear regression model for estimating the development duration of the PC Java applications was improved.

Practical value of obtained results. Developed software to for estimating the development duration of the PC Java applications.

Approbation of the thesis results. The 2nd All-Ukrainian scientific-practical Internet conference "Information technologies: models, algorithms, systems (ITMAS-2021)" (Mykolaiv, October 26-28, 2021).

Key words: Java-applications, duration estimation, software applications development, non-linear regression models, normalizing transformations.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API	Application programming interface
COCOMO	Constructive cost model
JRE	Java runtime environment
ISBSG	International software benchmarking standards group
БД	База даних
ОЗП	Оперативний запам’ятовувальний пристрій
ПК	Персональний комп’ютер
ПЗП	Постійний запам'ятовувальний пристрій
СУБД	Система управління базами даних

ВСТУП

Актуальність теми. Оцінювання тривалості розробки програмного забезпечення, в тому числі Java-застосунків для платформи ПК, є важливою задачею програмної інженерії. Це пов'язано з тим, що достовірні оцінки тривалості розробки є одним з факторів, які дозволяють успішно завершувати програмні проєкти [1], а кросплатформна мова програмування Java є однією з найбільш поширених мов для платформи ПК.

Звіти Standish Group та інші дослідження показують, що в більшості випадків проєкти з розробки програмного забезпечення виконуються пізніше запланованих термінів [2-5], тобто існує проблема низької достовірності відповідного оцінювання.

Одним з основних методів оцінювання тривалості розробки програмного забезпечення згідно [6] є параметричне оцінювання. При параметричному оцінюванні застосовуються нелінійні регресійні моделі тривалості розробки програмного забезпечення в залежності від трудомісткості розробки на основі логарифмічного перетворення COCOMO [7], ISBSG [8], та моделі на основі перетворення Джонсона [9-12]. Моделі [7] і [8] були побудовані для платформ mainframe, midrange та PC (персональні комп’ютери – ПК) без використання мови програмування як ознаки кластеризації даних. Ці моделі, в тому числі моделі для платформи ПК, побудовані за застарілими емпіричними даними (реліз ISBSG 1996 року). Крім того, моделі COCOMO і ISBSG не дозволяють проводити оцінювання довірчих інтервалів та інтервалів прогнозування тривалості розробки програмного забезпечення. Вказані фактори можуть призводити до зниження достовірності оцінювання тривалості розробки Java-застосунків для платформи ПК.

Аналіз даних репозитарію ISBSG з релізу 2021 року показав, що поділ моделей тільки по платформі вже є недостатнім, оскільки отримані моделі

мають погані значення PRED та MMRE [13]. Таким чином, побудова нелінійної регресійної моделі для оцінювання тривалості розробки Java-застосунків для платформи ПК є актуальною, а розробка програми для реалізації такої моделі має практичну цінність.

Мета і завдання дослідження.

Мета кваліфікаційної роботи полягає у підвищенні достовірності оцінювання тривалості розробки Java-застосунків для платформи ПК.

Для досягнення мети, поставленої в роботі, необхідно вирішити наступні завдання:

1. Дослідити існуючі моделі для оцінювання тривалості розробки застосунків.
2. Розглянути перетворення, які можна використати, щоб нормалізувати емпіричні дані тривалості та трудомісткості Java-застосунків для платформи ПК при побудові нелінійних регресійних моделей з використанням методу їх побудови на основі нормалізуючих перетворень.
3. Побудувати нелінійну регресійну модель для оцінювання тривалості розробки Java-застосунків для платформи ПК в залежності від трудомісткості розробки (рівняння нелінійної регресії, рівняння границь довірчого інтервалу нелінійної регресії, рівняння границь інтервалу прогнозування нелінійної регресії).
4. Розробити проєкт програми для реалізації нелінійної регресійної моделі для оцінювання тривалості розробки Java-застосунків для платформи ПК.
5. Провести розрахунок економічної ефективності від розробки та впровадження програми, що реалізує побудовану модель для оцінювання тривалості розробки Java-застосунків для платформи ПК;
6. Розглянути питання з охорони праці при використанні програми, що реалізує побудовану модель для оцінювання тривалості розробки Java-застосунків для платформи ПК;
7. Розглянути питання з охорони навколишнього середовища на підприємствах при використанні програми, що реалізує побудовану модель для оцінювання тривалості розробки Java-застосунків для платформи ПК.

Об’єктом дослідження є процес оцінювання тривалості розробки Java-застосунків для платформи ПК.

Предметом дослідження є нелінійна регресійна модель для оцінювання тривалості розробки Java-застосунків для ПК та програмне забезпечення для її реалізації.

Методи дослідження. При проведені дослідження використовувалися методи математичної статистики та теорії ймовірностей, регресійного аналізу, об`єктно-орієнтованого програмування.

Методи математичної статистики та теорії ймовірностей використовувалися для аналізу та обробки емпіричних даних тривалості та трудомісткості розробки Java-застосунків для платформи ПК. Методи регресійного аналізу використовувалися при побудові нелінійної регресійної моделі для оцінювання тривалості розробки Java-застосунків для платформи ПК. Методи об`єктно-орієнтованого програмування використовувалися при розробці програмного забезпечення, що реалізує побудовану модель для оцінювання тривалості розробки Java-застосунків для платформи ПК.

Наукова новизна одержаних результатів. *Отримала подальший розвиток* модель ISBSG для оцінювання тривалості розробки застосунків для платформи ПК в залежності від трудомісткості розробки *за рахунок* побудови моделі тільки для Java-застосунків та додавання в модель випадкової змінної, *що дозволяє* підвищити достовірність оцінювання тривалості розробки Java-застосунків для платформи ПК.

Практичне значення одержаних результатів. Розроблена програма, що реалізує побудовану нелінійну модель для оцінювання тривалості розробки Java-застосунків для платформи ПК, може використовуватися в ІТ компаніях для оцінювання тривалості розробки Java-застосунків для платформи ПК.

Особистий внесок здобувача. Кваліфікаційна робота є самостійно виконаною науковою працею. Усі наукові результати отримано автором особисто. У праці [13], яка опублікована у співавторстві, здобувачеві належить: побудова нелінійної регресійної моделі для оцінювання тривалості розробки

Java-застосунків для платформи ПК в залежності від трудомісткості розробки цих застосунків.

Апробація результатів роботи. Основні положення й результати кваліфікаційної роботи доповідалися на ІІ Всеукраїнській науково-практичній інтернет-конференції «Інформаційні технології: моделі, алгоритми, системи» (м. Миколаїв, 26-28 жовтня 2021 року).

Публікації. Основні положення й результати кваліфікаційної роботи опубліковано у 1 науковій праці – тезах конференції.

1 АНАЛІЗ МОДЕЛЕЙ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ РОЗРОБКИ JAVA-ЗАСТОСУНКІВ ДЛЯ ПЛАТФОРМИ ПК. ОБҐРУНТУВАННЯ НЕОБХІДНОСТІ ПРОВЕДЕННЯ ДОСЛІДЖЕННЯ

1.1 Аналіз існуючих моделей для оцінювання тривалості розробки Java-застосунків для платформи ПК

Основними методами для оцінювання тривалості розробки програмних застосунків, в тому числі Java-застосунків для платформи ПК є оцінювання за аналогами, експертне оцінювання та параметричне оцінювання (алгоритмічні методи та методи, що базуються на математично-статистичних моделях тривалості) [6].

Оцінювання за аналогами

Суть методу оцінювання за аналогами полягає в визначенні аналогічних програмних проєктів і оцінюванню тривалості, за рахунок визначення відмінностей між попередньою і новою розробкою [14]. Перевагою таких методів, в порівнянні з використанням методу експертного оцінювання, є базування отриманих оцінок на існуючих подібних розробках, а не тільки на досвіді експертів. При цьому виникає проблема щодо того, як порівнювати різні розробки та які основні характеристики повинні відстежуватися.

При оцінюванні за аналогами, в якості основи для оцінювання таких параметрів як бюджет, тривалість, розмір і складність нової розробки, використовуються ті ж параметри аналогічних проєктів з розробки програмного забезпечення [6]. Для оцінювання тривалості розробки програмних застосунків, цей метод бере реальну тривалість попередніх подібних розробок. При необхідності, отримана оцінка тривалості коректується в залежності від визначених відмінностей у складності розробки.

Зазвичай оцінювання тривалості за аналогами використовується на ранніх фазах розробки програмного забезпечення, коли кількість ґрунтовної інформації про проєкт обмежена.

В більшості випадків оцінювання за аналогами є дешевшим і вимагає менше часу в порівнянні з іншими методами, але при використанні цього методу достовірність оцінок зазвичай є нижчою. Оцінювання за аналогами може використовуватися при оцінюванню як проєкту повністю, так і його частин, а також може застосовуватися разом з іншими методами для оцінювання. Найбільш надійним оцінювання за аналогам буде у випадках, коли попередні операції схожі не тільки за формою, а і по суті, особливо якщо члени команди, що роблять оцінювання проєкту, мають потрібний досвід [6].

Експертне оцінювання тривалості розробки програмних застосунків

Експертне оцінювання тривалості розробки програмних застосунків опирається на досвід та інтуїцію, накопичені експертом в даній галузі [15, 16].

Згідно методу Delphi, на початку кожний експерт має самостійно оцінювати тривалість розробки, а потім для отриманих оцінок проводиться обговорення і уточнення, до того часу коли буде отримано загальне узгодження щодо тривалості розробки застосунку [17, 18].

Згідно методу Wideband Delphi [7, 18]), для прискорення оцінювання, експерти не пояснюють свої оцінки, а після проведення оцінювання проводиться зібрання, де обговорюються оцінки, для яких були отримані значні розбіжності.

При використанні експертного оцінювання для кожного нового застосунку доводиться знаходити кваліфікованих експертів, а достовірна оцінка може бути отримана лише в тому випадку, коли поточний проєкт і попередні є приблизно однакового розміру.

Алгоритмічні методи

В алгоритмічних методах використовується ітераційний підхід, що оснований на математичних моделях та формулах [17]. При цьому вхідними даними моделей будуть трудомісткість розробки програмних застосунків

(розрахована у людино-годинах чи людино-місяцях), а також інші параметри програмного проєкту, такі як вид програмного забезпечення, вид апаратного забезпечення, ступінь навичок менеджерів, ступінь досвіду команди розробки, та метод, що застосовується при розробці програмного забезпечення. При застосуванні алгоритмічних методів проявляються обмеження, коли ці методи застосовуються з недостовірними чи з неперевіреними вхідними даними. Алгоритмічні методи дозволяють оцінювати трудомісткість, тривалість та загальну вартість проєктів [17]. Алгоритмічні методи включають модель COCOMO 2.0, метод PERT, методи з використанням нейронних мереж, методи критичного ланцюга і критичного шляху.

Методи, які базуються на математично-статистичних моделях тривалості

До методів, які базуються на математично-статистичних моделях тривалості відносяться лінійна і множинна регресії та їх модифікації, в тому числі COCOMO [7], ISBSG [8], та моделі на основі перетворення Джонсона [9-12].

Моделі COCOMO [7] (Constructive Cost Model, модель витрат розробки) є нелінійними регресійними рівняннями для оцінювання вартості, тривалості та трудомісткості розробки проєктів з розробки програмного забезпечення. Коефіцієнти рівнянь регресії *COCOMO* були обчислені за емпіричними даними, зібраними для різних типів проєктів:

- organic (органічні) – 2-50 тисяч рядків коду програми (невеликі застосунки);
- semi-detached (напів-розділені) – 50-300 тисяч рядків коду програми (компілятори, прикладні системи);
- embedded (вбудовані) – понад 300 тисяч рядків коду програми (мережі АТМ, системи контролю повітряного руху, оборонні системи).

Вхідними даними моделей COCOMO при оцінюванні тривалості програмних застосунків є трудомісткість розробки цих застосунків в людино-місяцях. Враховуючи, що 1 людино-місяць дорівнює 152 людино-годинам, нелінійні регресійні рівняння COCOMO для оцінювання тривалості розробки

програмних застосунків в залежності від трудомісткості в людино-годинах можна записати наступним чином:

$$D = 0{,}371 \cdot E^{0{,}38} \text{ (органічний тип застосунків),} \quad (1.1)$$

$$D = 0{,}431 \cdot E^{0{,}35} \text{ (напів-розділений тип застосунків),} \quad (1.2)$$

$$D = 0{,}501 \cdot E^{0{,}32} \text{ (вбудований тип застосунків),} \quad (1.3)$$

де D – тривалість розробки програмного застосунку (вимірюється в місяцях);
E – трудомісткість розробки програмного застосунку (вимірюється в людино-годинах).

При побудові нелінійних регресійних моделей COCOMO виконувалась нормалізація емпіричних даних тривалості розробки програмних застосунків та трудомісткості розробки цих застосунків із використанням логарифмічного перетворення.

Нелінійні рівняння регресії (1.1), (1.2) та (1.3) показані на рисунку 1.1.

Моделі ISBSG [8] є нелінійними регресійними рівняннями для оцінювання тривалості розробки програмних застосунків для платформ mainframe, midrange та PC без використання мови програмування як ознаки кластеризації даних. Незалежною змінною (фактором) в цих рівняннях є трудомісткість розробки програмного забезпечення.

При побудові нелінійних регресійних моделей ISBSG виконувалась нормалізація емпіричних даних тривалості розробки програмних застосунків та трудомісткості розробки цих застосунків із використанням логарифмічного перетворення.

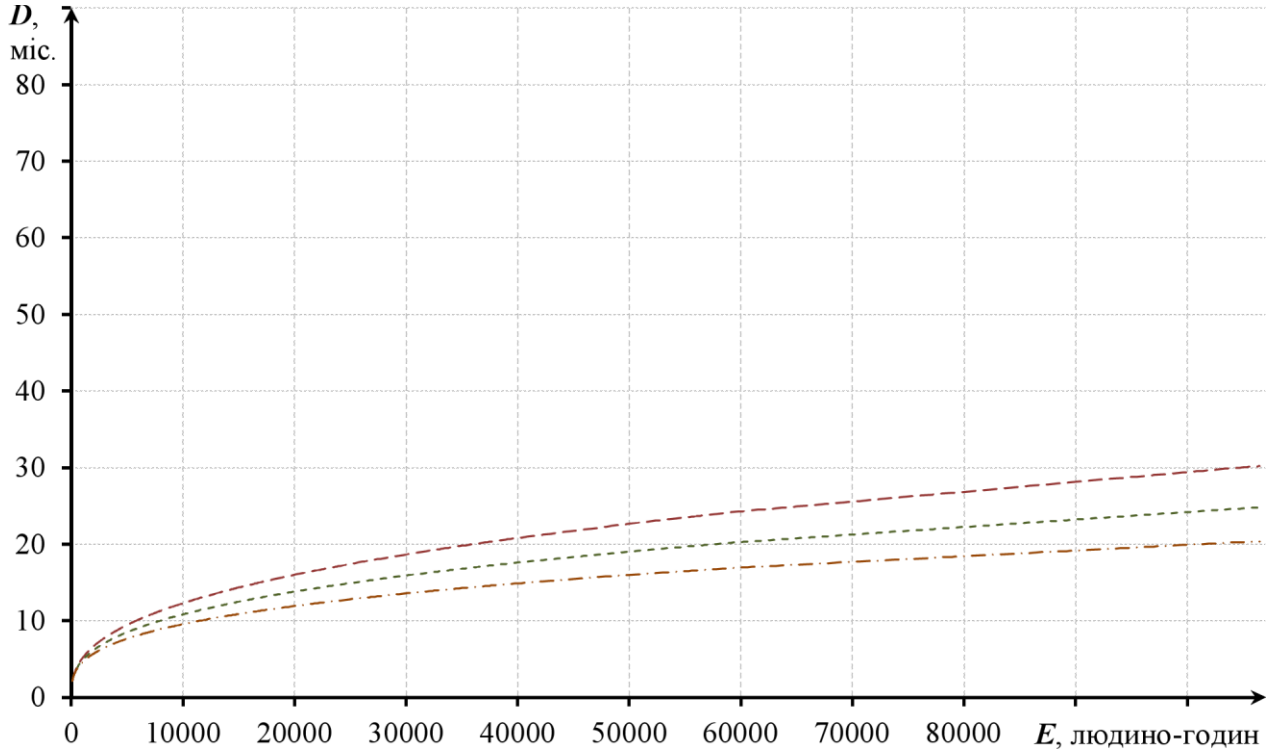

Рисунок 1.1 Нелінійні рівняння регресій COCOMO:

– – – – для organic (органічного) типу застосунків;

- - - - для semi-detached (напів-розділеного) типу застосунків;

–·–·– для embedded (вбудованого) типу застосунків

Моделі на основі перетворення Джонсона [9-12] також були побудовані для платформ mainframe, midrange та PC. Окрім нелінійних рівнянь регресії, для цих моделей побудовано рівняння довірчих інтервалів нелінійних регресій та рівняння інтервалів прогнозування нелінійних регресій.

В таблиці 1.1 узагальнено переваги і недоліки розглянутих методів для оцінювання тривалості розробки програмних застосунків.

Таблиця 1.1 – Переваги та недоліки методів та моделей оцінювання тривалості розробки застосунків

Назва	Переваги	Недоліки
Експертна оцінка	Гнучкість методу легко пристосовується від застосунку до застосунку, щоб підвищити достовірність оцінок тривалості розробки	Недостатній досвід у експертів може значно знизити достовірність оцінювання
Оцінювання за аналогами	Базування отриманих оцінок на існуючих подібних розробках	Складність визначення того, як порівнювати різні розробки та які основні характеристики повинні відстежуватися
Алгоритмічні методи	Ітеративно виконуючи алгоритм можна отримати прийнятну достовірність. Можна легко адаптувати до змін у вхідних значеннях.	Для неверифікованих і невідкаліброваних вхідних даних достовірність оцінок може бути дуже низькою
Методи, що базуються на математично-статистичних моделях тривалості	Легкість використання. Базуються на науковому підґрунті	Вимагають значну кількість емпіричних даних для побудови. Можуть не враховувати реальний розподіл даних

1.2 Обґрунтування необхідності проведення досліджень

Так як кросплатформна мова програмування Java є однією з найбільш поширених мов для платформи ПК, то оцінювання тривалості розробки Java-

застосунків для платформи ПК, є важливою задачею програмної інженерії. Це пов'язано з тим, що достовірні оцінки тривалості розробки є одним з факторів, які дозволяють успішно завершувати програмні проєкти [1].

Звіти Standish Group та інші дослідження показують, що в більшості випадків проєкти з розробки програмного забезпечення виконуються пізніше запланованих термінів [2-5], тобто існує проблема низької достовірності відповідного оцінювання.

Ця проблема пов’язана зі складністю побудови достовірних моделей для оцінювання тривалості розробки програмних застосунків. Складність пояснюється багатьма факторами, в тому числі:

- відсутність, мала кількість чи застарілість історичних оцінок, необхідних при побудові моделей для оцінювання тривалості розробки програмних застосунків;
- розробка програмних застосунків базується на взаємопов’язаних факторах, які прямо чи опосередковано впливають на тривалість розробки програмних застосунків.

Аналіз існуючих моделей для оцінювання тривалості розробки програмного забезпечення, в тому числі Java-застосунків для платформи ПК показав, що найчастіше відповідне оцінювання проводиться з використанням нелінійних регресійних моделей для оцінювання тривалості розробки в залежності від трудомісткості розробки програмних застосунків, таких як COCOMO [7], ISBSG [8], та моделі на основі перетворення Джонсона [9-12]. Ці моделі були побудовані за застарілими емпіричними даними, моделі COCOMO і ISBSG не дозволяють проводити оцінювання довірчих інтервалів та інтервалів прогнозування тривалості розробки програмного забезпечення, крім того поділ моделей тільки по платформі є недостатнім.

Емпіричні дані тривалості та трудомісткості розробки програмних застосунків мають закони розподілу, що відрізняються від нормального [7-12]. Через це неможливо виконати побудову адекватних регресійних моделей, які дозволять оцінювати довірчий інтервал тривалості розробки Java-застосунків

для платформи ПК. Деколи в такому випадку закон розподілу приймається за нормальний, або може бути використання непараметричного підходу. Використання непараметричного підходу для оцінювання довірчих інтервалів чи виконання інтервального оцінювання в припущенні про нормальний закон розподілу емпіричних даних [22] знижуватиме достовірність оцінювання тривалості розробки Java-застосунків для платформи ПК.

Вказані фактори приводять до необхідності побудови нелінійної регресійної моделі для оцінювання тривалості розробки Java-застосунків для платформи ПК з використанням нормалізуючого перетворення. Така модель також має дозволяти оцінювати довірчий інтервал та інтервал прогнозування тривалості розробки Java-застосунків для платформи ПК, так як інтервальні оцінки дозволять врахувати невизначеність точкової оцінки.

1.3 Висновки до розділу 1

В даному розділі було проаналізовано існуючі моделі для оцінювання тривалості розробки програмних застосунків, в тому числі Java-застосунків для платформи ПК. Проведено обґрунтування необхідності проведення досліджень та побудови нелінійної регресійної моделі для оцінювання тривалості розробки Java-застосунків для платформи ПК.

1. Визначено, що розробка більшості програмних застосунків виконується із затримками, часто з дуже значними, з перевищенням запланованих термінів розробки.
2. Показано, що один із факторів, який позначається на вдалому завершенні процесу розробки програмних застосунків, дозволяючи вчасно його завершувати, є достовірне оцінювання тривалості розробки цих застосунків. Достовірність оцінювання тривалості проєктів з розробки програмного

забезпечення в значній мірі залежить від моделей та методів оцінювання тривалості.

3. Для оцінювання тривалості розробки Java-застосунків для платформи ПК можуть використовуватися оцінювання за аналогами, експертне оцінювання, параметричне оцінювання.
4. Найчастіше для оцінювання тривалості розробки Java-застосунків для платформи ПК використовуються нелінійні регресійні моделі тривалості розробки в залежності від трудомісткості розробки цих застосунків (COCOMO, ISBSG, моделі на основі перетворення Джонсона). Ці моделі побудовані за застарілими емпіричними даними, у відповідних рівняннях відсутня випадкова змінна, а моделі COCOMO і ISBSG не дозволяють проводити оцінювання довірчих інтервалів та інтервалів прогнозування тривалості розробки застосунків.
5. Для підвищення достовірності оцінювання тривалості розробки Java-застосунків для платформи ПК потрібно побудувати відповідну нелінійну регресійну модель за новими емпіричними даними, з додаванням в модель випадкової змінної.

2 ПОБУДОВА НЕЛІНІЙНОЇ РЕГРЕСІЙНОЇ МОДЕЛІ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ РОЗРОБКИ JAVA-ЗАСТОСУНКІВ ДЛЯ ПЛАТФОРМИ ПК

Необхідність побудови нелінійної регресійної моделі для оцінювання тривалості розробки Java-застосунків для платформи ПК виникає через те, що закон розподілу емпіричних даних тривалості та трудомісткості розробки цих застосунків відрізняється від нормального закону розподілу. Тому побудувати адекватну лінійну регресійну модель неможливо (побудова лінійної регресійної моделі та знаходження інтервальних оцінок статистичних моментів випадкової величини можливі лише у випадках, коли закон розподілу емпіричних даних є нормальним).

Щоб побудувати адекватну нелінійну регресійну модель для оцінювання тривалості розробки Java-застосунків для платформи ПК, та довірчий інтервал і інтервал прогнозування тривалості розробки Java-застосунків для платформи ПК, потрібно використати метод побудови рівнянь нелінійної регресії та метод побудови довірчого інтервалу регресії на основі нормалізуючих перетворень [19, 20].

2.1 Метод побудови рівнянь нелінійної регресії та метод побудови довірчого інтервалу регресії на основі нормалізуючих перетворень

Метод побудови рівнянь нелінійної регресії та метод побудови довірчого інтервалу регресії на основі нормалізуючих перетворень [19, 20] полягає в тому, що емпіричні дані спочатку нормалізуються з застосуванням нормалізуючого перетворення, після чого будується лінійна регресійна модель

за нормалізованими даними, а далі будується нелінійна регресійна модель з застосуванням перетворення, зворотного до нормалізуючого.

Щоб було можливо побудувати лінійну регресійну модель за нормалізованими даними, повинні виконуватись наступні умови:

1. Нормалізовані дані мають мати багатовимірний нормальний розподіл.
2. Відстань Махаланобіса для кожної пари значень має бути меншою чи дорівнювати критичному значенню.
3. Розподіл залишків лінійної регресії повинен бути нормальним.
4. Нормалізовані значення емпіричних даних не повинні виходити за інтервал передбачення лінійної регресійної моделі.

Для виконання вказаних умов з емпіричних даних потрібно видалити викиди, для чого виконуються наступні кроки:

1. Обчислюється значення багатовимірної асиметрії нормалізованих емпіричних даних. Якщо отримане значення більше, ніж критичне значення, то з емпіричних даних видаляється пара значень з найбільшою відстанню Махаланобіса, після чого повторюємо п. 1, поки значення багатовимірної асиметрії нормалізованих емпіричних даних не буде відповідати нормальному закону.
2. Обчислюється значення багатовимірного ексцесу нормалізованих емпіричних даних. Якщо отримане значення більше, ніж критичне значення, то з емпіричних даних видаляється пара значень з найбільшою відстанню Махаланобіса та переходимо до п. 1.
3. Для кожного набору значень обчислюється відстань Махаланобіса. Якщо для одного чи декількох наборів даних відстань Махаланобіса перевищує відповідне критичне значення, то з емпіричних даних видаляється пара значень з найбільшою відстанню Махаланобіса та переходимо до п. 1.
4. За нормалізованими даними будуємо лінійну регресійну модель та перевіряємо закон розподілу залишків. Якщо розподіл не є нормальним, то з емпіричних даних видаляємо пару значень з залишком найбільшим за модулем та переходимо до п. 1.

5. Якщо одне або більше нормалізованих значень даних виходить за інтервал передбачення лінійної регресії для нормалізованих даних, то ці значення видаляються та переходимо до п. 1.

Після видалення викидів за отриманою лінійною регресійною моделлю будується нелінійна регресійна модель із застосуванням перетворення, зворотного до нормалізуючого.

2.2 Нормалізуючі перетворення для нормалізації випадкових величин

Необхідність використання нормалізуючих перетворень при побудові регресійних моделей виникає через те, що закон розподілу емпіричних даних тривалості та трудомісткості проєктів з розробки програмного забезпечення відрізняється від нормального закону розподілу [7-12], і тому неможливо побудувати адекватні лінійні регресійні моделі. Виконання інтервального оцінювання в припущенні про нормальний закон розподілу емпіричних даних [22] буде знижувати достовірність оцінювання тривалості розробки Java-застосунків для платформи ПК

Існує декілька нормалізуючих перетворень, які можуть бути використаними, щоб нормалізувати випадкову величину.

Десятковий логарифм. Нормалізацію випадкової величини x за перетворенням у вигляді десяткового логарифму виконуємо за

$$z = \log_{10} x \tag{2.1}$$

де z є нормованою нормально розподіленою випадковою величиною, яка має нульове математичне сподіванням і одиничну дисперсію;

x є випадковою величиною, яка нормалізується.

Перетворення Бокса-Кокса. Нормалізацію випадкової величини x за перетворенням Бокса-Кокса [21] виконуємо за

$$z = \begin{cases} \dfrac{x^{\lambda} - 1}{\lambda}, & якщо\, \lambda \neq 0; \\ \ln(x), & якщо\, \lambda = 0. \end{cases} \tag{2.2}$$

де z є нормованою нормально розподіленою випадковою величиною, яка має нульове математичне сподіванням і одиничну дисперсію;

x є випадковою величиною, яка нормалізується;

λ – параметр розподілу.

Для перетворення (2.2) значення параметру λ отримується через максимізацію логарифму функції правдоподібності

$$l(\lambda) = C - \frac{n}{2}\ln\sum_{i=1}^{n}\frac{\left(z_i - \bar{z}\right)^2}{n} + (\lambda - 1)\sum_{i=1}^{n}\ln(x_i), \tag{2.3}$$

де C є константою, яка визначається з умови нормування;

$\bar{z} = \dfrac{1}{n}\sum_{i=1}^{n} z_i$.

Перетворення Джонсона. В загальному випадку нормалізуюче перетворення Джонсона має вигляд (2.4) [23].

$$z = \gamma + \eta q(x, \varphi, \lambda), \tag{2.4}$$

де q – функція, яка залежить від вибраної сім'ї розподілу Джонсона: сім'я S_L (2.5), сім'я S_B (2.6), сім'я S_U (2.7).

$$\mathrm{q}(\mathrm{x}, \varphi, \lambda) = \ln\left(\frac{\mathrm{x} - \varphi}{\lambda}\right), \quad \mathrm{x} > \varphi \text{ (сім'я } S_L); \tag{2.5}$$

$$q(x,\varphi,\lambda)=\ln\left(\frac{x-\varphi}{\lambda+\varphi-x}\right),\qquad \varphi<x<\varphi+\lambda \text{ (сім'я } S_B\text{)};\tag{2.6}$$

$$q(x,\varphi,\lambda)=Arsh\left(\frac{x-\varphi}{\lambda}\right),\qquad -\infty<x<+\infty \text{ (сім'я } S_U\text{)},\tag{2.7}$$

де γ, η, λ, φ – параметри перетворень;

$$Arsh\left(\frac{x-\varphi}{\lambda}\right)=\ln\left[\frac{x-\varphi}{\lambda}+\sqrt{\left(\frac{x-\varphi}{\lambda}\right)^2+1}\right].$$

Сім'я перетворення Джонсона підбирається за значеннями оцінок асиметрії у квадраті A^2 та ексцесу ε емпіричних даних, що нормалізуються [24].

Параметри перетворення Джонсона γ, η, λ і φ знаходять за емпіричним розподілом [23] або використовуючи непараметричний шляхом рішення задачі математичного програмування (2.8):

$$\hat{\theta}=\arg\min_{\theta}\left\{A_z^2+(\varepsilon_z-3)^2+\bar{z}^2+(S_z^2-1)^2\right\},\tag{2.8}$$

де $\hat{\theta}$ – оцінка вектору невідомих параметрів;

θ – вектор невідомих параметрів, $\theta=\{\gamma,\eta,\lambda,\varphi\}$;

A_z – коефіцієнт асиметрії величини z, що обчислюється за (2.11);

ε_z – коефіцієнт ексцесу величини z, що обчислюється за (2.12);

$\bar{z}$ – вибіркове середнє величини z, $\bar{z}=\frac{1}{n}\sum_{i=1}^{n}z_i$;

S_z^2 – незміщена вибіркова дисперсія величини z, $S_z^2=\frac{1}{n-1}\sum_{i=1}^{n}(z_i-\bar{z})^2$;

z_i – i-значення нормалізованої випадкової величини z.

При використанні непараметричного підходу виникає необхідність обчислення статистичних характеристик емпіричних даних.

Вибіркове середнє випадкової величини x обчислюється як

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i , \tag{2.9}$$

де n - це кількість емпіричних значень випадкової величини x;

x_i – i-значення випадкової величини x.

Незміщена вибіркова дисперсія випадкової величини x обчислюється як

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2 , \tag{2.10}$$

де n - це кількість емпіричних значень випадкової величини x;

x_i – i-значення випадкової величини x;

$\bar{x}$ – вибіркове середнє випадкової величини x, що обчислюється за (2.9).

Тоді, коефіцієнт асиметрії випадкової величини x обчислюється як

$$A = \frac{n}{(n-1)(n-2)}\sum_{i=1}^{n}(x_i - \bar{x})^3 \frac{1}{\left(S^2\right)^{3/2}} , \tag{2.11}$$

де n - це кількість емпіричних значень випадкової величини x;

x_i – i-значення випадкової величини x;

$\bar{x}$ – вибіркове середнє випадкової величини x, що обчислюється за (2.9);

S^2 – незміщена вибіркова дисперсія величини x, що обчислюється за (2.10).

Коефіцієнт ексцесу випадкової величини x обчислюється як

$$\varepsilon = \frac{\left(n^2 - 2n + 3\right)\sum_{i=1}^{n}(x_i - \bar{x})^4 + (2n-3)\frac{3}{n}\left[\sum_{i=1}^{n}(x_i - \bar{x})^2\right]^2}{(n-1)(n-2)(n-3)\left(S^2\right)^2} , \tag{2.12}$$

де n - це кількість емпіричних значень випадкової величини x;

x_i – i-значення випадкової величини x;

$\bar{x}$ – вибіркове середнє випадкової величини x, що обчислюється за (2.9);

S^2 – незміщена вибіркова дисперсія величини x, що обчислюється за (2.10).

Вибір нормалізуючого перетворення, у випадку з використанням методу побудови рівнянь нелінійної регресії на основі нормалізуючих перетворень робиться виходячи зі значень R^2, PRED(0,25) та MMRE для побудованих нелінійних регресійних моделей. Серед перетворень (2.1), (2.2), (2.4) обирається нормалізуюче перетворення, при використанні якого вказані характеристики будуть кращими.

2.3 Вибір емпіричних даних для побудови нелінійної регресійної моделі тривалості розробки Java-застосунків для платформи ПК

Побудувати нелінійну регресійну модель для оцінювання тривалості розробки Java-застосунків для платформи ПК, можна використавши емпіричні дані по завершеним програмним проєктам з репозитарію International Software Benchmarking Standards Group (ISBSG) релізу 2021 року. Використання цього репозитарію пов'язане з великою кількістю програмних проєктів, що в ньому присутні (в тому числі Java-застосунків для платформи ПК), для яких відомі якість зібраних статистичних даних (що дозволить при побудові використати тільки статистичні дані з найвищою якістю), а також з присутністю даних щодо тривалості та трудомісткості розробки. З вказаного репозитарію було взято емпіричні дані тривалості розробки та трудомісткості розробки 39 Java-застосунків для платформи ПК. Значення тривалості знаходяться в інтервалі 1-15 місяців, значення трудомісткості знаходяться в інтервалі 105-47493 людино-годин.

Нехай випадкова величина D – це емпіричні значення тривалості розробки Java-застосунків для платформи ПК (в місяцях), а випадкова величина

E – це емпіричні значення трудомісткості проєктів з розробки програмного забезпечення (в людино-годинах). Загальна кількість застосунків $n_0 = 39$.

Статистичні характеристики випадкової величини E: математичне сподівання $\overline{E} = 4501{,}09$, середньоквадратичне відхилення $S_E = 11506{,}53$.

Статистичні характеристики випадкової величини D: математичне сподівання $\overline{D} = 3{,}89$, середньоквадратичне відхилення $S_D = 2{,}66$.

Багатовимірний розподіл випадкових величин E та D відрізняється від нормального закону розподілу (асиметрія та ексцес значно відрізняються від відповідних характеристик нормального розподілу: асиметрія $\beta_1 = 18{,}4$, ексцес $\beta_2 = 24{,}6$).

2.4 Побудова нелінійної регресійної моделі тривалості розробки Java-застосунків для платформи ПК

З випадкових величин D та E, використовуючи нормалізуюче перетворення (2.1), було отримано випадкові величини z_D та z_E. Для нормалізованих емпіричних даних було виконано кроки, наведені в підрозділі 2.1, щоб видалити викиди та побудувати лінійну регресійну модель.

Після виконання кроків 1-3 багатовимірний розподіл величин z_D та z_E буде нормальним, що дозволяє побудувати лінійну регресійну модель для нормалізованих значень:

$$z_D = b_0 + b_1 z_E + \varepsilon, \tag{2.13}$$

де b_0 і b_1 - коефіцієнти нелінійної регресійної моделі,

ε - випадкова помилка нелінійної регресійної моделі, що має нормальний розподіл.

Значення коефіцієнтів b_0 і b_1 лінійної регресійної моделі (2.13) знаходяться за методом найменших квадратів [27]:

$$b_1 = \frac{\overline{z_E z_D} - \overline{z_E} \cdot \overline{z_D}}{\overline{z_E^2} - \overline{z_E}^2};\ b_0 = \overline{z_D} - b_1 \cdot \overline{z_E}, \tag{2.14}$$

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

Рівняння верхньої та нижньої границь довірчого інтервалу лінійної регресійної моделі (2.13) задається як [27]

$$\left[\bar{z}_D\right] = b_0 + b_1 z_E \pm t_{\alpha/2,n-2} \cdot \sqrt{s_{z_D}^2} \sqrt{\frac{1}{n} + \frac{(z_E - \bar{z}_E)^2}{S_{z_E}}}, \tag{2.15}$$

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 $n-2$.

Рівняння верхньої та нижньої границь інтервалу прогнозування лінійної регресійної моделі (2.13) задається як [27]

$$\left[z_D\right] = zb_0 + b_1 z_E \pm t_{\alpha/2,n-2} \cdot \sqrt{s_{z_D}^2} \sqrt{1 + \frac{1}{n} + \frac{(z_E - \bar{z}_E)^2}{S_{z_E}}}, \tag{2.16}$$

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 $n-2$.

Результати побудови лінійної регресійної моделі з використанням перетворення (2.1) показано в таблиці 2.1.

Таблиця 2.1 – Результати побудови лінійної регресійної моделі

Кількість застос-унків	Багатовимірні асиметрія та ексцес	Відстань Махала-нобіса	Коефіцієнти рівняння лінійної регресії	Статистичні характеристики випадкової похибки	Кількість викидів та їх обгрунтування
n=39	β_1 = 1,54; β_2 = 10,94	11,81	-	-	1 Відстань Махаланобіса
n=38	β_1 = 0,94; β_2 = 10,56	11,48	-	-	1 Відстань Махаланобіса
n=37	β_1 = 2,98; β_2 = 9,59	10,01	-	-	1 Багатовимірний розподіл не є нормальним
n=36	β_1 = 2,47; β_2 = 9,43	10,36	b_0 = -0,425; b_1 = 0,329	M=-0,0; q=0,0972; A=1,2106; E=5,0234; Xi2=10,5311; Xi2Kr=5,991	1 Розподіл випадкової похибки не є нормальним
n=35	β_1 = 1,84 β_2 = 8,62	9,04	b_0 = -0,44; b_1 = 0,331	M=-0,0; q=0,0827; A=0,7309; E=3,6282; Xi2=4,0466; Xi2Kr=5,991	1 Вихід за межі інтервалу передбачення
n=34	β_1 = 1,28; β_2 = 7,94	7,9	b_0 = -0,423; b_1 = 0,323	M=-0,0; q=0,0725; A=0,368; E=2,5492; Xi2=1,423; Xi2Kr=5,991	2 Вихід за межі інтервалу передбачення
n=32	β_1 = 1,22; β_2 = 8,41	9,49	b0=-0,396; b1=0,311	M=-0,0; q=0,0623; A=0,1283; E=2,3775; Min=-0,1183; Max=0,128; Xi2=2,9033; Xi2Kr=5,991	-

Таким чином, після 7 повторень кроків, наведених в підрозділі 2.1, було видалено викиди та побудовано лінійну регресійну модель за нормалізованими даними. Отримані за (2.14) значення коефіцієнтів лінійної регресійної моделі після видалення всіх викидів були такими: b_0 = -0,423 і b_1 = 0,323. Значення t - розподілу Стьюдента, що використовувалось при знаходженні довірчого інтервалу (2.15) та інтервалу прогнозування (2.16) нелінійної регресії та було $t_{(0,05/2),(34-2)}$ = 2,042. Рівняння лінійної регресії, 95% довірчий інтервал та 95% інтервал прогнозування для моделі (2.13) з приведеними значеннями коефіцієнтів показані на рисунку 2.1.

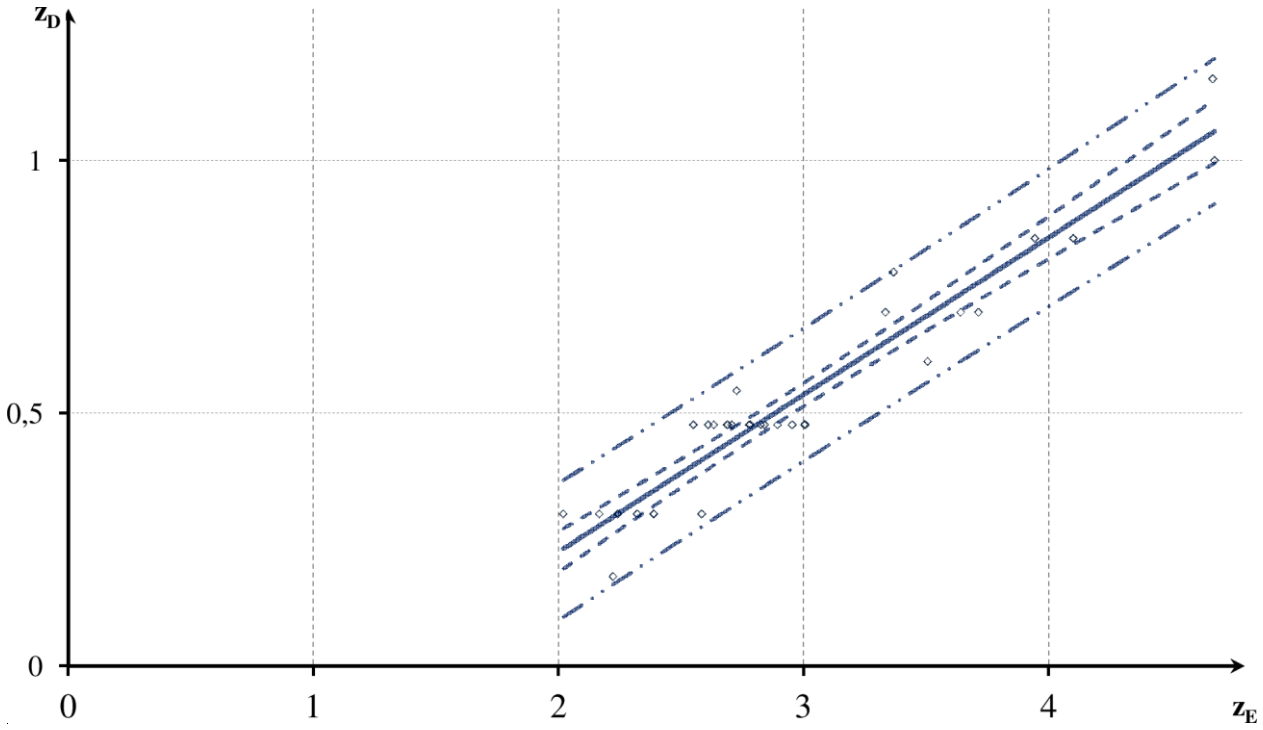

Рисунок 2.1 – Лінійна регресійна модель для нормалізованих даних на основі логарифмічного перетворення:

◦ - нормалізовані емпіричні дані (z_E – нормалізована трудомісткість; z_D – нормалізована тривалість);

—— - лінійна регресія;

– – - границі 95% довірчого інтервалу лінійної регресії;

—·· - границі 95% інтервалу прогнозування лінійної регресії

Враховуючи (2.13) і те, що для нормалізації емпіричних даних тривалості розробки Java-застосунків для платформи ПК та емпіричних даних

трудомісткості розробки цих застосунків використовувалося перетворення (2.1), то нелінійна регресійна модель тривалості розробки Java-застосунків для платформи ПК в залежності від трудомісткості розробки цих застосунків може бути представлена як

$$D = 10^{b_0+\varepsilon} E^{b_1}, \tag{2.17}$$

де b_0 і b_1 - коефіцієнти рівняння лінійної регресії (2.13) для нормалізованих значень.

Враховуючи (2.15), довірчий інтервал $[\overline{D}(E)]$ нелінійної регресійної моделі (2.17) задається як

$$\left[\overline{D}\right] = 10^{\left[\bar{z}_D\right]}, \tag{2.18}$$

де $\left[\bar{z}_D\right]$ задається як (2.15).

Враховуючи (2.16), інтервал прогнозування $[D(E)]$ нелінійної регресійної моделі (2.17) задається як

$$\left[D\right] = 10^{\left[z_D\right]}, \tag{2.19}$$

де $\left[z_D\right]$ задається як (2.16).

Побудована нелінійна регресійна модель (2.17) для оцінювання тривалості розробки Java-застосунків для платформи ПК в залежності від трудомісткості розробки цих застосунків показана на рисунку 2.2.

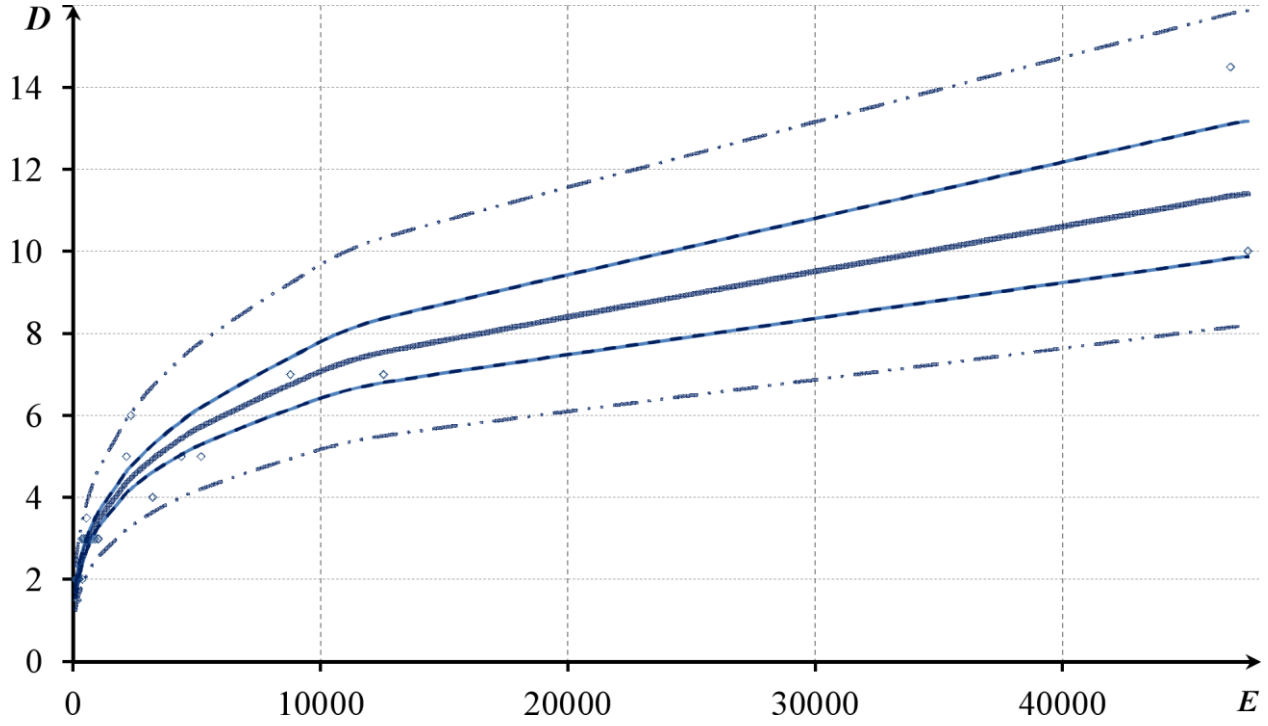

Рисунок 2.2 – Нелінійна регресійна модель для оцінювання тривалості розробки Java-застосунків для платформи ПК в залежності від трудомісткості розробки цих застосунків на основі логарифмічного перетворення:

◇ - емпіричні дані (E – трудомісткість; D – тривалість);

——— - нелінійна регресія;

– – – - границі 95% довірчого інтервалу нелінійної регресії;

–·–·– - границі 95% інтервалу прогнозування нелінійної регресії

Для порівняння нелінійних регресійних моделей побудованих на основі різних нормалізуючих перетворень можна використовувати значення коефіцієнта детермінації R^2:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(D_i - \hat{D}(E_i)\right)^2}{\sum_{i=1}^{n}\left(D_i - \overline{D}\right)^2}, \qquad (2.20)$$

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, $\overline{D} = \frac{1}{n}\sum_{i=1}^{n} D_i$.

Кращим є рівняння регресії, для якого значення R^2 є більшим.

Ще одним з критеріїв, який може бути використаний для порівняння нелінійних регресійних моделей є рівень передбачення (відносна кількість оцінок, похибка яких не перевищує l %):

$$PRED(l) = \frac{100}{n}\sum_{i=1}^{n}\begin{cases}1, & якщо\ MRE_i \le l\\ 0, & якщо\ MRE_i > l\end{cases}, \qquad (2.21)$$

де n - це кількість емпіричних даних; $MRE_i = \left|\frac{D_i - \hat{D}(E_i)}{D_i}\right|$;

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

Кращим є рівняння регресії, для якого значення $PRED(l)$ є більшим. При порівнянні звичайно використовують $PRED(0{,}25)$.

Характеристики побудованої нелінійної регресійної моделі (2.17) для оцінювання тривалості розробки Java-застосунків для платформи ПК в залежності від трудомісткості розробки цих застосунків були наступні: коефіцієнт детермінації (2.20) $R^2 = 0{,}91$; відносна кількість значень, які було оцінено з відносною похибкою до 25% (2.21) $PRED(0{,}25) = 0{,}91$; MMRE = 0,12.

Нелінійна регресійна модель тривалості розробки Java-застосунків для платформи ПК в залежності від трудомісткості розробки цих застосунків на основі десяткового логарифму (2.1) має кращі характеристики R^2, $PRED(0{,}25)$ та MMRE, ніж аналогічні моделі на основі перетворення Джонсона (2.4) чи на основі перетворення Бокса-Кокса (2.2).

2.5 Висновки до розділу 2

В даному розділі було побудовано нелінійну регресійну модель для оцінювання тривалості розробки Java-застосунків для платформи ПК в залежності від трудомісткості розробки цих застосунків на основі логарифмічного перетворення.

1. Розглянуто рівнянь нелінійної регресії та метод побудови довірчого інтервалу регресії на основі нормалізуючих перетворень. Розглянуто нормалізуючі перетворення, які можна використати разом з даним методом при побудові нелінійних регресійних моделей (логарифмічне перетворення, перетворення Бокса-Кокса та перетворення Джонсона).
2. Вибрано емпіричні дані з репозитарію International Software Benchmarking Standards Group (ISBSG) релізу 2021 року (39 проєктів) для побудови нелінійної регресійної моделі для оцінювання тривалості розробки Java-застосунків для платформи ПК.
3. Побудовано лінійну регресійну модель для нормалізованих значень тривалості та трудомісткості розробки Java-застосунків для платформи ПК, видаливши при цьому викиди з емпіричних даних.
4. Побудовано нелінійну регресійну модель для оцінювання тривалості розробки Java-застосунків для платформи ПК в залежності від трудомісткості розробки цих застосунків на основі логарифмічного перетворення. Побудована модель на основі десяткового логарифму має кращі характеристики R^2, $PRED(0{,}25)$ та MMRE, ніж аналогічні моделі на основі перетворення Джонсона (2.4) чи на основі перетворення Бокса-Кокса (2.2).

3 РОЗРОБКА ПРОЄКТУ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ РОЗРОБКИ JAVA-ЗАСТОСУНКІВ ДЛЯ ПЛАТФОРМИ ПК

На даний час при оцінюванні тривалості розробки Java-застосунків для платформи ПК можуть бути використані існуючі програми (наприклад, Comparative Estimating Tool [28] вартістю 152 USD за рік, Microsoft Project 2021 Standard [29] вартістю 679,99 USD та інші). Вказані програми мають високу вартість ліцензії. Тому розробка програми для реалізації побудованої моделі для оцінювання тривалості розробки Java-застосунків для платформи ПК має практичну цінність.

Розробка програми для оцінювання тривалості розробки Java-застосунків для платформи ПК буде проводитись згідно з технічним завданням, яке приведено в додатку А. Розробка вказаної програми включає розробку ескізного проєкту, технічного проєкту та робочого проєкту.

3.1 Розробка ескізного проєкту програми для оцінювання тривалості розробки застосунків

Моделювання варіантів використання

Програма передбачає наявність єдиного актора – користувача програми, який буде використовувати всі доступні в програмі функції.

Варіанти використання програми приведено в таблиці 3.1.

Таблиця 3.1 – Виявлені варіанти використання

Основні актори	**Найменування**	**Формулювання**
Користувач	Внесення назви застосунку	Вносить назву застосунку
Користувач	Внесення трудомісткості застосунку	Вносить значення трудомісткості розробки застосунку
Користувач	Внесення довірчої ймовірності	Вносить довірчу ймовірність, з якою буде виконуватися оцінювання довірчого інтервалу та інтервалу прогнозування розробки тривалості застосунку
Користувач	Внесення кількості моделювань тривалості	Вносить кількість моделювань тривалості розробки застосунку
Користувач	Оцінювання тривалості	Оцінює тривалість, довірчий інтервал та інтервал передбачення тривалості розробки застосунку
Користувач	Моделювання тривалості	Моделює значення тривалості розробки застосунку
Користувач	Збереження результатів оцінювання та моделювання	Зберігає результати оцінювання тривалості та моделювання тривалості застосунку в базу даних
Користувач	Перегляд результатів виконаних оцінювань та моделювань	Переглядає збережені в базі даних результати оцінювання та моделювання тривалості розробки застосунків

Проведемо конкретизацію основних варіантів використання:

1. *Найменування:* Внесення даних про застосунок

Головна дійова особа: Користувач

Зв'язок з іншими варіантами використання: Узагальнює прецеденти «Внесення назви застосунку», «Внесення трудомісткості застосунку», «Внесення довірчої ймовірності», «Внесення кількості моделювань тривалості».

Короткий опис:

Варіант використання «Внесення даних про застосунок» відображає чотири варіанти операцій і дає можливість внести інформацію про Java-застосунок у відповідні поля. Для варіанту «Внесення назви застосунку» – це назва застосунку. Для варіанту «Внесення трудомісткості застосунку» – це значення трудомісткості розробки Java-застосунку. Для варіанту «Внесення довірчої ймовірності» – це значення довірчої ймовірності для оцінювання довірчого інтервалу та інтервалу прогнозування тривалості розробки Java-застосунку. Для варіанту «Внесення кількості моделювань тривалості» – це кількість моделювань тривалості розробки Java-застосунку.

Потоки подій: Прецедент розпочинається при переході користувачем в відповідні комірки для введення даних вікна програми.

Базовий потік – Внесення даних про застосунок:

1) користувачем заповнюється поле «Назва застосунку» назвою застосунку;

2) користувачем заповнюється поле «Трудомісткість» значенням трудомісткості розробки застосунку;

3) користувачем заповнюється поле «Довірча ймовірність» значенням довірчої ймовірності;

4) користувачем заповнюється поле «Кількість моделювань» кількістю моделювань тривалості розробки Java-застосунку.

Спеціальні вимоги:

1) значення трудомісткості застосунку має бути цілим додатним числом;

2) значення довірчої ймовірності має бути дійсним додатним числом;

3) значення кількості моделювань має бути цілим додатним числом від 1 до 10.

Альтернативі потоки:

1) при внесенні у п. 2 значення, що є меншим ніж 1, буде виведено повідомлення про помилку та запропоновано повторити внесення значення; помилкове значення не буде збережене;

2) при внесенні у п. 3 значення, що є меншим ніж 1 чи більшим ніж 100, буде виведено повідомлення про помилку та запропоновано повторити внесення значення; помилкове значення не буде збережене;

3) при внесенні у п. 4 значення, що є меншим ніж 1 чи більшим ніж 10, то буде виведено повідомлення про помилку та запропоновано повторити внесення значення; помилкове значення не буде збережене.

2. *Найменування:* Оцінювання та моделювання тривалості

Головна дійова особа: Користувач

Зв'язок з іншими варіантами використання: Узагальнює прецеденти «Оцінювання тривалості» та «Моделювання тривалості».

Короткий опис:

Варіант використання «Оцінювання та моделювання тривалості» включає оцінювання тривалості розробки Java-застосунку, результатом якого є оцінка тривалості, довірчий інтервал, інтервал прогнозування тривалості розробки Java-застосунку, а також включає моделювання тривалості розробки Java-застосунку, результатом чого будуть змодельовані значення тривалості.

Потоки подій: Прецедент розпочинається після натискання користувачем на кнопку «Оцінити тривалість».

Базовий потік – Виконати оцінювання та моделювання тривалості:

1) Користувачем натискається кнопка «Виконати оцінювання та моделювання тривалості»;

2) У вікні будуть відображені: Оцінка тривалості розробки Java-застосунку, Довірчий інтервал нелінійної регресії, Інтервал прогнозування тривалості розробки Java-застосунку, Змодельовані значення тривалості розробки Java-застосунку.

3. *Найменування:* Збереження результатів оцінювання та моделювання

Головна дійова особа: Користувач

Зв'язок з іншими варіантами використання: відсутній.

Короткий опис:

Варіант використання «Збереження результатів оцінювання та моделювання» дозволяє зберегти результат оцінювання тривалості та моделювання тривалості розробки Java-застосунку в базу даних.

Потоки подій: Прецедент розпочинається після натискання користувачем на кнопку «Зберегти результати».

Базовий потік – Збереження результатів оцінювання тривалості:

1) Користувачем натискається кнопка «Зберегти результати»;

2) Результат оцінювання тривалості розробки Java-застосунку (включає оцінку тривалості, границі довірчого інтервалу, границі інтервалу прогнозування) та змодельовані значення тривалості розробки Java-застосунку надсилаються до бази даних.

4. *Найменування:* Перегляд результатів виконаних оцінювань та моделювань

Головна дійова особа: Користувач

Зв'язок з іншими варіантами використання: відсутній.

Короткий опис:

Варіант використання «Перегляд результатів виконаних оцінювань та моделювань» дозволяє переглянути результати виконаних раніше оцінювань (оцінки тривалості, довірчого інтервалу нелінійної регресії, інтервалу прогнозування) та моделювань тривалості розробки Java-застосунків.

Потоки подій: Прецедент розпочинається після вибору користувачем пункту меню «Перегляд виконаних оцінювань та моделювань».

Діаграму варіантів використання показано на рисунку 3.1.

Рисунок 3.1 – Діаграма варіантів використання

Діаграму станів показано на рисунку 3.2. Основний стан програми для оцінювання тривалості розробки застосунків є показ головного вікна з очікуванням взаємодії користувача. При виконані дій користувачем відбувається перехід до інших станів.

Рисунок 3.2 – Діаграма станів

Моделювання потоків даних

Діаграму потоків даних (нотація Гейна-Сарсона) показано на рисунку 3.3.

Рисунок 3.3 – Діаграма потоків даних програми

Атрибути складних потоків даних діаграми 3.3. є наступними:

параметри нелінійної регресійної моделі – параметри b_0, b_1, $s^2_{z_D}$, $\bar{z}_E$, S_{z_E}, n моделей (2.18), (2.19);

- довірчий інтервал тривалості застосунку – нижня границя довірчого інтервалу тривалості розробки застосунку $[\overline{D}(E)]$ (2.18), верхня границя довірчого інтервалу тривалості розробки застосунку $[\overline{D}(E)]$ (2.18);
- інтервал прогнозування тривалості застосунку – нижня границя інтервалу прогнозування тривалості розробки застосунку [D(E)] (2.19), верхня границя довірчого інтервалу тривалості розробки застосунку [D(E)] (2.19);

- результати оцінювання тривалості – оцінка середнього значення тривалості розробки застосунку, довірчий інтервал та інтервал прогнозування тривалості розробки застосунку.

3.2 Розробка технічного проєкту програми для оцінювання тривалості розробки застосунків

Логічна модель бази даних

Таблиця параметрів нелінійних регресійних моделей для оцінювання тривалості розробки застосунків:

- ідентифікатор запису – число, первинний ключ;
- назва моделі – рядковий тип даних, заборонені дублювання;
- параметр b_0 моделей (2.17), (2.18), (2.19) – число;
- параметр b_1 моделей (2.17), (2.18), (2.19) – число;
- параметр n моделей (2.18), (2.19) – число;
- параметр $s^2_{z_D}$ моделей (2.18), (2.19) – число;
- параметр $\bar{z}_E$ моделей (2.18), (2.19) – число;
- параметр S_{z_E} моделей (2.18), (2.19) – число.

Таблиця з результатами оцінювань та моделювань тривалості розробки:

- ідентифікатор запису – число, первинний ключ;
- назва застосунку – рядковий тип даних;
- назва моделі – рядковий тип даних;
- трудомісткість – число;
- тривалість – число;
- значення верхньої границі довірчого інтервалу тривалості – число;
- значення нижньої границі довірчого інтервалу тривалості – число;

- значення верхньої границі інтервалу прогнозування тривалості – число;
- значення нижньої границі інтервалу прогнозування тривалості – число;
- змодельовані значення тривалості розробки застосунку – рядковий тип даних;
- дата та час оцінювання – дата/час.

3.3 Розробка робочого проєкту програми для оцінювання тривалості розробки застосунків

Вибір мови програмування та системи управління базою даних

Критерії, що використовуються при виборі мови програмування для написання програми для оцінювання тривалості розробки застосунків:

- працездатність програми в операційних системах Windows, Unix, MacOS;
- наявність бібліотек та компонентів для побудови графічного інтерфейсу користувача;
- наявність бібліотек з математичними та статистичними функціями, необхідними для проведення оцінювання тривалості застосунків з використанням побудованої нелінійної регресійної моделі;
- наявність бібліотеки для доступу до БД.

За вказаними критеріями мовою програмування для написання програми було обрано об'єктно-орієнтовану мову програмування Java. Її перевагами є наступні можливості:

- доступ до СУБД з використанням Java Persistence API (JPA). JPA – це специфікація прикладного програмного інтерфейсу для роботи з реляційною СУБД в Java-застосунках;
- мова запитів Java Persistence Query Language (JPQL) – платформо-незалежна об'єктно-орієнтована мова запитів (частина специфікації JPA). Використання JPQL дозволяє зміну СУБД не змінюючи запити SQL;

- безкоштовні інтегровані середовища розробки програм (IDE), що дозволяє скоротити тривалість написання програми;

дозвіл на безкоштовне розповсюдження та використання Java в складі програми.

Системи управління базами даних (СУБД), що забезпечують можливості доступу та адміністрування баз даних, можуть мати клієнт-серверну архітектуру. Така архітектура потребує додаткових дозволів в операційній системі для встановлення та запуску СУБД. У випадку розробки програми для оцінювання тривалості розробки застосунків, критеріями вибору СУБД є можливість роботи без використання архітектури клієнт-сервер та підтримка мови запитів SQL.

За вказаними критеріями для роботи з базою даних вибрана система керування базами даних SQLite. В СУБД SQLite вся інформація зберігається в єдиному файлі у бінарному форматі. Використання цієї СУБД підходить найкраще, так як відсутня необхідність збереження великої кількості інформації в базі даних. В той же час СУБД SQLite за необхідності з часом може бути змінена на іншу СУБД з клієнт-серверною архітектурою, яка підтримує мову запитів SQL.

Фізична модель бази даних

Фізична модель таблиці estimation_models, призначеної для зберігання параметрів нелінійних регресійних моделей тривалості програмних застосунків включає наступні поля:

- id – INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT;
- name – 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 estimations, призначеної для зберігання результатів оцінювань та моделювань тривалості програмних застосунків включає наступні поля:

- id – INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT;
- name – TEXT NOT NULL UNIQUE;
- model_name – TEXT NOT NULL;
- effort – NUMERIC NOT NULL;
- duration – NUMERIC NOT NULL;
- duration_confidence_interval_top – NUMERIC NOT NULL;
- duration_confidence_interval_bottom – NUMERIC NOT NULL;
- duration_estimation_interval_top – NUMERIC NOT NULL;
- duration_estimation_interval_bottom – NUMERIC NOT NULL;
- generated_duration – TEXT NOT NULL,
- created – DATE NOT NULL.

Відображення структури класів програми класів програми «Оцінювання тривалості розробки застосунків» показано на рисунку 3.4.

SQL запит для додавання в базу даних параметрів нелінійної регресійної моделі для оцінювання тривалості розробки Java-застосунків для платформи ПК, побудованої в 2 розділі:

```
INSERT INTO estimation_models values
(1, 'Java-застосунки для платформи ПК',
-0.3963751428950725, 0.3107761581101411, 32,
0.12033593450909232, 2.9683499661246313, 14.301774637586515
);
     ;
```

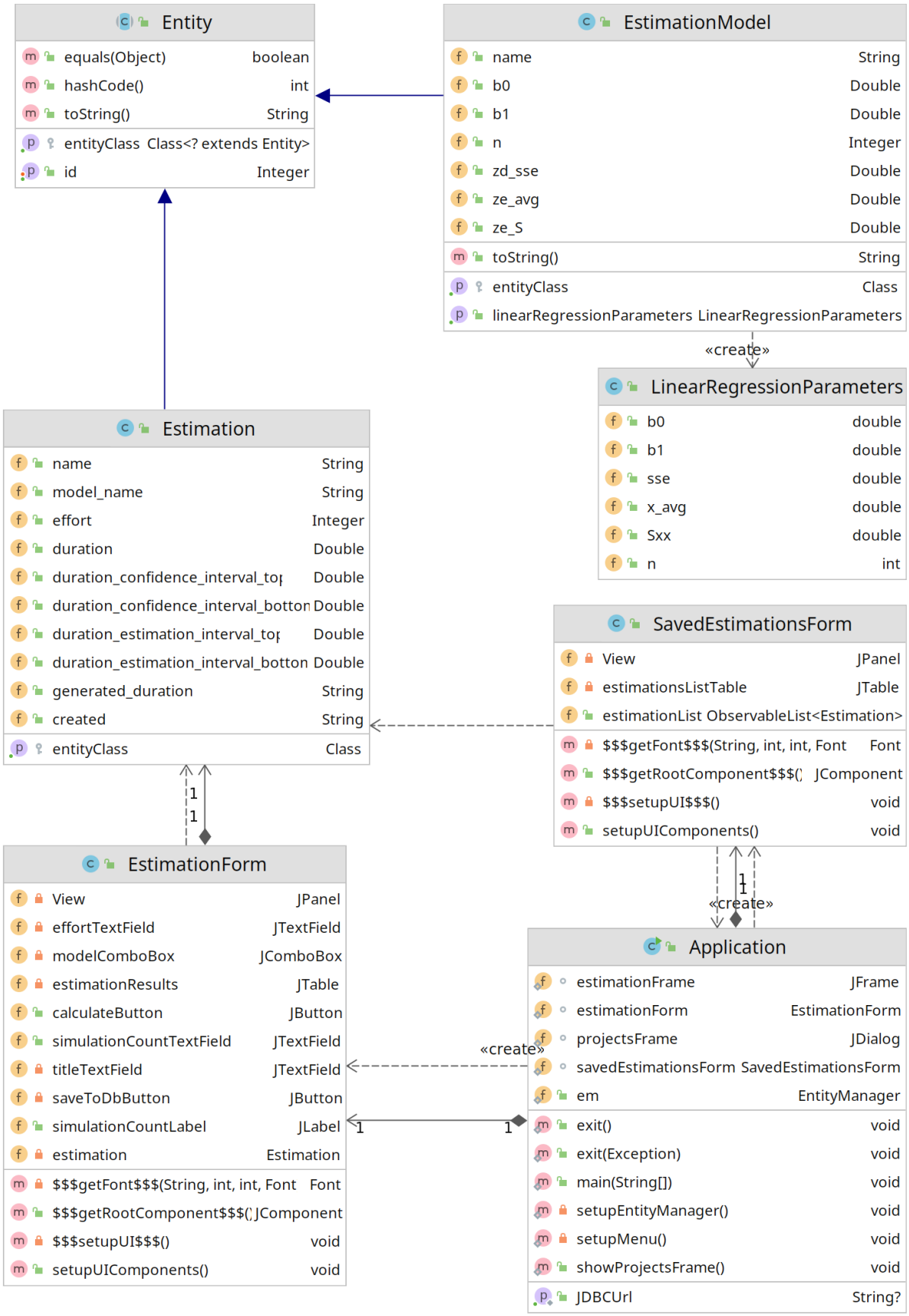

Рисунок 3.4 – Діаграма класів програми

Відразу після запуску програми в методі main() класу Application (головного класу програми) відбувається підключення до бази даних, після чого відображується головне вікно програми. Перед відображенням до головного вікна програми додаються меню, поля для введення вхідних даних та кнопки.

Вигляд головного вікна програми відразу після її запуску показано на рисунку 3.5.

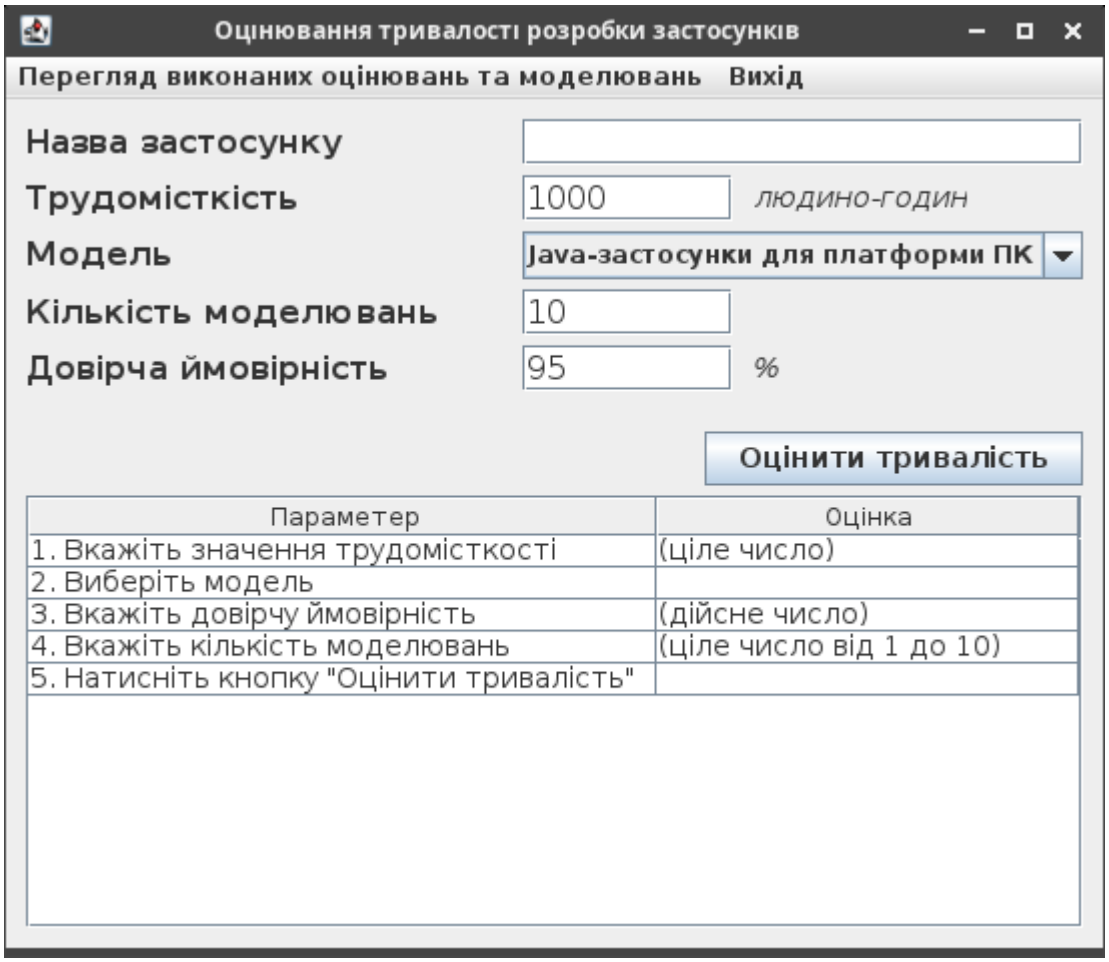

Рисунок 3.5 – Вигляд головного вікна програми

Інструкція програми «Оцінювання тривалості розробки застосунків» приведена в додатку В.

Випробування програми для оцінювання тривалості розробки Java-застосунків для платформи ПК

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

Таблиця 3.2 – Результати тестування методом "чорного ящика"

Тест	**Вхідні значення**	**Результат**
Ведення трудомісткості	Значення не введено або введено невірно	При спробі оцінювання тривалості виводиться повідомлення
	Значення введено вірно	Можливо виконати оцінювання тривалості
Ведення назви застосунку	Значення не введено або введено невірно	При спробі оцінювання тривалості виводиться повідомлення
	Значення введено вірно	Можливо виконати оцінювання тривалості
Ведення довірчої ймовірності	Значення не введено або введено невірно	При спробі оцінювання тривалості виводиться повідомлення
	Значення введено вірно	Можливо виконати оцінювання тривалості
Ведення кількості моделювань	Значення не введено або введено невірно	При спробі оцінювання тривалості виводиться повідомлення
	Значення введено вірно	Можливо виконати оцінювання тривалості

Тестуванням за методом "білого ящика" було перевірено коректність побудови всіх частин програми та правильність їх взаємодії. Випробування програми було виконано згідно з методикою випробувань (додаток Г). Випробування показали, що розроблена програма повністю відповідає поставленим вимогам.

3.4 Висновки до розділу 3

В даному розділі було розроблено проєкт програми для оцінювання тривалості розробки Java-застосунків для платформи ПК.

1. Розроблено ескізний проєкт програми для оцінювання тривалості розробки застосунків (моделювання варіантів використання, моделювання потоків даних).
2. Розроблено технічний проєкт програми для оцінювання тривалості розробки застосунків (логічну модель бази даних).
3. Розроблено робочий проєкт програми для оцінювання тривалості розробки застосунків. Для написання програми було обрано об'єктно-орієнтовану мову програмування Java. Для роботи з базою даних вибрана СУБД SQLite та побудовано фізичну модель БД.
4. Тестування показало, що розроблена програма повністю відповідає поставленим вимогам.

4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВІД РОЗРОБКИ ТА ВПРОВАДЖЕННЯ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ РОЗРОБКИ JAVA-ЗАСТОСУНКІВ ДЛЯ ПЛАТФОРМИ ПК

Вступ

Програму для оцінювання тривалості розробки Java-застосунків для платформи ПК можна використовувати в галузі розробки та забезпечення якості ПЗ, а також управління програмними проєктами. Ефективність розробки таких програм визначається ступенем їх позитивного впливу на підприємство.

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

4.1 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:

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

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:

$$З_е = 1587 * 3.19 = 5062.53 \text{ грн.}$$

Експлуатаційні витрати на ПК на рік складуть:

$$З_{зр} = 3750 + 750 + 5062.53 = 9562.53 \text{ грн.}$$

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

$$З_р = 42870.56 + 9562.53 = 52433.09 \text{ грн.}$$

4.2 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:

$$E_{рік} = \Delta C_п – E_п * k \qquad (5.3)$$

де $\Delta C_п$ – вивільнені кошти після впровадження продукту (144000 грн.) мінус експлуатаційні витрати (52433.09 грн.) = 91566.91 грн.;

$E_п$ – 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:

$$T = k / \Delta C_n = 42870.56 / 91566.91 = 0.47 \text{ року}$$

Отже, термін окупності ПЗ становить приблизно 6 місяців.

4.3 Висновки до розділу 4

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

5 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].

5.1 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. 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.

5.2 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 IT-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:

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

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

5.3 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.

6 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. 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.

6.1 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.

6.2 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 їжі), скорочення споживання та використання меншої кількості упаковки.

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

Для цього слід вимикати з електромережі комп’ютери та обладнання, з якими ніхто не працює. Також необхідно робити перерви в роботі з комп'ютером.

ВИСНОВКИ

В кваліфікаційній роботі було підвищено достовірність оцінювання тривалості розробки Java-застосунків для платформи ПК.

В процесі виконання кваліфікаційної роботи одержано наступні результати:

1. Проаналізовано існуючі моделі для оцінювання тривалості розробки програмних застосунків, в тому числі Java-застосунків для платформи ПК. Проведено обґрунтування необхідності проведення досліджень та побудови нелінійної регресійної моделі для оцінювання тривалості розробки Java-застосунків для платформи ПК.
2. Побудовано нелінійну регресійну модель для оцінювання тривалості розробки Java-застосунків для платформи ПК в залежності від трудомісткості розробки цих застосунків на основі логарифмічного перетворення. Побудована модель на основі десяткового логарифму має кращі характеристики R^2, $PRED(0{,}25)$ та MMRE, ніж аналогічні моделі на основі перетворення Джонсона чи на основі перетворення Бокса-Кокса.
3. Розроблено проєкт програми для оцінювання тривалості розробки Java-застосунків для платформи ПК. Тестування показало, що розроблена програма повністю відповідає поставленим вимогам.
4. Виконано розрахунок економічної ефективності від розробки та впровадження програми для оцінювання тривалості розробки Java-застосунків для платформи ПК. Показано, що розробка такого застосунку є економічно обґрунтованою.
5. Розглянуто питання з охорони праці та охорони навколишнього середовища.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДеМарко, Т. Человеческий фактор. Успешные проекты и команды, 3-е издание [Текст] / Т. ДеМарко , Т. Листер // Символ-Плюс, 2014. – 288 с. – ISBN 978-5-93286-217-9.

2. 5 Main Causes Of Project Delay In IT Industry And How To Avoid Them [Electronic resource] // Режим доступу : https://www.projectpractical.com/5-main-causes-of-project-delay-in-it-industry-and-how-to-avoid-them/. Retrieved September, 2021.

3. Standish Group 2015 Chaos Report - Q&A with Jennifer Lynch [Electronic resource] // S. Hastie, S. Wojewoda. – InfoQ, 2015. – Режим доступу : https://www.infoq.com/articles/standish-chaos-2015. Retrieved September, 2021.

4. Guide to the Software Engineering Body of Knowledge (SWEBOK Guide): Version 3.0. [Text] // IEEE Computer Society, 2014.

5. McConnell, S. Rapid Development: Taming Wild Software Schedules [Text] / S. McConnell // Redmond, WA : Microsoft Press, 1996. – 660 p.

6. A guide to the project management body of knowledge. Fifth edition [Text] // Project Management Institute, 2013. – 589 p.

7. COCOMO [Electronic resource] // Режим доступу : https://ru.wikipedia.org/wiki/COCOMO. Retrieved September, 2021.

8. Oligny, S. Exploring the relation between effort and duration in software engineering projects [Text] / S. Oligny, P. Bourque, A. Abran, B. Fournier // In proc. of the World Computer Congress, Aug. 2000. – Pp. 175-178.

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

10. Prykhodko, S. B. Developing PC Software Project Duration Model based on Johnson transformation [Text] / S. B. Prykhodko, A. V. Pukhalevich // Proceedings of the 12th International Conference Modern Problems of Radio Engineering, Telecommunications and Computer Science TCSET'2014, Lviv-Slavske, Ukraine. – Lviv : Polytechnic National University, 2014. – Pp. 114-116.

11. Приходько, С. Б. Розробка нелінійних регресійних моделей тривалості програмних проектів на основі перетворення Джонсона [Текст] / С. Б. Приходько, А. В. Пухалевич // Збірник наукових праць НУК. – Миколаїв : НУК, 2014. – № 2 (2014). – С. 76-80. – ISSN 2313-0415.

12. Приходько, С. Б. Confidence interval estimation of PC software project duration regression based on Johnson transformation [Текст] / С. Б. Приходько, А. В. Пухалевич // Радіоелектронні і комп'ютерні системи. – Харків : Харківський авіаційний інститут, 2014. – № 2 (66). - С. 104-107. – ISSN 1814-4225.

13 Приходько, С.Б. Нелінійні регресійні моделі для оцінювання тривалості розробки програмних застосунків, що створюються мовами Java та Visual Basic для ПК [Електронний ресурс] / С. Б. Приходько, А. В. Пухалевич, І.П. Халанчук // Інформаційні технології: моделі, алгоритми, системи (ITMAS – 2021) : Матеріали II Всеукраїнської науково-практичної інтернет конференції (26-28 жовтня 2021 р.). – Миколаїв: НУК, 2021. – С. 36-37. Режим доступу: http://itconf.nuos.edu.ua/2021/proceedings/.

14. Software Development Time Estimation: How to calculate hours for building product MVP [Electronic resource] // Режим доступу : https://inoxoft.com/blog/software-development-time-estimation-how-to-calculate-hours-for-building-product-mvp/. Retrieved September, 2021.

15. Faria, P. Expert Judgment in Software Estimation During the Bid Phase of a Project - An Exploratory Survey [Text] / Faria P., Miranda E. // IWSM/Mensura. IEEE Computer Society, 2012. – Pp. 126–131.

16. Callahan, J. Reducing software product development time [Text] / J. Callahan, B. Moretton // International Journal of Project Management. – Vol. 19, No.1, January 2001.

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

18. Wiegers, C. Stop Promising Miracles [Text] / C. Wiegers // Software Development Magazine. – Vol. 8, No.2, February 2000. – P. 49-54.

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

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

21. Box, G. E. P. An analysis of transformations [Text] / G. E. P. Box, D. R. Cox // Journal of the Royal Statistical Society. Series B (Methodological). – Vol. 26. No. 2, 1964. – Pp. 211-252.

22. Орлов, А. И. Прикладная статистика [Текст] / А. И. Орлов. – Москва : "Экзамен", 2004. – 656 с.

23. Приходько, С. Б. Аналитические зависимости для выбора семейств распределения Джонсона [Текст] / С. Б. Приходько, Л. Н. Макарова // Комп`ютерні науки: освіта, наука, практика: матеріали Міжнародної науково-технічної конференції. – Миколаїв: НУК, 2014. – С.152-154.

24. Коваленко, І. І. Сучасні методи статистичного аналізу даних: Навчальний посібник [Текст] / І. І. Коваленко, С. Б. Приходько, Л. О. Латанська. – Миколаїв : НУК, 2011. – 192 с.

25. Приходько, С. Б. Аналитическая зависимость для выбора распределения Джонсона семейства SL [Текст] / С. Б. Приходько, Л. Н. Макарова // Вестник ХНТУ. – Херсон : ХНТУ, 2012. – №2 (45). – С. 101-104.

26. Приходько, С. Б. Інтервальне оцінювання математичного сподівання тривалості програмних проектів [Текст] / С. Б. Приходько, А. В. Пухалевич // Тези доповідей IX Міжнародної науково-практичної конференції «Сучасні інформаційні технології в економіці та управлінні підприємствами, програмами та проектами». – Харків, 2011. – С. 214-216.

27. Pardoe, Iain. Applied regression modeling [Text] / Iain Pardoe. – Wiley, 2012. – 325 p.

28. ISBSG Productivity Data Query Tool [Electronic resource] // International Software Benchmarking Standards Group. – Режим доступу : https://www.isbsg.org/isbsg-productivity-data-query/. Retrieved October, 2021.

29. Microsoft Project 2021 Standard [Electronic resource] // Microsoft Corporation. – Режим доступу : https://www.microsoft.com/uk-ua/microsoft-365/p/project-standard-2021/cfq7ttc0hh09. Retrieved October, 2021.

ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ

Вступ

Назва програми: «Оцінювання тривалості розробки застосунків».

Область застосування:

Область застосування програми обмежена галуззю розробки програмного забезпечення.

1 Підстави для розробки

Підставою для розробки програмного продукту є завдання на кваліфікаційну (магістерську) роботу «Нелінійна регресійна модель для оцінювання тривалості розробки Java-застосунків для ПК та розробка програми для її реалізації».

2 Призначення програми:

2.1 Функціональне призначення

Даний програмний продукт призначений для оцінювання тривалості розробки застосунків.

2.2 Експлуатаційне призначення

Програма може використовуватися для оцінювання тривалості розробки застосунків.

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 розробки застосунків;

- Оцінювання довірчого інтервалу середнього значення тривалості розробки застосунків;
- Оцінювання інтервалу прогнозування тривалості розробки застосунків;
- Моделювання тривалості розробки застосунків;
- Збереження результатів оцінювання та моделювання тривалості в базу даних;
- Перегляд результатів раніше проведених оцінювань та моделювань тривалості розробки програмних застосунків.

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, змодельовані значення) повинні виводитися в таблиці на формі програми.

3.2 Вимоги до надійності

Надійне (стійке) функціонування програми має бути забезпечено валідацію вхідних даних та відображенням повідомлень про некоректні дані при їх виявленні.

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;

Настанова користувача.

5 Техніко-економічні показники

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

6 Порядок контролю та приймання

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

ДОДАТОК Б ОПИС ПРОГРАМИ «ОЦІНЮВАННЯ ТРИВАЛОСТІ РОЗРОБКИ ЗАСТОСУНКІВ»

Програма для оцінювання тривалості розробки застосунків в залежності від трудомісткості представляє собою застосунок на мові програмування Java, який дозволяє виконувати наступні дії:

- Оцінювання середнього значення тривалості розробки застосунку з використанням нелінійної регресійної моделі в залежності від трудомісткості;
- Оцінювання довірчого інтервалу середнього значення тривалості розробки застосунку;
- Оцінювання інтервалу прогнозування тривалості розробки застосунку;
- Моделювання тривалості розробки застосунку;
- Збереження результатів оцінювання та моделювання в базу даних;
- Перегляд результатів раніше проведених оцінювань та моделювань тривалості розробки програмних застосунків.

ДОДАТОК В НАСТАНОВА КОРИСТУВАЧА КОМП'ЮТЕРНОЇ ПРОГРАМИ «ОЦІНЮВАННЯ ТРИВАЛОСТІ РОЗРОБКИ ЗАСТОСУНКІВ»

1 ВСТУП

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

Дана програма призначена для точкового та інтервального оцінювання тривалості розробки Java-застосунків для платформи ПК в залежності від трудомісткості, а також моделювання тривалості розробки Java-застосунків для платформи ПК. Програма є незалежним програмним продуктом, що має графічний інтерфейс, який дозволяє введення даних, перегляд результатів оцінювання тривалості розробки Java-застосунків для платформи ПК та збереження отриманих результатів в базу даних.

2 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ПРОГРАМУ

2.1 Найменування програми

Найменування програми - «Оцінювання тривалості розробки застосунків».

2.2 Мови програмування, на яких написана програма

Програма написана на мові програмування Java.

2.3 Призначення програми

Оцінювання тривалості розробки Java-застосунків для платформи ПК в залежності від трудомісткості, та моделювання тривалості розробки Java-застосунків для платформи ПК для вказаних платформ.

2.4 Можливості програми

Програма виконує наступні функції:

- Оцінювання середнього значення тривалості розробки застосунку з використанням нелінійної регресійної моделі в залежності від трудомісткості;
- Оцінювання довірчого інтервалу середнього значення тривалості проєкту з розробки програмного забезпечення;
- Оцінювання інтервалу прогнозування тривалості розробки застосунку;
- Моделювання тривалості розробки застосунку;
- Збереження результатів оцінювання і моделювання в базу даних;
- Перегляд результатів раніше проведених оцінювань та моделювань тривалості розробки Java-застосунків для платформи ПК.

2.5 Обмеження області застосування програми

- Область застосування програми обмежена галуззю 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 рис. 1.

Стартове вікно містить наступні поля для ведення вхідних даних:

- Текстове поле «Назва застосунку» – дозволяє вказати назву застосунку для подальшого збереження результатів оцінювання і моделювання в базу даних;
- Текстове поле «Трудомісткість» – дозволяє вибрати значення трудомісткості, яке буде основою для оцінювання тривалості розробки Java-застосунків для платформи ПК;
- Випадаючий список «Модель» – при наявності декількох моделей, збережених в базі даних, дозволяє вибрати модель для оцінювання;

- Текстове поле «Довірча ймовірність» – дозволяє вказати довірчу ймовірність для інтервального оцінювання.
- Текстове поле «Кількість моделювань» – дозволяє вказати кількість моделювань тривалості розробки застосунків.

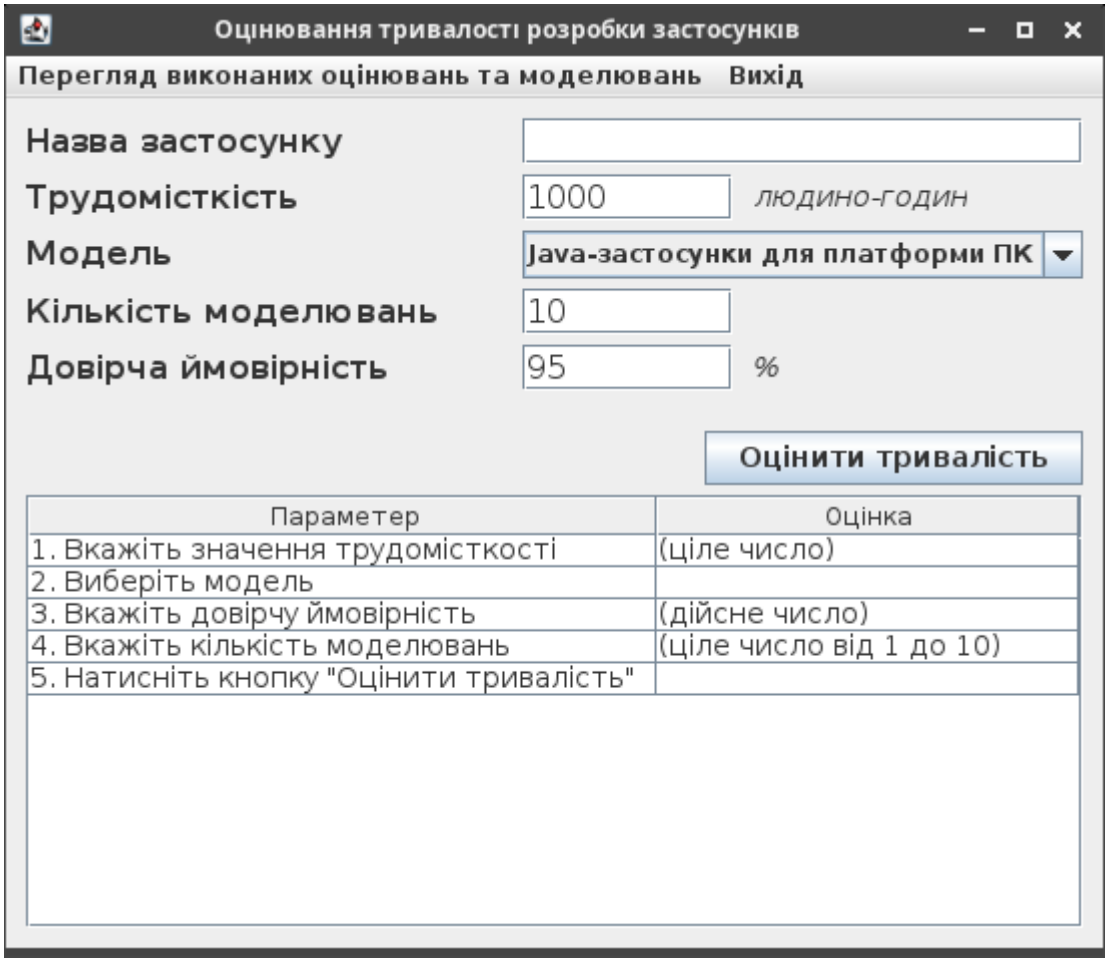

Рисунок 1 – Загальний вигляд стартового вікна програми

4.2 Результати оцінювання та моделювання тривалості розробки Java-застосунків для платформи ПК

Після введення вхідних даних у вказані поля стартового вікна користувачу потрібно натиснути кнопку «Оцінити тривалість». Результати оцінювання тривалості розробки застосунку (Оцінка тривалості, Довірчий інтервал нелінійної регресії, Інтервал прогнозування) та вказана кількість змодельованих значень тривалості розробки застосунків будуть показані в таблиці в нижній частині вікна програми (рис. 2).

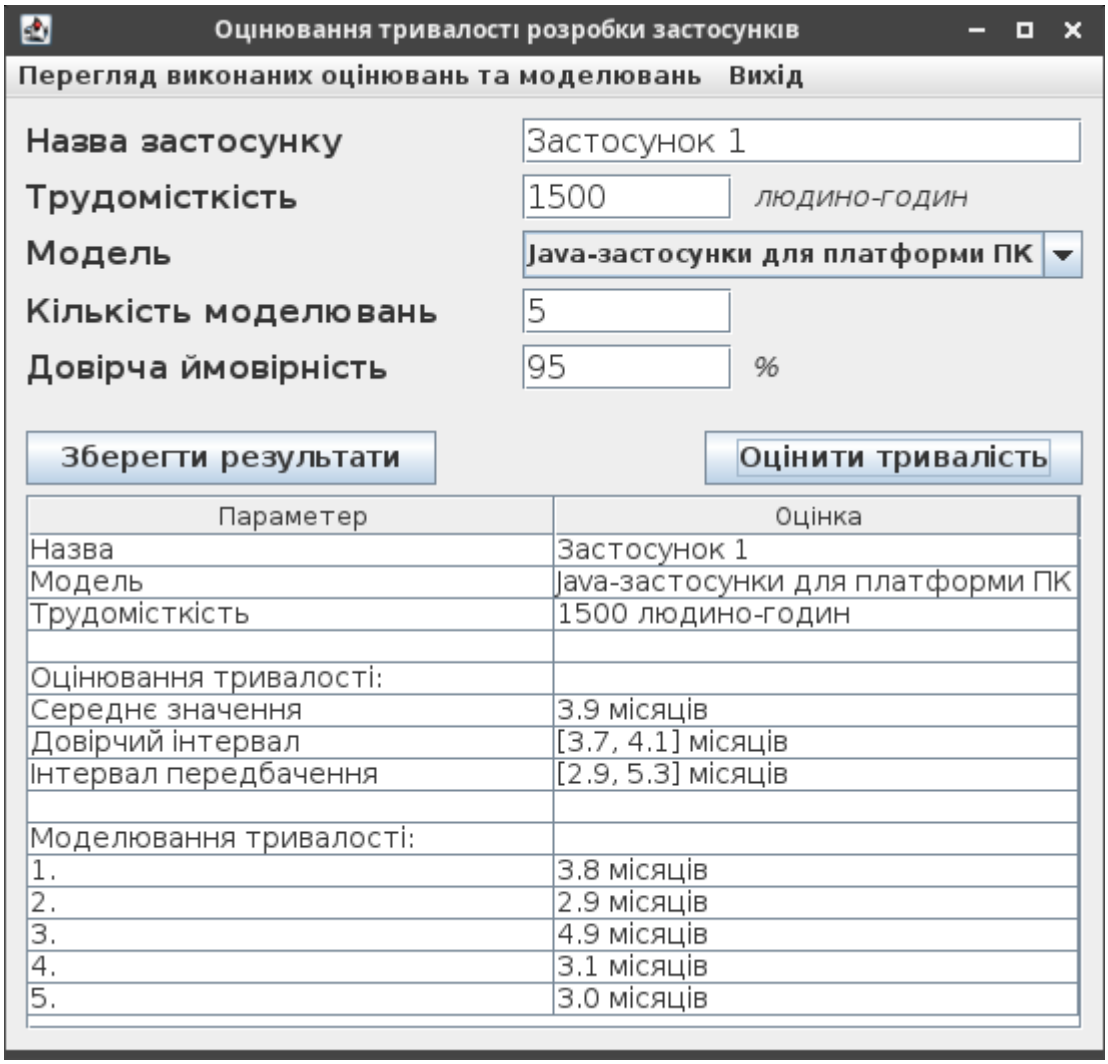

Рисунок 2 – Результати оцінювання та моделювання тривалості розробки Java-застосунків для платформи ПК

При необхідності результати оцінювання і моделювання можна зберегти в базу даних, натиснувши на кнопку «Зберегти результати». Для перегляду раніше збережених результатів оцінювання і моделювання (рис. 3) потрібно вибрати пункт меню «Перегляд виконаних оцінювань та моделювань» вверху програми.

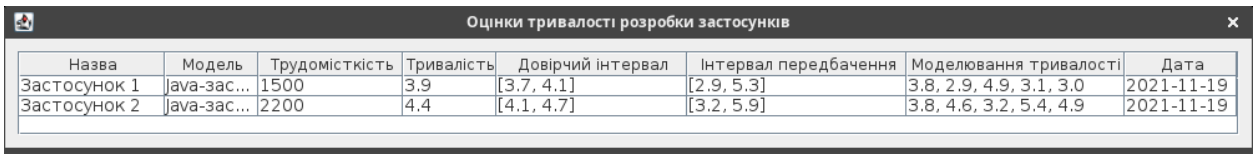

Рисунок 3 – Перегляд збережених результатів оцінювання і моделювання тривалості розробки Java-застосунків для платформи ПК

ДОДАТОК Г ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ

Об'єкт випробувань

Повне найменування програми: Оцінювання тривалості розробки застосунків.

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 розробки застосунку з використанням нелінійної регресійної моделі в залежності від трудомісткості;
- Оцінювання довірчого інтервалу середнього значення тривалості розробки застосунку;
- Оцінювання інтервалу прогнозування тривалості розробки застосунку;
- Моделювання тривалості розробки застосунку;
- Збереження результатів оцінювання та моделювання в базу даних;
- Перегляд результатів раніше проведених оцінювань тривалості розробки застосунків.

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 I3 2,0GHz.

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 розробки застосунків».

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 Application

```
package EstimationApp;

import EstimationApp.forms.EstimationForm;
import EstimationApp.forms.SavedEstimationsForm;
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 Application {

     static JFrame estimationFrame;
     static EstimationForm estimationForm;

     static JDialog projectsFrame;
     static SavedEstimationsForm savedEstimationsForm;

     static public EntityManager em;

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

     estimationFrame.setContentPane(estimationForm.$$$getRootCompo
nent$$$());

     estimationFrame.setDefaultCloseOperation(WindowConstants.EXIT
_ON_CLOSE);
          estimationFrame.pack();
          estimationFrame.setVisible(true);
          estimationFrame.setLocationRelativeTo(null);
     }

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

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

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

          menuItem = new JMenu("Перегляд виконаних оцінювань та
моделювань");
          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("dur_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
проектів з розробки програмного забезпечення", true);
          projectsFrame.setResizable(true);
```

```
            savedEstimationsForm = new SavedEstimationsForm();
            savedEstimationsForm.setupUIComponents();

        projectsFrame.setContentPane(savedEstimationsForm.$$$getRootC
omponent$$$());

        projectsFrame.setDefaultCloseOperation(WindowConstants.DISPOS
E_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 Entity

```
package EstimationApp.entities;

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

@MappedSuperclass
public abstract class Entity implements Serializable {

    @Id
    public Integer id;

    public Entity() {
    }

    public Entity(Integer id) {
        this.id = id;
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer 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;

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

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

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

Клас Estimation

```
package EstimationApp.entities;

import javax.persistence.*;

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

     public String name;
     public String model_name;
     public Integer effort;
     public Double duration;
     public Double duration_confidence_interval_top;
     public Double duration_confidence_interval_bottom;
     public Double duration_estimation_interval_top;
     public Double duration_estimation_interval_bottom;
     public String generated_duration;
     public String created;
```

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

Клас EstimationModel

```
package EstimationApp.entities;


import EstimationApp.LinearRegressionParameters;

import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;

@javax.persistence.Entity
@Table(name = "estimation_models", catalog = "", schema = "")
@NamedQueries({
	@NamedQuery(name="EstimationModel.findAll", query="SELECT m
FROM EstimationModel m"),
})
public class EstimationModel extends Entity {

	public String name;
	public Double b0;
	public Double b1;
	public Integer n;
	public Double zd_sse;
	public Double ze_avg;
	public Double ze_S;


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

	public LinearRegressionParameters
getLinearRegressionParameters() {
		return new LinearRegressionParameters(b0, b1, zd_sse,
ze_avg, ze_S, n);
	}

	@Override
	public String toString() {
		return name;
	}
}
```

Клас LinearRegressionParameters

```
package EstimationApp;

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

    public LinearRegressionParameters(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;
    }
}
```

Клас EstimationForm

```
package EstimationApp.forms;

import EstimationApp.Application;
import EstimationApp.LinearRegressionParameters;
import EstimationApp.entities.Estimation;
import EstimationApp.entities.EstimationModel;
import Util.MathUtil;
import com.intellij.uiDesigner.core.GridConstraints;
import com.intellij.uiDesigner.core.GridLayoutManager;
import com.intellij.uiDesigner.core.Spacer;
import org.apache.commons.math3.distribution.TDistribution;

import javax.swing.*;
import javax.swing.plaf.FontUIResource;
import javax.swing.text.StyleContext;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.Locale;
import java.util.Random;

public class EstimationForm {
     private JPanel View;
     private JTextField effortTextField;
     private JComboBox modelComboBox;
```

```
    private JTable estimationResults;
    public JButton calculateButton;
    public JTextField simulationCountTextField;
    private JTextField titleTextField;
    private JButton saveToDbButton;
    public JLabel simulationCountLabel;
    private JTextField confidenceLevelTextField;
    private Estimation estimation;

    public void setupUIComponents() {
        final DefaultComboBoxModel modelComboBoxModel = new
DefaultComboBoxModel();

    modelComboBoxModel.addAll(Application.em.createNamedQuery("Es
timationModel.findAll").getResultList());
        modelComboBox.setModel(modelComboBoxModel);

    modelComboBoxModel.setSelectedItem(modelComboBoxModel.getElem
entAt(0));

        final EstimationResults dataModel = new
EstimationResults();
        dataModel.addRow(new String[]{"1. Вкажіть значення
трудомісткості", "(ціле число)"});
        dataModel.addRow(new String[]{"2. Виберіть модель", ""});
        dataModel.addRow(new String[]{"3. Вкажіть довірчу
ймовірність", "(дійсне число)"});
        dataModel.addRow(new String[]{"4. Вкажіть кількість
моделювань", "(ціле число від 1 до 10)"});
        dataModel.addRow(new String[]{"5. Натисніть кнопку \"" +
calculateButton.getText() + "\"", ""});
        estimationResults.setModel(dataModel);

        saveToDbButton.setVisible(false);
        calculateButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                EstimationModel model = (EstimationModel)
modelComboBox.getSelectedItem();
                Double confidenceLevel =
Double.parseDouble(confidenceLevelTextField.getText()) / 100;
                Integer simulationCount =
Integer.parseInt(simulationCountTextField.getText());

                estimation = new Estimation();
                estimation.name = titleTextField.getText();
                estimation.effort =
Integer.parseInt(effortTextField.getText());
                estimation.model_name = model.name;

                LinearRegressionParameters rp =
model.getLinearRegressionParameters();
                int v = rp.n - 2; //Degrees of freedom
```

```
                    double tStudent = (new
TDistribution(v)).inverseCumulativeProbability(1 - (1 -
confidenceLevel) / 2);

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

                    estimation.duration =
MathUtil.round(Math.pow(10, t_z_estimate), 1);
                    estimation.duration_estimation_interval_top =
MathUtil.round(Math.pow(10, t_z_estimate_r2), 1);
                    estimation.duration_estimation_interval_bottom
= MathUtil.round(Math.pow(10, t_z_estimate_l2), 1);
                    estimation.duration_confidence_interval_top =
MathUtil.round(Math.pow(10, t_z_estimate_r1), 1);
                    estimation.duration_confidence_interval_bottom
= MathUtil.round(Math.pow(10, t_z_estimate_l1), 1);

                    dataModel.showEstimation(estimation);


                    Random r = new Random();
                    ArrayList<Double> generated_duration = new
ArrayList<Double>();
                    double delta =
estimation.duration_estimation_interval_top -
estimation.duration_estimation_interval_bottom;
                    for (int i = 0; i < simulationCount && i < 10;
i++) {
                         double r1 =
MathUtil.round(estimation.duration_estimation_interval_bottom +
r.nextDouble() * delta, 1);

                         generated_duration.add(r1);
                         dataModel.addRow(new String[]{(i + 1) +
".", r1 + " місяців"});
                    }
                    estimation.generated_duration =
generated_duration.toString().replace("[", "").replace("]", "");
```

```
                    saveToDbButton.setVisible(true);
               }
          });

          saveToDbButton.addActionListener(new ActionListener() {
               @Override
               public void actionPerformed(ActionEvent e) {
                    estimation.created = new
SimpleDateFormat("yyyy-MM-dd").format(new Date());
                    Application.em.getTransaction().begin();
                    Application.em.persist(estimation);
                    Application.em.getTransaction().commit();
               }
          });
     }

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

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

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

     {
// 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(8, 3, new Insets(10,
10, 10, 10), -1, -1));
          Font ViewFont = this.$$$getFont$$$(null, -1, 16,
View.getFont());
          if (ViewFont != null) View.setFont(ViewFont);
```

```
        final JLabel label1 = new JLabel();
        Font label1Font = this.$$$getFont$$$(null, -1, 16,
label1.getFont());
        if (label1Font != null) label1.setFont(label1Font);
        label1.setHorizontalAlignment(2);
        label1.setHorizontalTextPosition(2);
        label1.setText("Трудомісткість");
        View.add(label1, new GridConstraints(1, 0, 1, 1,
GridConstraints.ANCHOR_WEST, GridConstraints.FILL_NONE,
GridConstraints.SIZEPOLICY_FIXED,
GridConstraints.SIZEPOLICY_WANT_GROW, null, null, null, 0,
false));
        final JLabel label2 = new JLabel();
        Font label2Font = this.$$$getFont$$$(null, -1, 16,
label2.getFont());
        if (label2Font != null) label2.setFont(label2Font);
        label2.setHorizontalAlignment(2);
        label2.setHorizontalTextPosition(2);
        label2.setText("Модель");
        label2.setVerticalAlignment(0);
        label2.setVerticalTextPosition(0);
        View.add(label2, new GridConstraints(2, 0, 1, 1,
GridConstraints.ANCHOR_WEST, GridConstraints.FILL_NONE,
GridConstraints.SIZEPOLICY_FIXED,
GridConstraints.SIZEPOLICY_WANT_GROW, null, null, null, 0,
false));
        effortTextField = new JTextField();
        Font effortTextFieldFont = this.$$$getFont$$$(null, -1,
16, effortTextField.getFont());
        if (effortTextFieldFont != null)
effortTextField.setFont(effortTextFieldFont);
        effortTextField.setText("1000");
        View.add(effortTextField, new GridConstraints(1, 1, 1, 1,
GridConstraints.ANCHOR_CENTER, GridConstraints.FILL_HORIZONTAL,
GridConstraints.SIZEPOLICY_WANT_GROW,
GridConstraints.SIZEPOLICY_FIXED, null, null, null, 0, false));
        modelComboBox = new JComboBox();
        View.add(modelComboBox, new GridConstraints(2, 1, 1, 2,
GridConstraints.ANCHOR_CENTER, GridConstraints.FILL_HORIZONTAL,
GridConstraints.SIZEPOLICY_CAN_GROW,
GridConstraints.SIZEPOLICY_FIXED, null, null, null, 0, false));
        final JScrollPane scrollPane1 = new JScrollPane();
        View.add(scrollPane1, new GridConstraints(7, 0, 1, 3,
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));
        estimationResults = new JTable();
        estimationResults.setAutoCreateRowSorter(false);
        estimationResults.setFillsViewportHeight(true);
```

```
        Font estimationResultsFont = this.$$$getFont$$$(null, -1,
14, estimationResults.getFont());
        if (estimationResultsFont != null)
estimationResults.setFont(estimationResultsFont);
        estimationResults.setShowHorizontalLines(true);
        scrollPane1.setViewportView(estimationResults);
        final Spacer spacer1 = new Spacer();
        View.add(spacer1, new GridConstraints(5, 0, 1, 1,
GridConstraints.ANCHOR_CENTER, GridConstraints.FILL_VERTICAL, 1,
GridConstraints.SIZEPOLICY_WANT_GROW, null, new Dimension(238,
15), null, 0, false));
        final JLabel label3 = new JLabel();
        Font label3Font = this.$$$getFont$$$(null, Font.ITALIC,
14, label3.getFont());
        if (label3Font != null) label3.setFont(label3Font);
        label3.setHorizontalAlignment(2);
        label3.setHorizontalTextPosition(2);
        label3.setText("людино-годин");
        label3.setVerticalAlignment(0);
        label3.setVerticalTextPosition(0);
        View.add(label3, new GridConstraints(1, 2, 1, 1,
GridConstraints.ANCHOR_WEST, GridConstraints.FILL_NONE,
GridConstraints.SIZEPOLICY_WANT_GROW,
GridConstraints.SIZEPOLICY_WANT_GROW, null, null, null, 0,
false));
        calculateButton = new JButton();
        Font calculateButtonFont = this.$$$getFont$$$(null, -1,
14, calculateButton.getFont());
        if (calculateButtonFont != null)
calculateButton.setFont(calculateButtonFont);
        calculateButton.setText("Оцінити тривалість");
        View.add(calculateButton, new GridConstraints(6, 1, 1, 2,
GridConstraints.ANCHOR_EAST, GridConstraints.FILL_NONE,
GridConstraints.SIZEPOLICY_CAN_SHRINK |
GridConstraints.SIZEPOLICY_CAN_GROW,
GridConstraints.SIZEPOLICY_FIXED, null, null, null, 0, false));
        simulationCountLabel = new JLabel();
        Font simulationCountLabelFont = this.$$$getFont$$$(null,
-1, 16, simulationCountLabel.getFont());
        if (simulationCountLabelFont != null)
simulationCountLabel.setFont(simulationCountLabelFont);
        simulationCountLabel.setHorizontalAlignment(2);
        simulationCountLabel.setHorizontalTextPosition(2);
        simulationCountLabel.setText("Кількість моделювань");
        simulationCountLabel.setVerticalAlignment(0);
        simulationCountLabel.setVerticalTextPosition(0);
        View.add(simulationCountLabel, new GridConstraints(3, 0,
1, 1, GridConstraints.ANCHOR_WEST, GridConstraints.FILL_NONE,
GridConstraints.SIZEPOLICY_FIXED,
GridConstraints.SIZEPOLICY_WANT_GROW, null, null, null, 0,
false));
        simulationCountTextField = new JTextField();
```

```
        Font simulationCountTextFieldFont =
this.$$$getFont$$$(null, -1, 16,
simulationCountTextField.getFont());
        if (simulationCountTextFieldFont != null)
simulationCountTextField.setFont(simulationCountTextFieldFont);
        simulationCountTextField.setText("10");
        View.add(simulationCountTextField, new GridConstraints(3,
1, 1, 1, GridConstraints.ANCHOR_CENTER,
GridConstraints.FILL_HORIZONTAL,
GridConstraints.SIZEPOLICY_WANT_GROW,
GridConstraints.SIZEPOLICY_FIXED, null, null, null, 0, false));
        final JLabel label4 = new JLabel();
        Font label4Font = this.$$$getFont$$$(null, -1, 16,
label4.getFont());
        if (label4Font != null) label4.setFont(label4Font);
        label4.setHorizontalAlignment(2);
        label4.setHorizontalTextPosition(2);
        label4.setText("Назва застосунку");
        View.add(label4, new GridConstraints(0, 0, 1, 1,
GridConstraints.ANCHOR_WEST, GridConstraints.FILL_NONE,
GridConstraints.SIZEPOLICY_FIXED,
GridConstraints.SIZEPOLICY_WANT_GROW, null, null, null, 0,
false));
        titleTextField = new JTextField();
        Font titleTextFieldFont = this.$$$getFont$$$(null, -1,
16, titleTextField.getFont());
        if (titleTextFieldFont != null)
titleTextField.setFont(titleTextFieldFont);
        titleTextField.setText("");
        View.add(titleTextField, new GridConstraints(0, 1, 1, 2,
GridConstraints.ANCHOR_CENTER, GridConstraints.FILL_HORIZONTAL,
GridConstraints.SIZEPOLICY_WANT_GROW,
GridConstraints.SIZEPOLICY_FIXED, null, null, null, 0, false));
        saveToDbButton = new JButton();
        Font saveToDbButtonFont = this.$$$getFont$$$(null, -1,
14, saveToDbButton.getFont());
        if (saveToDbButtonFont != null)
saveToDbButton.setFont(saveToDbButtonFont);
        saveToDbButton.setText("Зберегти результати");
        View.add(saveToDbButton, new GridConstraints(6, 0, 1, 1,
GridConstraints.ANCHOR_WEST, GridConstraints.FILL_NONE,
GridConstraints.SIZEPOLICY_CAN_SHRINK |
GridConstraints.SIZEPOLICY_CAN_GROW,
GridConstraints.SIZEPOLICY_FIXED, null, null, null, 0, false));
        final JLabel label5 = new JLabel();
        Font label5Font = this.$$$getFont$$$(null, -1, 16,
label5.getFont());
        if (label5Font != null) label5.setFont(label5Font);
        label5.setHorizontalAlignment(2);
        label5.setHorizontalTextPosition(2);
        label5.setText("Довірча ймовірність");
        label5.setVerticalAlignment(0);
        label5.setVerticalTextPosition(0);
```

```
          View.add(label5, new GridConstraints(4, 0, 1, 1,
GridConstraints.ANCHOR_WEST, GridConstraints.FILL_NONE,
GridConstraints.SIZEPOLICY_FIXED,
GridConstraints.SIZEPOLICY_WANT_GROW, null, null, null, 0,
false));
          confidenceLevelTextField = new JTextField();
          Font confidenceLevelTextFieldFont =
this.$$$getFont$$$(null, -1, 16,
confidenceLevelTextField.getFont());
          if (confidenceLevelTextFieldFont != null)
confidenceLevelTextField.setFont(confidenceLevelTextFieldFont);
          confidenceLevelTextField.setText("95");
          View.add(confidenceLevelTextField, new GridConstraints(4,
1, 1, 1, GridConstraints.ANCHOR_CENTER,
GridConstraints.FILL_HORIZONTAL,
GridConstraints.SIZEPOLICY_WANT_GROW,
GridConstraints.SIZEPOLICY_FIXED, null, null, null, 0, false));
          final JLabel label6 = new JLabel();
          Font label6Font = this.$$$getFont$$$(null, Font.ITALIC,
14, label6.getFont());
          if (label6Font != null) label6.setFont(label6Font);
          label6.setHorizontalAlignment(2);
          label6.setHorizontalTextPosition(2);
          label6.setText("%");
          label6.setVerticalAlignment(0);
          label6.setVerticalTextPosition(0);
          View.add(label6, new GridConstraints(4, 2, 1, 1,
GridConstraints.ANCHOR_WEST, GridConstraints.FILL_NONE,
GridConstraints.SIZEPOLICY_WANT_GROW,
GridConstraints.SIZEPOLICY_WANT_GROW, null, null, null, 0,
false));
          label1.setLabelFor(effortTextField);
          label2.setLabelFor(modelComboBox);
          label3.setLabelFor(effortTextField);
          simulationCountLabel.setLabelFor(modelComboBox);
          label4.setLabelFor(titleTextField);
          label5.setLabelFor(modelComboBox);
          label6.setLabelFor(effortTextField);
     }

     /**
      * @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();
                }
            }
            Font font = new Font(resultName, style >= 0 ? style :
currentFont.getStyle(), size >= 0 ? size : currentFont.getSize());
            boolean isMac = System.getProperty("os.name",
"").toLowerCase(Locale.ENGLISH).startsWith("mac");
            Font fontWithFallback = isMac ? new
Font(font.getFamily(), font.getStyle(), font.getSize()) : new
StyleContext().getFont(font.getFamily(), font.getStyle(),
font.getSize());
            return fontWithFallback instanceof FontUIResource ?
fontWithFallback : new FontUIResource(fontWithFallback);
        }

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

}
```