МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

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

«Допущений до захисту»
Завідувач кафедри
_____
«___» _____ 2022 р.

КВАЛІФІКАЦІЙНАРОБОТА
на здобуття ступеня вищої освіти «магістр»

на тему: «Двохфакторна нелінійна регресійна модель для оцінювання розміру web-застосунків, що створюються мовою Java, та розробка програми для її реалізації»

Виконав: студент групи 6151м

Котигробов С. В.
(підпис, ПІБ)

Керівник роботи:
доцент кафедри ПЗАС, к.т.н., доцент
(посада, науковий ступень, вчене звання)

Макарова Л.М.
(підпис, ПІБ)

Миколаїв – 2022 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

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

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____проф. С.Б.Приходько
(підпис)
«10» жовтня 2022 р.

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

Студенту Котигробову Станіславу Вікторовичу
(Прізвище, ім'я, по батькові)

1. Тема роботи: Двохфакторна нелінійна регресійна модель для оцінювання розміру web-застосунків, що створюються мовою Java, та розробка програми для її реалізації

Керівник роботи: Макарова Лідія Миколаївна, к.т.н., доцент

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

2. Термін подання роботи: 14.11.2022 р.

3. Вихідні дані по роботі:

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

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

5. Перелік презентаційних матеріалів Титульний аркуш, Мета, об`єкт та предмет дослідження, Наукова новизна та практичне значення одержаних результатів, Існуючі моделі для оцінювання розміру web-застосунків, Способи удосконалення моделі для оцінювання розміру web-застосунків, Ескізний проект програми, Технічний проект програми, Робочий проект програми, Висновки, Заключний аркуш

6. Консультанти розділів роботи

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

7. Дата видачі завдання 10.10.2022 р.

КАЛЕНДАРНИЙ ПЛАН

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

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

Студент _____ Котигробов С.В.
(підпис) (ПІБ)

Керівник роботи _____ Макарова Л.М.
(підпис) (ПІБ)

РЕФЕРАТ

Котигробов Станіслав Вікторович

Двохфакторна нелінійна регресійна модель для оцінювання розміру web-застосунків, що створюються мовою Java, та розробка програми для її реалізації

Кваліфікаційна робота на здобуття освітнього рівня магістра зі спеціальності 121 – «Інженерія програмного забезпечення». Національний університет кораблебудування імені адмірала Макарова. Миколаїв, 2022 р.

Обсяг роботи: 107 стор., 10 табл., 10 рис., 32 використаних джерела, 5 додатків.

Актуальність теми роботи: не існує єдиного вимірювання, набору метрик або показників для оцінювання розміру ПЗ, деякі моделі обчислюють розмір, беручи до уваги функціональні, технічні або інші аспекти, тому проблема отримання ефективної моделі для оцінювання розміру web застосунків, що створюються мовою Java, є важливим та актуальним завданням.

Мета та завдання дослідження: підвищення достовірності оцінювання розміру web застосунків, що створюються мовою Java

Завдання дослідження: проаналізувати існуючі моделі для оцінювання розміру web застосунків; обґрунтувати необхідність удосконалення регресійної моделі; розробити програму для оцінювання розміру web застосунків.

Об'єкт дослідження: процес оцінювання розміру web застосунків, що створюються мовою Java.

Предмет дослідження: двохфакторна нелінійна регресійна модель для оцінювання розміру web застосунків, що створюються мовою Java.

Наукова новизна одержаних результатів: удосконалено нелінійну регресійну модель для оцінювання розміру web застосунків, що створюються мовою Java, на основі нормалізуючого перетворення десяткового логарифму і

викидів регресії. Ця двохфакторна нелінійна регресійна модель у порівнянні з однофакторною нелінійною регресійною моделлю та існуючою нелінійною регресійною моделлю, яка була побудована для оцінювання розміру web застосунків, що створюються мовою Java, має більше значення для параметрів якості R^2 та *PRED*(0,25) та менше значення для параметра якості *MMRE*.

Практичне значення одержаних результаті: розроблена програма для оцінювання розміру web застосунків, що створюються мовою Java, дозволила підвищити точність та достовірність відповідних розрахунків.

Апробація результатів роботи. Основні положення і результати досліджень, викладені у магістерській роботі, пройшли апробацію на V Всеукраїнській науково-практичній інтернет-конференції студентів, аспірантів та молодих вчених «Сучасні комп'ютерні системи та мережі в управлінні» (м. Хмельницький, 30 листопада 2022 р.).

Публікації. Основні результати магістерської роботи викладено у 1 науковій праці – матеріалах конференції.

Ключові слова: РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, НОРМАЛІЗУЮЧЕ ПЕРЕТВОРЕННЯ, ОЦІНЮВАННЯ РОЗМІРУ WEB ЗАСТОСУНКІВ, РІВНЯННЯ РЕГРЕСІЇ, JAVA.

ABSTRACT

Stanislav Kotigrobov

A two-factor nonlinear regression model for estimating the size of web applications created in the Java language and developing a program for its implementation

Qualification work for obtaining a master's degree in specialty 121 - "Software engineering". Admiral Makarov National Shipbuilding University. Mykolaiv, 2022

The work contains: 107 стор., 10 табл., 10 рис., 32 використаних джерела, 5 додатків.

Relevance of the topic: there is no single measurement, set of metrics or indicators for estimating the size of software, some models calculate the size taking into account functional, technical or other aspects, so the problem of obtaining an effective model for estimating the size of web applications created in Java language is an important and urgent task.

The purpose and objectives of the research: increasing the reliability of estimating the size of web applications created in the Java language

The task of the research: to analyze the existing models for estimating the size of web applications; justify the necessity of improving the regression model; to develop a program for estimating the size of web applications.

The object of the research: the process of evaluating the size of web applications created in the Java language.

The subject of the research: a two-factor nonlinear regression model for estimating the size of web applications created in the Java language.

Scientific novelty of the obtained results: a non-linear regression model has been improved for estimating the size of web applications created in the Java language, based on the normalizing transformation of the decimal logarithm and regression outliers. This two-variable nonlinear regression model, compared to the one-variable nonlinear regression model and the existing nonlinear regression model that was built to estimate the size of Java web applications, has a higher

value for the quality parameters R2 and PRED(0.25) and a lower value for MMRE quality parameter.

The practical value of the results: the developed program for estimating the size of web applications created in the Java language made it possible to increase the accuracy and reliability of the corresponding calculations.

Approbation of work results: The main provisions and results of the research presented in the master's thesis were approved at the 5th All-Ukrainian Scientific and Practical Internet Conference of Students, Postgraduate Students and Young Scientists "Modern Computer Systems and Networks in Management" (Khmelnytskyi, November 30, 2022) .

Publications: The main results of the master's thesis are presented in 1 scientific work - conference materials.

Keywords: SOFTWARE DEVELOPMENT, NORMALIZED TRANSFORMATION, WEB APPLICATION SIZE ESTIMATION, REGRESSION EQUATION, JAVA.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ, СКОРОЧЕНЬ, ТЕРМІНІВ І ПОЗНАЧЕНЬ

ПЗ	програмне забезпечення
СКБД	система керування базами даних
HTTP	HyperText Transfer Protocol
KLOC	Kilo Lines Of Code
MMRE	Mean of Magnitude of Relative Errors
MSD	Mahalanobis Squared Distance
ORM	Object-Relational Mapping
SQL	Structured Query Language
UI	User Interface
UML	Unified Modeling Language
VIFs	Variance Inflation Factors

ВСТУП

Актуальність теми. З кожним роком збільшується кількість підприємств, які замовляють розробку ПЗ. Основними причинами для цього є: прагнення оптимізувати процеси управління ресурсами, складання звітності та проведення різних операцій. І великі міжнародні корпорації, і порівняно невеликі місцеві компанії, які тільки замислюються про розширення, вбачають необхідність автоматизації як окремих бізнес-процесів, так і роботи взагалі.

На даний момент створення web застосунків у багатьох компаній, що працюють в сфері цифрових і комп'ютерних технологій, вважається одним з перспективних напрямків діяльності. Перенесення бізнес-інструментів і перехід від традиційного ПЗ на web – це напрям, який стрімко розвивається. Той, хто володіє об'єктивною інформацією і грамотно реалізує свої ідеї, розробляючи сучасні бізнес-моделі, може створювати якісний продукт [1].

Web застосунок – це клієнт-серверна програма, де клієнт взаємодіє із сервером через браузер. Логіка web застосунків розподілена між сервером та клієнтом, зберігання даних здійснюється на сервері, обмін інформацією відбувається по мережі інтернет. Клієнти не залежать від конкретної операційної системи, бо web застосунки є кросплатформенними [2].

Розробка web застосунків стає все більш актуальною темою для багатьох компаній, що ведуть свою діяльність в галузі розробки ПЗ, і одночасно доступною для простих користувачів. Однак сьогодні, з розвитком інтернет-технологій, робота в цій сфері значно ускладнилася, одному розробнику вже не під силу впоратися з поставленими завданнями. Поступово ця сфера діяльності переходить в руки як невеликих, так і більших, але професійних компаній. Для них розробка web застосунків стає одним із головних завдань, і всі зусилля спрямовуються на вдосконалення існуючих розробок або створення нових [3].

Можна впевнено казати, що успіх програмних проектів на початкових стадіях залежить від рішень, які приймають менеджери. Для того, щоб рішення були ефективними, вони повинні ґрунтуватися на конкретних фактах, а не на інтуїції [4]. І тут виникає проблема оцінювання розміру майбутнього проекту. Адже розмір ПЗ являє собою один з атрибутів, який використовується в різних моделях для прогнозування вартості, зусиль, ресурсів, необхідних для розробки та впровадження ПЗ [5].

Слід зауважити, що у даній роботі в якості розміру web застосунку розуміється кількість рядків вихідного коду.

До теперішнього часу існує певна проблема в галузі інформаційних технологій, а саме: не існує єдиного вимірювання, набору метрик або показників для оцінювання розміру ПЗ. Існує багато методів для оцінювання розміру програми. Деякі з них обчислюють розмір, беручи до уваги функціональні, технічні або інші аспекти.

Незважаючи на існування великої кількості публікацій, як у світових, так і в українських виданнях, на тему оцінювання розміру ПЗ, розробленого з використанням різних мов програмування, зокрема і для Java [6 – 10], проблема отримання ефективної моделі для оцінювання розміру web застосунків, що створюються мовою Java, в даний час є важливим та актуальним завданням, що вимагає удосконалення існуючих моделей.

Мета дослідження: підвищення достовірності оцінювання розміру web застосунків, що створюються мовою Java.

Для досягнення поставленої мети необхідно вирішити наступні **завдання**:

– проаналізувати існуючі моделі для оцінювання розміру web застосунків;

– обґрунтувати необхідність удосконалення регресійної моделі для оцінювання розміру web застосунків, що створюються мовою Java;

– дослідити різні джерела з відкритим вихідним кодом (open source resources) та визначити web додатки, створені мовою Java;

– отримати (розрахувати) метрики, які необхідні для дослідження;

– виконати передобробку отриманих даних, зокрема, перевірити їх на викиди;

– побудувати лінійну регресійну модель на основі обраного нормалізуючого перетворення для нормалізованих даних;

– перейти до нелінійної регресійної моделі для оцінювання розміру web застосунків, що створюються мовою Java;

– порівняти отриману нелінійну регресійну модель з існуючими моделями за показниками якості;

– розробити програму для оцінювання розміру web застосунків, що створюються мовою Java на основі побудованої нелінійної регресійної моделі.

Об’єкт дослідження: процес оцінювання розміру web застосунків, що створюються мовою Java.

Предмет дослідження: двохфакторна нелінійна регресійна модель для оцінювання розміру web застосунків, що створюються мовою Java.

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

Наукова новизна одержаних результатів: удосконалено нелінійну регресійну модель для оцінювання розміру web застосунків, що створюються мовою Java, на основі нормалізуючого перетворення десяткового логарифму і викидів регресії. Ця двохфакторна нелінійна регресійна модель у порівнянні з однофакторною нелінійною регресійною моделлю та існуючою нелінійною регресійною моделлю, яка була побудована для оцінювання розміру web застосунків, що створюються мовою Java, має більше значення для параметрів якості R^2 та $PRED(0{,}25)$ та менше значення для параметра якості $MMRE$.

Практичне значення одержаних результатів: розроблена програма для оцінювання розміру web застосунків, що створюються мовою Java, дозволила підвищити точність та достовірність відповідних розрахунків.

Особистий внесок здобувача. Магістерська робота є самостійно виконаною працею. Усі результати, викладені у роботі, отримані автором особисто. У праці, яка опублікована у співавторстві з керівником магістерської роботи [11], здобувачеві належить: розробка двохфакторної нелінійної регресійної моделі для оцінювання розміру web застосунків, що створюються мовою Java.

Апробація результатів роботи. Основні положення і результати досліджень, викладені у магістерській роботі, пройшли апробацію на V Всеукраїнській науково-практичній інтернет-конференції студентів, аспірантів та молодих вчених «Сучасні комп'ютерні системи та мережі в управлінні» (м. Хмельницький, 30 листопада 2022 р.).

Публікації. Основні результати магістерської роботи викладено у 1 науковій праці – матеріалах конференції.

1 АНАЛІЗ ІСНУЮЧИХ РЕГРЕСІЙНИХ МОДЕЛЕЙ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ WEB ЗАСТОСУНКІВ, ЩО СТВОРЮЮТЬСЯ МОВОЮ JAVA

1.1 Web застосунки: сутність, особливості та технології створення, переваги використання

Web застосунок – це програма, яка призначена для користувача, головна частина якої знаходиться на вилученому сервері, а призначений для користувача інтерфейс UI користувач бачить у браузері у вигляді веб-сторінок.

Технологія «клієнт-сервер» яскраво виражена у web застосунках, бо вони складаються з двох частин: серверної та клієнтської.

На сервер клієнт відправляє запит, він його одержує та здійснює необхідні обчислення, потім формує веб-сторінку і посилає її клієнту через мережу, використовуючи HTTP. За допомогою клієнтської частини на екрані відображається інтерфейс, відбувається формування запитів до сервера та обробляється відповідь, отримана від нього.

Іноді web застосунок може виступати в якості клієнта інших служб, таких як, бази даних або web застосунку, розташованого на іншому сервері [12].

Згідно зі статистикою [13], у деяких з найбільш популярних застосунків у світі, таких як Facebook та YouTube, більше користувачів, аніж жителів у Китаї. Мета-платформи вважають, що майбутнє знаходиться в Metaverse, а інші, такі як розробник Tiktok, Bytedance, вбачають майбутнє в алгоритмі кураторських новин, музиці та відео. Зазвичай, вони мають як мобільний інтерфейс, так і web версію користувацького інтерфейсу UI.

Для створення web застосунків на серверній частині використовуються різні технології та мови програмування, за допомогою яких можна здійснити

вивід в стандартну консоль. Серед популярних технологій та мов програмування, котрі застосовують для розробки web застосунків на серверній частині можна виокремити: PHP, Java, ASP, ASP.NET, C/C++, Perl, Python, Ruby, Node.Js.

На клієнтській частині у той час застосовують для реалізації UI: XHTML, HTML, CSS, для формування запитів, створення інтерактивного і незалежного від браузера інтерфейсу: Adobe Flash (Flex), ActiveX, Java, Javascript, Silverlight [14].

Підхід до розробки web застосунків Ajax також набув популярності. Він дозволяє оновлювати web застосунок та отримувати дані за запитом, не перезавантажуваючи його повністю, що створює надзвичайну інтерактивність.

Крім того технологія WebSocket також отримала популярність. У разі її використання створюється двонаправлене з'єднання між сервером та клієнтом, при якому клієнт може отримувати дані від сервера без постійних запитів. У результаті контент може динамічно змінюватись безпосередньо на очах.

За своєю суттю запуск web застосунку нічим не відрізняється від завантаження звичайної web сторінки: вводиться посилання в браузер і він перед користувачем, точніше, верхня частина айсберга, якою є інтерфейс користувача UI. В цьому є кілька переваг. Перша – це те, що сам по собі web застосунок абсолютно не залежить від того, яка операційна система та web браузер встановлені на комп'ютері користувача, тобто він, по суті, є крос-платформеними.

Друга – сам факт існування web застосунку повністю змінює спосіб поширення продукту. Тут розробники відходять від традиційних способів поширення програмних продуктів шляхом продажу копій і установки їх на кожен комп'ютер користувачів. Тепер все набагато простіше: єдина версія web застосунку розташована на сервері, а всі користувачі мають доступ до неї, вірніше, до її призначеного для користувача інтерфейсу з будь-якого місця в

світі. При цьому користувачеві навіть не потрібно встановлювати нову версію web застосунку – відразу після своєї появи вона доступна всім, причому багато хто може і не помітити яких-небудь змін, тим більше якщо ці зміни не стосуються зовнішнього вигляду інтерфейсу. У всьому цьому явно видно і позитивний момент для розробників – їм не потрібно піклуватися про сумісність версій своїх програмних продуктів, оскільки всі користувачі одноразово отримують доступ і працюють з самою останньою версією.

Третьою перевагою для користувача є те, що немає потреби встановлювати і налаштовувати програмне забезпечення – все вже встановлено на серверах і налаштоване розробниками. Це великий плюс для користувачів, оскільки більша частина з них не любить мати справу з налаштуваннями і вважає за краще, щоб програмні продукти були повністю готові до використання відразу після їх інсталяції, хоча є й такі, які вважають за краще повністю налаштувати програму під свій смак і потреби. Однак в разі web застосунків користувачі позбавлені навіть процесу інсталяції.

Четвертою перевагою можна назвати те, що для роботи з web застосунком від користувача нічого не потрібно, окрім комп'ютера і встановленого браузера. Інтернет-браузер є в будь-якій операційній системі, і для доступу до необхідного web застосунку досить просто завантажити його веб-адресу у браузер. Використання web застосунків багато в чому знімає обмеження, що накладаються на апаратну частину комп'ютера. Тобто певні системні вимоги до персонального комп’ютера все ж є, але їх рівень автоматично досягнутий комп'ютером, якщо на ньому вже запущені операційна система та браузер.

Наступний позитивний момент web застосунків стосується їх розробників. З огляду на те, що основна частина web застосунків сконцентрована на сервері в одному місці, його налаштування стає набагато простішим, не потрібно утримувати величезні команди фахівців технічної підтримки, що займаються консультаціями користувачів і налаштуванням програм на комп'ютерах користувачів. Це менш затратно в фінансовому плані

і в той же час ефективніше. При цьому користувачеві невидима архітектура web застосунку, в будь-який момент можна додати будь-яку кількість серверів, на яких встановлена основна складова web застосунку, додати обчислювальні потужності і користувач цього навіть не помітить [12].

Таким чином, бачимо, що web застосунки мають велику кількість переваг при відсутності видимих недоліків, найбільшим і очевидним з яких є неможливість використання web застосунків при відсутності доступу до мережі Інтернет.

1.2 Існуючі регресійні моделі для оцінювання розміру web застосунків

Серед основних причин для оцінювання розміру web застосунків можна виділити наступні: оцінка вартості проектів; оцінка продуктивності застосування нових засобів і методів розробки; прогноз потреб у персоналі; поліпшення якості і як наслідки, визначення якості існуючого ПЗ або його експлуатації та прогнозування якості ПЗ при його розробці у майбутньому.

Існуючі моделі для оцінювання розміру web застосунків базуються на моделях, які характерні для оцінювання розміру ПЗ загалом, враховуючи особливості розробки саме web застосунків.

Серед моделей оцінювання розміру ПЗ виділяють декілька категорій: моделі на основі експертних оцінок; аналогові моделі; моделі, які базуються на функціональних точках; параметричні моделі; регресійні моделі [15].

Існують різні методи для оцінювання розміру ПЗ, які використовуються сьогодні. Деякі з них походять від методу аналізу функціональних точок (Function Point Analysis). Інший підхід полягає в тому, щоб провести функціональне вимірювання, щоб виразити функціональність у кількості, що представляє розмір ПЗ. Ряд методів визначення розміру ПЗ включають оцінювання на основі варіантів використання (Use Case). Але найбільш поширеною та найбільш вживаною методологією визначення розміру ПЗ є

підрахунок кількості рядків вихідного коду програми та застосування методів регресійного аналізу.

Серед методик підрахунку кількості рядків можна виділити такі: по числу фізичних рядків (LOC) – визначається як загальне число рядків вихідного коду програми, включаючи коментарі та порожні рядки; або по числу логічних рядків коду (LLOC) – визначається як загальна кількість команд і залежить від використовуваної мови програмування. Якщо мова програмування підтримує розміщення кількох команд в одному рядку, то один фізичний рядок повинен бути врахований як кілька логічних, якщо він містить більше однієї команди. Також є похідні від основних методик, які в залежності від завдання можуть містити додаткову інформацію за такими показниками: число порожніх рядків; число рядків, що містять коментарі; відсоток коментарів (відношення рядків коду до рядків з коментарями, похідна метрика стилістики); середнє число рядків для функцій (класів, файлів); середня кількість рядків, що містять вихідний код для функцій (класів, файлів); середнє число рядків для модулів і т.д.

Багатьма відомими вченими, зокрема М. Холстедом [16], було запропоновано низку метрик, що базуються на аналізі кількості рядків вихідного коду програми та кількості синтаксичних елементів.

Регресійні моделі для оцінювання розміру web застосунків можуть будуватися без урахування мови програмування та фреймворків розробки, зокрема, по даним проектів однієї фірми-розробника ПЗ або з урахуванням як мови програмування, так і відповідного інструментарію розробки, зокрема, фреймворку окремо для різних мов. Також можна брати до розгляду всі типи програм або розбити програми за призначенням. Зрозуміло, що регресійні моделі, які будуються з урахуванням мови та інструментарію розробки, а також типу програм дають більш точні та якісні прогнози.

Серед таких моделей, які побудовані з урахуванням особливостей розробки, можна виділити моделі, побудовані для таких мов програмування, як C++, JavaScript, PHP, VB, Java.

Якщо розглядати регресійні моделі для мови програмування Java, то їх розроблено чи не найбільше у порівнянні з іншими мовами програмування. Окрім моделей, які вже згадувалися у вступі [6 – 10], можна привести ще наступні моделі [17, 18]. Однак, як вже було відмічено вище, найкращі результати дають моделі, які побудовані по конкретним результатам спостережень з урахуванням усіх особливостей розробки саме цих програмних продуктів.

При використанні точної та надійної регресійної моделі для оцінювання розміру ПЗ можна зробити висновок про якість усього процесу розробки ПЗ і при необхідності вжити подальших заходів.

1.3 Способи удосконалення регресійної моделі для оцінювання розміру web застосунків, що створюються мовою Java

Регресійна модель у загальному вигляді може бути представлена наступним виразом:

$$y = \hat{y} + \varepsilon = f(\mathbf{x}) + \varepsilon, \tag{1.1}$$

де y – залежна змінна;

f – функція, яка визначає вид рівняння регресії;

$\mathbf{x}$ – вектор незалежних змінних, $\mathbf{x}=\{x_1, x_2, ..., x_k\}$;

ε – випадкова похибка, розподіл якої підпорядковується нормальному закону розподілу.

У випадку однієї незалежної змінної регресійна модель (1.1) може бути представлена наступним виразом:

$$y = \hat{y} + \varepsilon = f(x) + \varepsilon. \tag{1.2}$$

У випадку двох незалежних змінних регресійна модель (1.1) може бути представлена наступним виразом:

$$y = \hat{y} + \varepsilon = f(x_1, x_2) + \varepsilon . \quad (1.3)$$

Проблема побудови регресійних моделей для оцінювання розміру web застосунків, що створюються мовою Java, полягає в тому, що вихідні емпіричні дані, на основі яких будується регресійна модель та робиться прогноз розміру web застосунку, не підпорядковуються нормальному закону розподілу. Це призводить до необхідності нормалізації вихідних емпіричних даних.

Використання нормалізованих значень дозволить побудувати лінійну регресійну модель, яка у загальному вигляді може бути представлена наступним виразом:

$$Z_y = \hat{Z}_y + \varepsilon = \overline{Z}_y + (\mathbf{Z}_x^+)\hat{\mathbf{b}} + \varepsilon , \quad (1.4)$$

де Z_y – залежна змінна;

ε – випадкова похибка, розподіл якої підпорядковується нормальному закону розподілу;

$\boldsymbol{Z}_x^+$ – матриця центрованих регресорів, $\mathbf{Z}_x^+ = \{Z_1 - \overline{Z}_1, Z_2 - \overline{Z}_2, ...Z_k - \overline{Z}_k\}$;

$\hat{\mathbf{b}}$ – оцінка вектору параметрів рівняння лінійної регресії, $\mathbf{b}=\{b_1, b_2, ..., b_k\}^T$.

У випадку однієї незалежної змінної лінійна регресійна модель (1.4) може бути представлена наступним виразом:

$$Z_y = \hat{Z}_y + \varepsilon = b_0 + b_1 Z_1 + \varepsilon \cdot \quad (1.5)$$

У випадку двох незалежних змінних лінійна регресійна модель (1.4) може бути представлена наступним виразом:

$$Z_y = \hat{Z}_y + \varepsilon = b_0 + b_1 Z_1 + b_2 Z_2 + \varepsilon \cdot \quad (1.6)$$

У загальному вигляді довірчий інтервал лінійного рівняння регресії може бути представлений наступним виразом:

$$\widehat{Y}_\iota \pm t_{\alpha/2,\nu} S_Y \left\{ \frac{1}{N} + (x^+)^T [(\boldsymbol{X}^+)^T \boldsymbol{X}^+]^{-1} (x^+) \right\}^{1/2}, \quad (1.7)$$

де $\widehat{\boldsymbol{Y}}_\iota$ – результат передбачення за лінійним рівнянням регресії;

$\mathbf{X}^+$ – матриця центрованих регресорів, яка містить значення $\boldsymbol{X}^+ = \{\boldsymbol{X_{1i}} - \bar{\boldsymbol{X}}_\mathbf{1}, \boldsymbol{X_{2i}} - \bar{\boldsymbol{X}}_\mathbf{2}, \ldots \boldsymbol{X_{ki}} - \bar{\boldsymbol{X}}_\boldsymbol{k}\}$;

$\boldsymbol{S_Y^2} = \frac{\mathbf{1}}{\boldsymbol{\nu}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \left(\boldsymbol{Y_i} - \widehat{\boldsymbol{Y}}_\boldsymbol{i}\right)^\mathbf{2}$;

ν=*N*-*k*-1;

$(\boldsymbol{X}^+)^{\boldsymbol{T}} \boldsymbol{X}^+$ – *k* x *k* матриця, яка може бути представлена наступним виразом:

$$(\boldsymbol{X}^+)^{\boldsymbol{T}} \boldsymbol{X}^+ = \begin{pmatrix} \boldsymbol{S_{X_1X_1}} & \boldsymbol{S_{X_1X_2}} & \cdots & \boldsymbol{S_{X_1X_k}} \\ \boldsymbol{S_{X_1X_2}} & \boldsymbol{S_{X_2X_2}} & \cdots & \boldsymbol{S_{X_2X_k}} \\ \cdots & \cdots & \cdots & \cdots \\ \boldsymbol{S_{X_1X_k}} & \boldsymbol{S_{X_2X_k}} & \cdots & \boldsymbol{S_{X_kX_k}} \end{pmatrix},$$

де $S_{X_q X_r} = \sum_{i=1}^{N} \left[X_{q_i} - \bar{X}_q\right] \left[X_{r_i} - \bar{X}_r\right]$;

q, *r*=1, 2, ..., *k*.

Відповідно інтервал передбачення лінійного рівняння регресії у загальному вигляді може бути представлений наступним виразом:

$$\widehat{Y}_i \pm t_{\alpha/2,\nu} S_Y \left\{1 + \frac{1}{N} + (x^+)^T [(\boldsymbol{X}^+)^T \boldsymbol{X}^+]^{-1} (x^+)\right\}^{1/2}. \quad (1.8)$$

У випадку однієї незалежної змінної довірчий інтервал лінійного рівняння регресії (1.7) може бути представлений наступним виразом:

$$\hat{Y} \pm t_{\alpha/2,\nu} S_Y \left(\frac{1}{N} + \frac{(x - \bar{x})^2}{\sum_{i=1}^{N} (x_i - \bar{x})^2} \right)^{1/2}. \quad (1.9)$$

Відповідно у випадку однієї незалежної змінної інтервал передбачення лінійного рівняння регресії (1.8) може бути представлений наступним виразом:

$$\hat{Y} \pm t_{\alpha/2,\nu} S_Y \left(1 + \frac{1}{N} + \frac{(x - \bar{x})^2}{\sum_{i=1}^{N} (x_i - \bar{x})^2} \right)^{1/2}. \quad (1.10)$$

Також одним із способів удосконалення регресійної моделі є перевірка даних на викиди. Нормальні дані не повинні містити викидів, тож якщо у наборі даних викиди присутні, це може бути причиною того, що дані не розподіляються за нормальним законом. Якщо викиди будуть знайдені, потрібно видалити їх з набору даних та провести повторну перевірку. Лише після цього можна переходити до наступного кроку аналізу даних, тобто побудови регресійної моделі [20, 21].

Для перевірки даних на викиди будемо використовувати квадрат відстані Махаланобіса MSD, який може бути представлений наступним виразом [21]:

$$d_i^2 = (Z_i - \bar{Z})^T S_N{}^{-1} (Z_i - \bar{Z}), \qquad (1.11)$$

де $\bar{Z}$ – середній вектор вибірки,

S_N – матриця кореляції вибірки, яка може бути представлена наступним виразом:

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

Розрахувавши d_i^2 можна використати або критерій Пірсона χ^2, або критерій Фишера $F_{m,N-m,\alpha}$. При використанні критерію Фишера F потрібно додатково розрахувати тестову статистику T_S, яка може бути представлена наступним виразом:

$$Ts = \frac{(N-m)Nd_i^2}{(N^2-1)m} \qquad (1.12)$$

та мати апроксимований розподіл F з m та $N - m$ ступенями свободи.

Значення d_i^2 або T_S порівнюється з відповідним квантилем розподілу χ^2 або F. Точки, для яких значення d_i^2 або T_S, розраховані за формулами (1.11) та (1.12) відповідно будуть більше, ніж квантиль відповідного розподілу, вважаються викидами і ці значення видаляються з набору даних.

Процедура повторюється до того моменту, поки всі значення d_i^2 або T_S не будуть менше або дорівнюватимуть квантилю відповідного розподілу.

Таким чином, застосовуючи нормалізацію негаусівських вихідних емпіричних даних, побудову лінійної регресійної моделі на основі нормалізованих даних та побудову нелінійної регресійної моделі з використанням зворотного нормалізуючого перетворення, а також усунення

викидів регресії, отримаємо більш якісну нелінійну регресійну модель для оцінювання розміру web застосунків, що створюються мовою Java.

1.4 Перевірка якості регресійної моделі

Для перевірки якості регресійної моделі використаємо коефіцієнт детермінації R^2 [19]:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, \qquad (1.13)$$

де y_i – емпіричне значення y;

$\hat{y}_i$ – розрахункове значення y;

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

R^2 характеризує частку дисперсії, яка обумовлена регресією, в загальній дисперсії показника у. Коефіцієнт детермінації R^2 приймає значення від 0 до 1. Чим ближче значення коефіцієнта до 1, тим тісніше зв'язок результативної ознаки з досліджуваними факторами.

При значенні $R^2 \geq 0{,}5$ можна вважати, що дана модель є прийнятною. Достатньо ефективною та результативною можна вважати модель з показником детермінації $R^2 \geq 0{,}8$. Якщо ж $R^2=1$, то лінія регресії точно відповідає усім спостереженням та вимогам, а модель можна вважати адекватною та достовірною.

Величина коефіцієнта детермінації виступає важливим критерієм оцінки якості лінійних і нелінійних моделей. Чим вагоміша частка пояснюваної варіації, тим менше роль інших факторів, а отже, модель

регресії краще апроксимує вихідні дані і такою регресійної моделлю можна скористатися для прогнозу значень результативного показника.

Для перевірки якості регресійної моделі також використовуються середня величина відносної похибки *MMRE* та рівень прогнозування *PRED*(0,25), які можуть бути представленими наступними виразами:

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} MRE_i\,, \tag{1.14}$$

де n – кількість значень у вибірці;

$MRE_i = \left|\frac{y_i - \hat{y}_i}{y_i}\right|$ – величина відносної похибки;

$$PRED(0{,}25) = \frac{k}{n}, \tag{1.15}$$

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

Ці параметри якості також можна використовувати для порівняння регресійних моделей.

2 УДОСКОНАЛЕННЯ НЕЛІНІЙНОЇ РЕГРЕСІЙНОЇ МОДЕЛІ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ WEB ЗАСТОСУНКІВ, ЩО СТВОРЮЮТЬСЯ МОВОЮ JAVA

2.1 Удосконалення нелінійної регресійної моделі для оцінювання розміру web застосунків із застосуванням нормалізуючих перетворень

Як було зазначено вище у підрозділі 1.3., проблема побудови регресійних моделей для оцінювання розміру web застосунків, що створюються мовою Java, полягає в тому, що вихідні емпіричні дані, на основі яких будується регресійна модель та робиться прогноз розміру web застосунку, не підпорядковуються нормальному закону розподілу. Це призводить до необхідності нормалізації вихідних емпіричних даних.

Для нормалізації негаусівських даних можуть бути використані перетворення на основі десяткового або натурального логарифму, перетворення Бокса-Кокса, перетворення Джонсона та інші. У даній роботі в якості нормалізуючого перетворення буде використовуватись десятковий логарифм:

$$z = \lg(x) \tag{2.1}$$

із зворотним до нього перетворенням:

$$x = 10^{z}. \tag{2.2}$$

З урахуванням (2.1) та (2.2), у випадку однієї незалежної змінної нелінійна регресійна модель (1.1) може бути представлена наступним виразом:

$$Y=10^{\varepsilon+b_0} X^{b_1} . \quad (2.3)$$

З урахуванням (2.1) та (2.2), у випадку двох незалежних змінних регресійна модель (1.1) може бути представлена наступним виразом:

$$Y=10^{\varepsilon+b_0} X_1^{b_1} X_2^{b_2} . \quad (2.4)$$

Тобто отриманий вираз (2.4) є двохфакторною нелінійною регресійною моделлю, яку можна застосувати для оцінювання розміру web застосунків, що створюються мовою Java.

У випадку двох та більше незалежних змінних перед побудовою регресійної моделі потрібно перевірити ці змінні на мультиколінеарність, тобто перевірити, чи є лінійна залежність між цими змінними. Якщо така лінійна залежність існує, ці змінні не варто використовувати для побудови множинної лінійної регресії.

Наявність мультиколінеарності між метриками, які планується використовувати в якості незалежних змінних при побудові двохфакторної нелінійної регресійної моделі (2.4), можна перевірити за допомогою коефіцієнтів впливу дисперсії *VIFs*. У разі лінійної моделі множинної регресії з k факторами X_i, $i = 1,\ldots, k$, *VIFs* – це діагональні елементи оберненої кореляційної матриці kxk [22]. Якщо значення *VIFs*>10, це свідчить про те, що дані мають проблеми з мультиколінеарністю. Якщо якщо значення *VIFs* знаходяться у діапазоні від 1 до 5, це свідчить про те, то мультиколінеарності немає.

Для перевірки на відповідність нормальному закону розподілу випадкової похибки ε при побудові лінійної регресійної моделі використаємо критерій згоди χ^2 Пірсона [23, 24].

Розрахункова формула критерію наступна:

$$\chi^2 = \sum_{i=1}^{m} \frac{(n_i - np_i)^2}{np_i}, \qquad (2.5)$$

де m – кількість підінтервалів, яка залежить на від обсягу вибірки n та може бути розрахована за формулою Старджеса $m = 3{,}31 \cdot \lg n + 1$ або формулою Брукса і Карузера $m = 5 \cdot \lg n$;

n_i – кількість значень випадкової величини, які потрапили в i-й підінтервал;

p_i – ймовірність потрапляння випадкової величини в i-й підінтервал відповідно з гіпотетичним законом розподілу випадкової величини.

В залежності від рівня значимості α та кількості ступенів вільності ν за таблицею верхніх 100α% точок розподілу χ^2 знаходять значення $\chi^2_{кр}$. У разі, коли $\chi^2 \leq \chi^2_{кр}$, з ймовірністю 1-α можна стверджувати, що закон розподілу випадкової похибки ε відповідає нормальному закону розподілу.

2.2 Побудова двохфакторної нелінійної регресійної моделі для оцінювання розміру web застосунків, що створюються мовою Java

Для оцінювання розміру web застосунків, що створюються мовою Java, в рамках даної роботи на інтернет-ресурсі GitHub [25] було знайдено 30 web застосунків, створених мовою Java, з відкритим вихідним кодом (open source software). Кожен web застосунок було проаналізовано та побудовано діаграму класів. Було прийнято рішення будувати двохфакторну нелінійну регресійну модель на основі десяткового логарифму за такими метриками:

KLOC – кількість рядків коду програми, у тисячах, Y;

NC – загальна кількість класів, X_1;

ANM – середня кількість методів, X_2.

Згідно із [26], із метрик отриманих на основі діаграм класів можна побудувати нелінійну регресійну модель для багатовимірних негаусівських даних для оцінювання розміру ПЗ, у тому числі, і розміру web застосунків, що створюються мовою Java, та отримати непогані результати у порівнянні з іншими регресійними моделями. Вказані метрики наведені у табл. 2.1.

Оскільки ми будуємо двохфакторну нелінійну регресійну модель, перед побудовою регресійної моделі перевіримо змінні X_1 та X_2 на мультиколінеарність за допомогою коефіцієнтів впливу дисперсії *VIFs*. Обернена кореляційна матриця 2x2 наведена нижче:

$$\begin{pmatrix} \mathbf{1,17} & \mathbf{-0,45} \\ \mathbf{-0,45} & \mathbf{1,17} \end{pmatrix}.$$

Оскільки потрібні значення дорівнюють 1,17 та знаходяться у діапазоні від 1 до 5, мультиколінеарності між змінними X_1 та X_2 немає, тож можна використовувати її в якості незалежних змінних для побудови двохфакторної нелінійної регресійної моделі.

Також у табл. 2.1 наведені нормалізовані значення вказаних метрик. Нормалізація виконувалася за допомогою десяткового логарифму за формулою (2.1).

Для нормалізованих даних були розраховані квадрат відстані Махаланобіса d_i^2 за формулою (1.11) та тестова статистика критерію Фішера T_S за формулою (1.12), значення яких також наведені у табл. 2.1.

Проведена перевірка на виявлення викидів за допомогою d_i^2 та T_S показала відсутність викидів: для всіх рядків даних значення d_i^2 є меншим ніж величина квантіля χ^2, яка дорівнює 7,81 для рівня значимості 0,05 та для всіх рядків даних значення T_S є меншим ніж величина квантіля $F_{m,N-m,\alpha}$, яка дорівнює 2,96 для рівня значимості 0,05.

Таблиця 2.1 – Метрики web застосунків, що створюються мовою Java

№ проекту	Y, KLOC	X_1, NC	X_2, ANM	Z_Y	Z_{X1}	Z_{X2}	d_i^2	T_S
1	1,32	18	2,61	0,1206	1,2553	0,4166	0,13	0,04
2	1,02	17	2,41	0,0086	1,2304	0,3820	0,18	0,05
3	0,739	12	2,08	-0,1314	1,0792	0,3181	1,25	0,37
4	6,878	38	4,11	0,8375	1,5798	0,6138	3,34	1,00
5	0,616	19	3,26	-0,2104	1,2788	0,5132	5,51	1,66
6	9,444	49	3,24	0,9752	1,6902	0,5105	5,14	1,54
7	2,229	28	6,18	0,3481	1,4472	0,7910	4,80	1,44
8	0,261	5	2,6	-0,5834	0,6990	0,4150	6,29	1,89
9	1,177	17	4,29	0,0708	1,2304	0,6325	1,78	0,53
10	1,109	22	2,73	0,0449	1,3424	0,4362	0,93	0,28
11	0,543	6	3,83	-0,2652	0,7782	0,5832	7,20	2,16
12	1,267	15	2,6	0,1028	1,1761	0,4150	0,89	0,27
13	1,21	15	2,47	0,0828	1,1761	0,3927	0,84	0,25
14	2,034	31	1,68	0,3084	1,4914	0,2253	2,27	0,68
15	2,263	17	3,82	0,3547	1,2304	0,5821	2,38	0,71
16	7,915	30	5,47	0,8985	1,4771	0,7380	6,63	1,99
17	1,05	15	2,13	0,0212	1,1761	0,3284	0,86	0,26
18	6,2	57	4,26	0,7924	1,7559	0,6294	3,44	1,03
19	1,457	20	5,05	0,1635	1,3010	0,7033	2,82	0,85
20	0,516	11	2,64	-0,2874	1,0414	0,4216	1,57	0,47
21	0,95	26	3,08	-0,0223	1,4150	0,4886	4,69	1,41
22	0,423	11	1,45	-0,3737	1,0414	0,1614	2,62	0,79
23	8,47	65	5,49	0,9279	1,8129	0,7396	5,15	1,55
24	0,507	18	0,94	-0,2950	1,2553	-0,0269	6,21	1,87
25	3,999	37	2,41	0,6020	1,5682	0,3820	2,07	0,62
26	1,395	28	1,86	0,1446	1,4472	0,2695	1,62	0,49
27	0,586	13	2,23	-0,2321	1,1139	0,3483	1,01	0,30
28	1,327	26	1	0,1229	1,4150	0,0000	6,36	1,91
29	2,93	40	2,73	0,4669	1,6021	0,4362	1,53	0,46
30	1,488	25	2,24	0,1726	1,3979	0,3502	0,50	0,15

Тепер будуємо двохфакторну лінійну регресійну модель для нормалізованих даних згідно з (1.6) та за допомогою методу найменших квадратів знаходимо коефіцієнти рівняння лінійної регресії: b_0=-1,7778, b_1=1,2757, b_2=0,6140. Двохфакторна лінійна регресійна модель для нормалізованих даних має наступний вигляд:

$$Z_Y = -1{,}7778 + 1{,}2757 Z_1 + 0{,}6140 Z_2 + \varepsilon .$$

Тепер перевіримо випадкову похибку ε на відповідність нормальному закону розподілу за допомогою критерію згоди χ^2 Пірсона за формулою (2.5): χ^2=0,28.

При рівні значимості α=0,05 та кількості ступенів вільності ν=2 за таблицею верхніх 100α% точок розподілу χ^2 знаходимо значення $\chi^2_{кр}$: $\chi^2_{кр}$=5,99. Оскільки $\chi^2<\chi^2_{кр}$, з ймовірністю 0,95 можна стверджувати, що закон розподілу випадкової похибки ε відповідає нормальному закону, тобто лінійна регресійна модель була побудована обгрунтовано.

Тепер для лінійного рівняння регресії будуємо довірчий інтервал за формулою (1.7) та інтервал передбачення за формулою (1.8).

Далі можемо будувати двохфакторну нелінійну регресійну модель (2.4) на основі побудованої двохфакторної лінійної регресійної моделі (1.6) та зворотного перетворення (2.2):

$$Y = 10^{\varepsilon - 1{,}7778} X_1^{1{,}2757} X_2^{0{,}6140} .$$

Тепер для нелінійного рівняння регресії будуємо довірчий інтервал та інтервал передбачення, використовуючи аналогічні побудовані інтервали для лінійного рівняння регресії.

Для точки даних 5 зі значеннями метрик Y=0,616, X_1=19, X_2=3,26 значення нижньої границі інтервалу передбачення дорівнює 0,6299, а значення верхньої границі інтервалу передбачення дорівнює 3,4512, тобто значення метрики Y виходить за границі інтервалу передбачення, а, отже, ця точка даних є викидом і її потрібно видалити з набору даних та знову перевірити отриманий набор даних на викиди за допомогою квадрату відстані Махаланобіса, побудувати для нього двохфакторну лінійну та нелінійну регресійні моделі.

Для нового набору даних з 29 точок перевірка на виявлення викидів за допомогою d_i^2 та T_S показала відсутність викидів. Побудована двохфакторна лінійна регресійна модель для нормалізованих даних згідно з (1.6), за допомогою методу найменших квадратів знайдено коефіцієнти рівняння лінійної регресії: b_0=-1,7599, b_1=1,2622, b_2=0,6436, двохфакторна лінійна регресійна модель для нормалізованих даних з 29 точок має наступний вигляд:

$$Z_Y = -1{,}7599 + 1{,}2622 Z_1 + 0{,}6436 Z_2 + \varepsilon .$$

Перевірка випадкової похибки ε на відповідність нормальному закону розподілу за допомогою критерію згоди χ^2 Пірсона за формулою (2.5) показала, що при рівні значимості α=0,05 та кількості ступенів вільності ν=2 закон розподілу випадкової похибки ε відповідає нормальному закону: χ^2=0,45 при $\chi^2_{кр}$=5,99.

Для лінійного рівняння регресії побудували довірчий інтервал за формулою (1.7) та інтервал передбачення за формулою (1.8).

Побудована двохфакторна нелінійна регресійна модель (2.4) на основі лінійної регресійної моделі (1.6) та зворотного перетворення (2.2) має наступний вигляд:

$$Y = 10^{\varepsilon - 1,7599} X_1^{1,2622} X_2^{0,6436}.$$

Для нелінійного рівняння регресії побудували довірчий інтервал та інтервал передбачення, використовуючи аналогічні побудовані інтервали для відповідного лінійного рівняння регресії.

У цьому випадку для точки даних 21 зі значеннями метрик Y=0,95, X_1=26, X_2=3,08 значення нижньої границі інтервалу передбачення дорівнює 1,0003, а значення верхньої границі інтервалу передбачення дорівнює 4,7967, тобто значення метрики Y знов виходить за границі інтервалу передбачення, а, отже, і ця точка даних є викидом і її потрібно видалити з набору даних та знову провести обробку даних.

Для отриманого набору даних з 28 точок перевірка на виявлення викидів за допомогою d_i^2 та T_S також показала відсутність викидів. Побудована двохфакторна лінійна регресійна модель для нормалізованих даних згідно з (1.6), за допомогою методу найменших квадратів знайдено коефіцієнти рівняння лінійної регресії: b_0=-1,7741, b_1=1,2791, b_2=0,6551, двохфакторна лінійна регресійна модель для нормалізованих даних з 28 точок має наступний вигляд:

$$Z_Y = -1,7741 + 1,2791 Z_1 + 0,6551 Z_2 + \varepsilon.$$

Перевірка випадкової похибки ε на відповідність нормальному закону розподілу за допомогою критерію згоди χ^2 Пірсона за формулою (2.5) показала, що при рівні значимості α=0,05 та кількості ступенів вільності ν=2 закон розподілу випадкової похибки ε відповідає нормальному закону: χ^2=3,03 при $\chi^2_{кр}$=5,99.

Для лінійного рівняння регресії побудували довірчий інтервал за формулою (1.7) та інтервал передбачення за формулою (1.8).

Побудована двохфакторна нелінійна регресійна модель (2.4) на основі лінійної регресійної моделі (1.6) та зворотного перетворення (2.2) для 28 точок даних має наступний вигляд:

$$Y = 10^{\varepsilon - 1,7741} X_1^{1,2791} X_2^{0,6551}. \quad (2.6)$$

Для нелінійного рівняння регресії побудували довірчий інтервал та інтервал передбачення, використовуючи аналогічні побудовані інтервали для відповідного лінійного рівняння регресії.

У цьому випадку усі точки знаходяться в межах інтервалу передбачення, тобто викидів не виявлено, отже формула (2.6) є остаточним виглядом двохфакторної нелінійної регресійної моделі для оцінювання розміру web застосунків, що створюються мовою Java.

Розрахуємо показники якості регресійної моделі для моделі (2.6): R^2 згідно з (1.13), *MMRE* згідно з (1.14) та *PRED*(0,25) згідно з (1.15). R^2=0,8463, *MMRE*=0,2733, *PRED*(0,25)=0,5333.

Порівняємо отримані показники якості з аналогічними показниками інших моделей, для чого використаємо такі моделі для порівняння: побудовану двохфакторну лінійну регресійну модель без нормалізації даних, побудовану однофакторну нелінійну регресійну модель та існуючу однофакторну нелінійну регресійну модель [9]. Незважаючи на те, що для оцінювання розміру web застосунків, що створюються мовою Java, існує велика кількість моделей, зокрема, ті, які згадувалися у вступі [6 – 10] та у підрозділі 1.2 [17, 18], підібрати існуючу модель для порівняння було важко, оскільки більшість моделей використовують в якості незалежних змінних 3 або 4 параметри. Тому в якості моделі для порівняння була обрана існуюча однофакторна нелінійна регресійна модель на основі логарифмічного перетворення з десятковим логарифмом, представлена в [9].

Будуємо двохфакторну лінійну регресійну модель без нормалізації даних згідно з (1.6) та за допомогою методу найменших квадратів знаходимо коефіцієнти рівняння лінійної регресії: b_0=-2,3801, b_1=0,1371, b_2=0,4673. Двохфакторна лінійна регресійна модель має наступний вигляд:

$$Z_Y = -2{,}3801 + 0{,}1371Z_1 + 0{,}4673Z_2 + \varepsilon .$$

Перевірка випадкової похибки ε на відповідність нормальному закону розподілу за допомогою критерію згоди χ^2 Пірсона за формулою (2.5) показала, що при рівні значимості α=0,05 та кількості ступенів вільності ν=2 закон розподілу випадкової похибки ε не відповідає нормальному закону: χ^2=9,85 при $\chi^2_{кр}$=5,99. Тобто цю двохфакторну лінійну регресійну модель без нормалізації даних неможливо застосовувати для оцінювання розміру web застосунків, що створюються мовою Java.

Будуємо однофакторну нелінійну регресійну модель згідно з (2.3) та за допомогою методу найменших квадратів знаходимо коефіцієнти рівняння лінійної регресії: b_0=-1,6631, b_1=1,3937. Однофакторна нелінійна регресійна модель для нормалізованих даних має наступний вигляд:

$$Y = 10^{\varepsilon - 1{,}6631} X_1^{1{,}3937} . \qquad (2.7)$$

Перевірка випадкової похибки ε на відповідність нормальному закону розподілу за допомогою критерію згоди χ^2 Пірсона за формулою (2.5) показала, що при рівні значимості α=0,05 та кількості ступенів вільності ν=2 закон розподілу випадкової похибки ε відповідає нормальному закону: χ^2=1,19 при $\chi^2_{кр}$=5,99. Тобто цю однофакторну нелінійну регресійну модель можливо застосовувати для оцінювання розміру web застосунків, що створюються мовою Java.

Розрахуємо показники якості регресійної моделі для моделі (2.8): R^2 згідно з (1.13), *MMRE* згідно з (1.14) та *PRED*(0,25) згідно з (1.15). R^2=0,6496, *MMRE*=0,3740, *PRED*(0,25)=0,4333.

Будуємо однофакторну нелінійну регресійну модель згідно з [9], яка має наступний вигляд:

$$Y = 10^{\varepsilon - 1,4690} X_1^{1,2266}. \quad (2.8)$$

Перевірка випадкової похибки ε на відповідність нормальному закону розподілу за допомогою критерію згоди χ^2 Пірсона за формулою (2.5) показала, що при рівні значимості α=0,05 та кількості ступенів вільності ν=2 закон розподілу випадкової похибки ε відповідає нормальному закону: χ^2=0,94 при $\chi^2_{кр}$=5,99. Тобто цю однофакторну нелінійну регресійну модель можливо застосовувати для оцінювання розміру web застосунків, що створюються мовою Java.

Розрахуємо показники якості регресійної моделі для моделі (2.8): R^2 згідно з (1.13), *MMRE* згідно з (1.14) та *PRED*(0,25) згідно з (1.15). R^2=0,5339, *MMRE*=0,3587, *PRED*(0,25)=0,4667.

Для зручності аналізу, представимо показники якості регресійних моделей, побудованих за допомогою виразів (2.6), (2.7) та (2.8), у таблиці 2.2.

Таблиця 2.2 – Показники якості регресійних моделей

Параметр	Двохфакторна нелінійна регресійна модель (2.6)	Однофакторна нелінійна регресійна модель (2.7)	Однофакторна нелінійна регресійна модель (2.8)
R^2	0,8463	0,6496	0,5339
MMRE	0,2733	0,3740	0,3587
PRED(0,25)	0,5333	0,4333	0,4667

Із аналізу даних, представлених у табл. 2.2, можна зробити наступні висновки:

1. Двохфакторна нелінійна регресійна модель, яка побудована за допомогою виразу (2.6) за всіма показниками якості краща, ніж однофакторна нелінійна регресійна модель, яка побудована за допомогою виразу (2.7): R^2 на 30,28%, *MMRE* на 26,93%, *PRED*(0,25) на 23,08%.

2. Двохфакторна нелінійна регресійна модель, яка побудована за допомогою виразу (2.6) за всіма показниками якості краща, ніж однофакторна нелінійна регресійна модель, яка побудована за допомогою виразу (2.8): R^2 на 58,51%, *MMRE* на 23,81%, *PRED*(0,25) на 14,27%.

3. Включення в модель другого фактору суттєво покращує якість регресійної моделі для оцінювання розміру web застосунків, що створюються мовою Java.

4. Незважаючи на кращі показники якості двохфакторної нелінійної регресійної моделі у порівнянні з обома однофакторними нелінійними регресійними моделями, слід відмітити, що значення показників якості *MMRE* та *PRED*(0,25) для двохфакторної нелінійної регресійної моделі не відповідають необхідним допустимим значенням, а саме: не більше 0,25 для *MMRE* та не менше 0,75 для *PRED*(0,25).

Усунути цей недолік можливо, по-перше, за рахунок збільшення кількості web застосунків, створених мовою Java, або точок даних для побудови двохфакторної нелінійної регресійної моделі, тим більше, що після прибирання викидів із первинного набору даних, для остаточного аналізу та побудови моделі залишилося всього 28 точок, тобто отримали вже малу вибірку для аналізу. По-друге, треба спробувати інші нормалізуючі перетворення для побудови двохфакторної нелінійної регресійної моделі.

3 РОЗРОБКА ПРОЄКТУ ПЗ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ WEB-ЗАСТОСУНКІВ, ЩО СТВОРЮЮТЬСЯ МОВОЮ JAVA

Потрібно створити ПЗ, яке може приймати метрики web-застосунків створених мовою програмування Java, а потім обробляти ці дані за допомогою побудови двохфакторної нелінійної регресійної моделі для оцінки їх розміру. і відобразити результат користувачеві.

3.1 Ескізний проект програмного забезпечення

Під час розробки програмного проекту за допомогою структурованого підходу розробляються ескізи, технічні та робочі елементи.

Коли розробляється ескіз, створюються діаграми варіантів використання які описують модель програмного забезпечення. Щоб формалізувати представлення сценаріїв використання, будуються діаграми діяльності для основних варіантів використання, належним чином проектується дизайн інтерфейсу користувача та розробляються концептуальна модель відповідних предметних областей.

Ескізний проект призначений для опису ідей або функцій на абстрактному рівні, для отримання загального уявлення про рішення та особливості, які є основою для подальшого розвитку.

За ескізним проектом можна вирішити, чи розробляти технічний проект та документацію. Тобто на етапі окреслення ескізного проекту частини розглядаються як єдине ціле. І деталі рішення повинні бути зроблені настільки, щоб різні варіанти можна було порівняти один з одним.

Після аналізу вимог до ПЗ в технічних завданнях розробляється проект дизайну ПЗ, побудова діаграм, концептуальної моделі даних, моделі переходу між станами та дизайну інтерфейсу.

3.1.1 Вибір мови моделювання

UML – Unified Modeling Language – Уніфікована мова моделювання стала стандартом для створення програмного забезпечення. За допомогою діаграми UML легко створювати систему на абстрактному рівні яка допомагає у обговорюванні вимог до програмного забезпечення, що в свою чергу допомагає працювати в команді, вести переговори та вирішувати проблеми між різними групами розробників.

Прийнята версія UML 1.1 у 1997 році, містить структурні та поведінкові моделі [27].

Структурна модель описує:

– діаграму компонентів – ієрархія системних компонентів;

– діаграму класів – модель структури класів та зв’язків;

– діаграму розгортання – архітектура системи.

Поведінкова модель складається з діаграм:

– варіантів використання (взаємодії користувача з системою) – функціональні вимоги системи;

– діаграми станів – поведінка об’єктів системи;

– діаграми активності – моделювання поведінки системи.

Тому для візуалізації процесу створення програмного забезпечення та відстеження послідовності перетворення від початкової моделі до готової програмної системи вибір пав на UML [28].

UML допомагає у вирішенні:

– надання простої для розуміння мови моделювання, яка використовується для документування та розробки складних системних

моделей для різних цілей;

– забезпечення більш точного представлення системних моделей при аналізі та проектуванні;

– представлення моделей які не залежать від особливостей її реалізації у мовах програмування.

UML не повинна залежати від мов програмування, але повинна включати в себе найновіші та найкращі результати практики та можливість вдосконалюватися з часом.

3.1.2 Побудова діаграми варіантів використання

Діаграми варіантів використання дозволяють нам зрозуміти, які сутності є в системі, хто є головними учасниками програми та які їхні основні дозволи на основні функції системи.

Було визначено сценарії та побудовано варіанти використання, а також розроблено специфікацію програмного забезпечення для оцінки розміру web-застосунку, реалізованого на Java, і стану програми під час виконання [29].

Програмне забезпечення використовується лише одним користувачем, який буде використовувати функції ПЗ. Основні варіанти використання програмного забезпечення наведено в таблиці 3.1.

Таблиця 3.1 – Основні варіанти використання програмного забезпечення

Найменування	Формулювання
1	2
Занесення даних про класи до програми	Занесення даних про класи, за якими буде виконуватись оцінка розміру web-застосунків, що створюються мовою Java
Обчислення вихідних даних	Обчислення та корекція занесених даних

Продовження табл. 3.1

1	2
Перевірка даних на нормальність	Виконує перевірку даних на відповідність гаусівському розподілу
Нормалізація даних	Нормалізує дані за допомогою логарифмічного перетворення
Оцінювання нормалізованих даних на нормальність	Виконує перевірку даних на відповідність гаусівському розподілу
Побудова довірчого інтервалу та інтервалу передбачення нелінійної регресії	Будує нелінійне рівняння регресії для оцінювання розміру веб-застосунків

Основні варіанти використання програмного забезпечення для оцінювання розміру web-застосунків, реалізованих мовою Java, наведено на рис. 3.1.

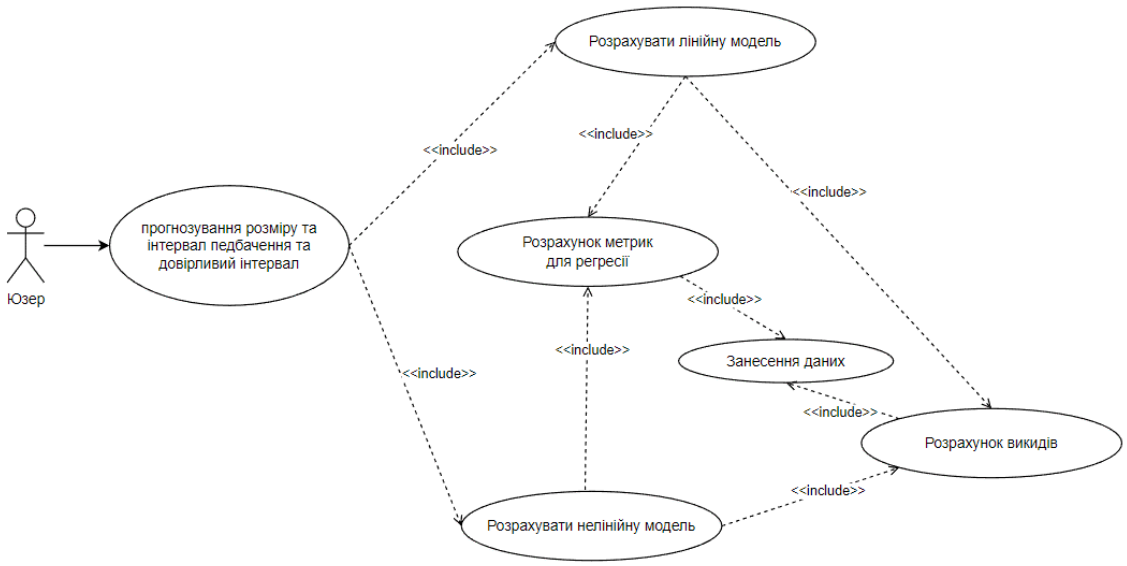


Рисунок 3.1 – Основні варіанти використання програмного забезпечення

З функціональними можливостями програмного забезпечення можна ознайомитись на діаграмі варіантів використання.

3.1.3 Розробка специфікації варіантів використання

Більш детально основні варіанти використання програмного забезпечення представлені у табл. 3.2 - 3.5.

Таблиця 3.2 – Специфікація варіанту використання «Внесення даних»

Призначення	Внесення даних до системи
Учасник	Система, користувач
Передумова	Правильний формат даних
Процедура	– користувач обирає файл; – система читає та валідує дані; – користувач отримує відповідь від системи.
Пост умова	користувачу доступно більше функцій після успішного завантаження та валідації даних.
Альтернативний потік	– користувач обирає файл; – система читає та валідує дані; – користувач отримує від системи відповідь з помилкою

Таблиця 3.3 – Специфікація варіанту використання «Нормалізація даних»

Призначення	Нормалізація даних
Учасник	Система, користувач
Передумова	Наявність правильних даних
Процедура	– система застосовує нормалізуюче перетворення до даних і отримує розподіл Гауса.
Пост умова	збереження параметрів нормалізації

Таблиця 3.4 – Специфікація варіанту використання «Побудувати довірчий інтервал»

Призначення	Побудова довірчого інтервалу для оцінки розміру ПЗ
Учасник	Система, користувач
Передумова	Наявність даних після нормалізації
Процедура	– обчислення меж довірчого інтервалу для оцінки ПЗ;
Пост умова	збереження параметрів

Таблиця 3.5 – Специфікація варіанту використання «Прогнозування розміру»

Призначення	Побудова інтервалу прогнозування для оцінки розміру ПЗ
Учасник	Система, користувач
Передумова	Наявність даних після побудови довірчого інтервалу
Процедура	– обчислення меж інтервалу прогнозування; – збереження параметрів.
Пост умова	вивід результату

В додатку А представлено розроблене ТЗ на основі наведеного переліку основних варіантів використання програмного забезпечення.

3.1.4 Розробка діаграми діяльності

Діаграми діяльності використовуються для опису поведінки об’єктів. Приклад діаграми діяльності продемонстровано на рис. 3.2 – 3.3.

Мабуть один із найголовніших варіантів – це занесення даних до системи, від цього залежить як буде працювати вся програма. Якщо дані були з помилкою – треба зразу повідомити користувача, інакше програма може припинити працювати в середині обчислень.

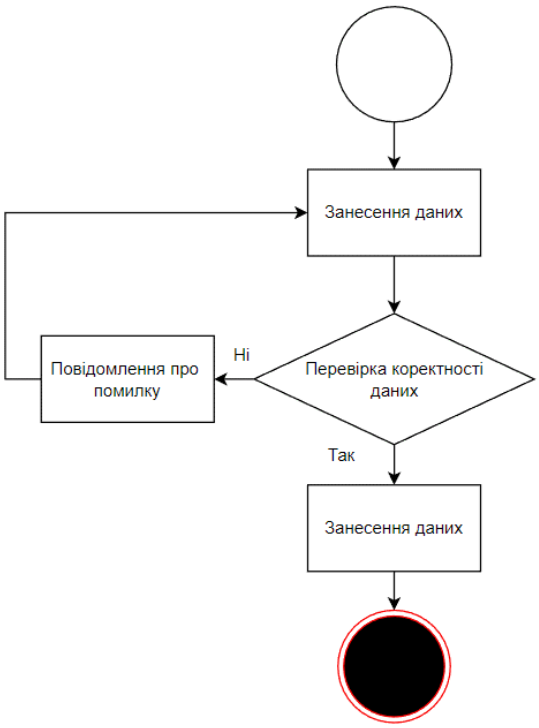


Рисунок 3.2 – Діаграма діяльності «Внесення даних»

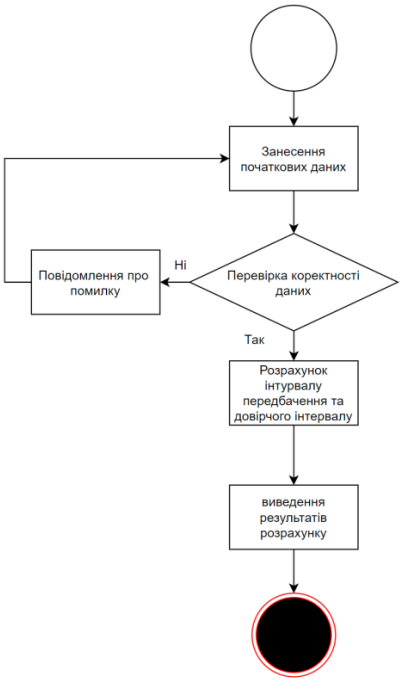


Рисунок 3.3 – Діаграма діяльності «Прогнозування розміру»

Для більш детального розуміння основних способів використання програмного забезпечення було створено діаграму послідовності.

3.1.5 Розробка прототипу користувацького інтерфейсу

Інтерфейс є основною частиною програмного коду, який взаємодіє з користувачем і має чітку та просту для вивчення структуру, яка в ідеалі дозволяє користувачеві зрозуміти як досягти своїх цілей.

Але не всі ці фактори можна взяти до уваги для професійних програм, оскільки професійні програми мають багато різних вимог, і важко розробити інтерфейс, який користувачі можуть одразу зрозуміти, оскільки для цього потрібна система, а не простота використання.

У цьому розділі описано основний інтерфейс користувача для обчислення даних та аналізу.

Для введення початкових даних розрахунку було розроблено ескіз екранної форми, що представлено на рисунку 3.4.

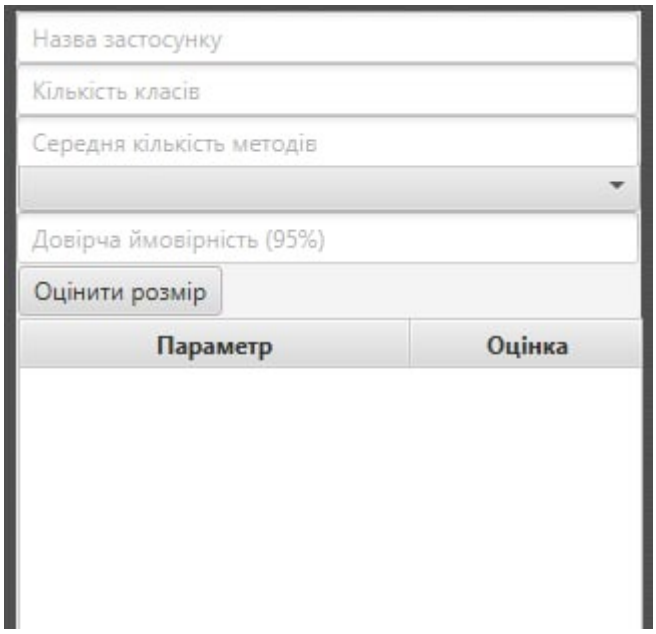


Рисунок 3.4 – Форма введення початкових даних

Користувацький інтерфейс, розроблений за таким прототипом, буде зрозумілим для користувача програмного забезпечення.

3.2 Технічний проект програмного забезпечення

3.2.1 Побудова динамічної моделі програмного забезпечення

Зміни в програмі або конкретні функції показують за допомогою динамічної моделі. Динамічні моделі описують діаграми послідовності, які є діаграмами UML, які показують взаємодію між елементами системи в часі. Діаграми послідовності моделюють у вигляді життєвого циклу процесів та об’єктів.

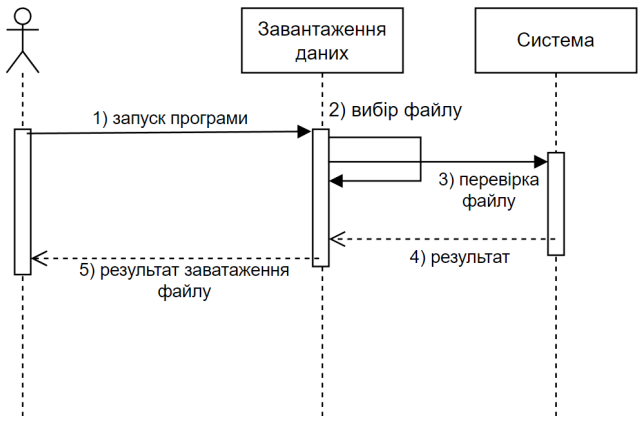


Рисунок 3.5 – Діаграма послідовності «Завантаження даних»

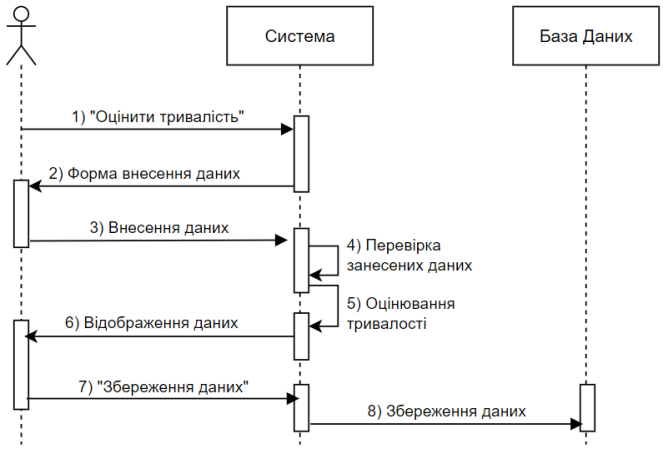


Рисунок 3.6 – Діаграма послідовності «Оцінювання тривалості та збереження»

На рисунку 3.5 продемонстрована діаграма послідовності «Завантаження даних» - яка описує що повинен робити користувач під час

занесення даних. Користувач повинен вибирати файл, далі перевіркою даних займається система, котра і поверне результат.

Також, діаграма послідовності «Оцінювання тривалості та збереження» продемонстрована на рисунку 3.6 - яка описує порядок дій для оцінювання тривалість, спочатку проходить етап занесення даних, валідація та саме обчислення даних, на кінець, система виводить розрахунки та користувач може зберегти ці дані.

3.2.2 Розробка специфікації класів програмного забезпечення

Трирівнева архітектура — це модель проектування програмного забезпечення та структурна архітектура програмного забезпечення (рисунок 3.6).

Три рівні в трирівневій архітектурі:

1) рівень презентаційний займає верхній рівень та відповідає за відображення інформацію;

2) програмний рівень, називається середнім рівнем, відповідає за логіку системи.

3) рівень даних, бази даних, який відповідає за зберігання та пошук інформації.

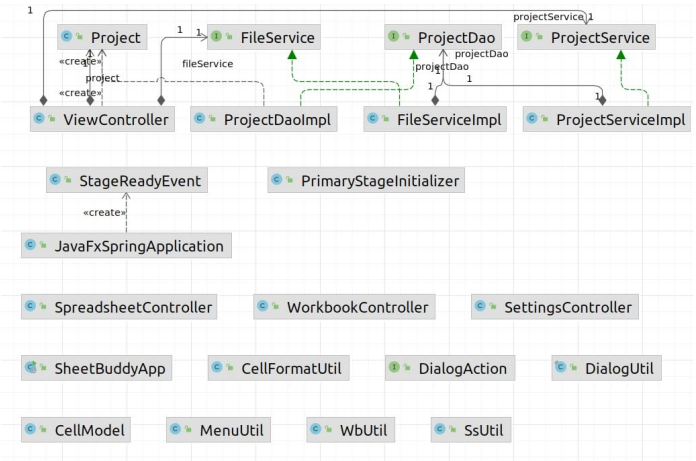


Рисунок 3.7 – Трирівнева архітектура ПЗ

Було використано принципи SOLID, які допомогли розбити ПЗ на 3 рівні, які можуть бути легко модифіковані. SOLID - акронім з п’яти літер які описуюсь принципи об’єктно-орієнтованого проектування, популяризованих Робертом Мартіном. (рисунок 3.8)

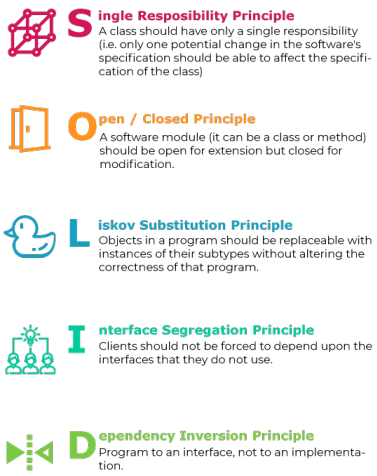


Рисунок 3.8 – SOLID

Їх використання робить кодову базу програми легшою для розуміння, більш гнучкою та придатною для розширення та подальшої підтримки.

3.3 Робочий проект програмного забезпечення

3.3.1 Обґрунтування вибору мови програмування та СКБД

Одну із головних ролей у створенні програмного забезпечення відіграє архітектура та мова програмування.

Мова програмування повинна взаємодіяти з архітектурою що допоможе у:

- вирішенні завдань;
- спрощенні читання коду;
- написання нового функціоналу;
- тестуванні;
- гнучкості.

Щоб вирішити ці питання, необхідно обрати мову, яка зарекомендувала себе на ринку – що дозволить мати рішення можливих проблем за короткий час; мова яка дозволяє швидко та легко виконувати складні математичні розрахунки. У цих умовах можна обрати мову програмування Java.

Java об'єктно-орієнтована мова програмування, взято за основу об'єктну модель C++. Допомагає створювати гнучкі програми за рахунок ООП та патернів. Розроблялась як платформо-незалежна мова має менше низькорівневих можливостей для роботи з апаратним забезпеченням, що дозволяє пришвидшити розробку та впровадження, не потребує багато знань для вивчення. За необхідності Java дозволяє викликати підпрограми, написані іншими мовами програмування. За довгі роки існування набула прихильників які створили бібліотеки для математичного оброблення даних, які допомагають в аналізі та оцінці.

Це не лише потужна мова програмування, розроблена з урахуванням питань безпеки, платформної сумлінності, але також постійно вдосконалений інструмент, що доповнюється новими можливостями та бібліотеками, які елегантно вписуються у вирішення традиційно складних завдань програмування: багатозадачності, доступу до баз даних, мережевого програмування та розподілених обчислень

Тому мова Java ідеальна для розробки програми для оцінюнки розміру web-застосунків, що створюються мовою Java.

Графічний інтерфейс було вирішено створити за допомогою потужної графічної бібліотеки JavaFX.

Необхідно створити ПЗ для оцінки для оцінювання розміру web-застосунків, що створюються мовою Java, щоб відповідати наступним вимогам:

– оцінка розміру web-застосунків: середня чисельність методів, загальне число класів;

– оцінка розміру web-застосунків, що створюються мовою Java;

– обчислення двохфакторної нелінійної регресійної моделі;

– ПЗ має працювати на комп’ютері з процесором Intel Pentium\Celeron, 1 Гб оперативної пам’яті, 512 Мб місця на жорсткому диску;

– працювати на різних ОС (старих, нових, UNIX, Windows);

– доступ до СКБД.

База даних - це структура для зміни, обробки, зберігання та інформації, переважно великих обсягів інформації. Головним питанням при виборі бази даних зазвичай є реляційна (SQL) чи нереляційна (NoSQL)

Реляційна база даних (SQL) - це база даних, у якій дані зберігаються у формі таблиць, які жорстко структуровані та пов’язані одна з одною. Таблиця містить рядки та стовпці, кожен з яких є записом, а стовпець є полем із призначеним типом даних. У кожній комірці інформація зберігається за шаблоном [30].

Нереляційна база даних (NoSQL) - зберігає дані, які не мають чіткого зв'язку один з одним або чіткої структури. Замість структурованих таблиць база даних містить багато різних документів, включаючи зображення, відео та навіть публікації в соціальних мережах. На відміну від реляційних баз даних, бази даних NoSQL не підтримують запити SQL [30].

На рисунку 3.9 продемонстрований SQL запит створення та внесення даних в базу для збереження та подальшого оцінювання розміру web-застосунків, що створюються мовою Java.

```
INSERT
    INTO estimation_models
    values (
        1,
        'WEB-застосунки, що створюються мовою java',
        -0.3963751428950725,
        0.3107761581101411,
        32,
        0.12033593450909232,
        2.9683499661246313,
        14.301774637586515
    );
```

Рисунок 3.9 – SQL запит створення та внесення даних в базу

Трирівнева архітектура дозволяє розробляюти та підтримувати на різних як незалежні модулі: функціональна логіка процесу, доступ до даних, зберігання комп’ютерних даних та інтерфейс користувача. незалежно оновлювати або замінювати будь-який із трьох рівнів.

3.3.2 Реалізація та тестування основних класів програмного забезпечення

На кожному етапі додавання функціоналу були створені Unit тести - метод тестування програмного забезпечення, який перевіряє кожен модуль програмного коду окремо. Модуль - це найменша частина програми, яку можна протестувати. У процедурному програмуванні модуль розглядається як одна функція або процедура. Метод – в об'єктно-орієнтованому програмуванні.

Модульні тести використовуються, щоб переконатися, що ваш код відповідає архітектурним вимогам і має очікувану поведінку.

Тестами було покрито 95% відсотків програми (рисунок 3.10).

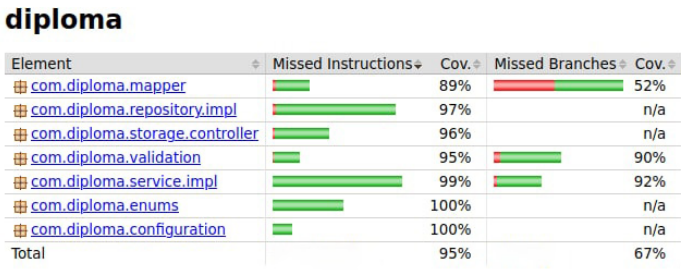


Рисунок 3.10 – Покриття ПЗ тестами

Для оцінювання розміру web-застосунків, що створюються мовою Java, було розроблено ескізний, технічний та робочий проекти програми (варіанти використання, діаграми послідовності, тощо).

Спроектоване та розроблене ПЗ для оцінювання розміру web-застосунків, що створюються мовою Java, було реалізоване також мовою

програмування Java, яка була обрана через її популярність, час розробки скоротять фреймворки та нададуть більше можливостей.

4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВІД РОЗРОБКИ І ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Впровадження інформаційних технологій пов'язано з капітальними вкладеннями як на придбання техніки, так і на розробку проектів, виконання підготовчих робіт і підготовку кадрів. Тому впровадженню повинно передувати економічне обґрунтування доцільності впровадження інформаційної системи. Це означає, що повинна бути визначена ефективність використання автоматизованих комп'ютерних технологій.

4.1 Розрахунок витрат на створення й експлуатацію програмного забезпечення

Витрати на розробку ПЗ складаються з витрат на зарплату розробника, на амортизацію комп`ютера, на якому виконується розробка, на експлуатацію цього комп`ютера, на засоби розробки та витратні матеріали і комплектуючі.

Розробка ПЗ виконується програмістом, місячний оклад якого складає 10000 грн. Додаткова заробітна плата складає 20% від основної. Виходячи з цього, основна і додаткова заробітна плата розроблювача системи 12000 грн/міс, а вартість сучасного комп`ютера складає 15000 грн. (приведена вартість комп`ютера на базі Intel Core i9).

Вартість розробки ПЗ розраховується по формулі:

$$Спр = (Ззп + Зсз + Ззг + Зе) * Т + Зм,$$

де Т – тривалість розробки, міс.;

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

Зсз – відрахування на соціальні заходи (38% від основної і додаткової заробітної плати), грн.;

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

Зе – витрати на електроенергію;

Зм – витрати на основні і допоміжні матеріали.

Розрахуємо Зе при споживаній потужності 0,5 кВт, тривалості роботи на місяць, рівної 22*6 = 132 годин, вартості кіловат-години електроенергії 1,68 грн. складає:

Зе = 132 * 1,68 * 0,5 = 110,88 грн.

Витрати на допоміжні матеріали приведені в табл. 4.1.

Таблиця 4.1 – Витрати на допоміжні матеріали

Пункти витрат	Сума, грн.
Папір	90,00
Заправлення картриджа до принтера	120,00
Література	450,00
Непередбачені витрати	250,00
Разом	910,00

Витрати на розрозку ПЗ приведені в табл. 4.2.

Таблиця 4.2 – Витрати на розробку системи

Найменування витрат	Одиниця виміру	Кількість
1	2	3
Тривалість розробки	Міс.	1
Основна і додаткова заробітна плата	Грн.	12000,00

Продовження табл. 4.2

1	2	3
Відрахування на соціальні заходи	Грн.	4560,00
Загальногосподарські витрати	Грн.	1000,00
Витрати на допоміжні матеріали	Грн.	910,00
Витрати на електроенергію	Грн.	110,88

Відповідно до формули, вартість розробки ПЗ складає:

Спр = (12000 + 4560 + 1000 + 110,88) * 1 + 910 = 18580,88 грн.

Амортизаційні відрахування на устаткування складають 60% балансової вартості в рік:

Аоб = 15000 * 0,6 = 9000 грн.

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

Вм = 15000 * 0,05 = 750грн.

Річний обсяг робіт комп`ютера у годинах визначається в такий спосіб:

Фм = 264,5 * Тз

де Тз – це середнє місячне завантаження устаткування (близько 4 годин); 264,5 – середня кількість робочих днів у році.

Отже, річний обсяг роботи комп`ютера складе:

Фм = 264,5 * 4 = 1058 годин.

Витрати на електроенергію Зе при 1058 годинах роботи устаткування в рік складуть

Зе = 1058 * 1,68 * 0,5 = 888,72 грн.

Експлуатаційні витрати для комп`ютера за рік складуть:

Ззр = 9000 + 750 + 888,72 = 10638,72 грн.

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

Зсе = 18580,88 + 10638,72 = 29219,60 грн.

4.2 Економічна ефективність розробки і впровадження програмного забезпечення

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

Крім багатьох інших негативних ефектів, ручна обробка інформації спричиняє наступні негативні економічні ефекти:

– високі витрати на складування паперових документів (сейфи, шафи, папки й ін.);

– підвищені витрати на канцтовари;

– витрати, пов'язані з коригуванням раніше допущених помилок (людський фактор).

До числа основних факторів, що визначають приріст прибутку в зв'язку з впровадженням ПЗ, відносяться:

– підвищення продуктивності праці;

– вивільнення робочого часу.

Крім того, не піддається прямій грошовій оцінці підвищення оперативності керування, якість одержуваних результатів, поліпшення організації праці і т.д.

Обов'язковою умовою визначення економічної ефективності ПЗ є порівнянність усіх показників у часі, за цінами й іншими нормами, використовуваними для визначення показників, за змістом і колом елементів витрат.

Визначимо пряму економічну ефективність, ґрунтуючись на тому, що впровадження ПЗ вивільняє 0,6 працівника (за експертною оцінкою фахівців).

Зарплата 0,6 працівника в рік складає:

(8000 * 12)* 0,6 = 57600 грн.

Річний економічний ефект розраховується по формулі:

Эгод = ΔCn − En · k

де ΔCn – вивільнені кошти після впровадження ПЗ (57600 грн.) мінус експлуатаційні витрати (10638,72 грн.) = 46961,28грн.

En – коефіцієнт ефективності (дорівнює коефіцієнту амортизації(0,6)); k – одноразові витрати на впровадження продукту (29219,60);

Эгод = 46961,28 – 29219,60 * 0,6 = 10645,01 (грн.)

Строк окупності системи розраховується по формулі:

$$T = \frac{k}{\Delta C} = \frac{29219,60}{46961,28} \approx 0,62$$

Отже строк окупності системи складає приблизно 7 місяців.

5 ОХОРОНА ПРАЦІ

5.1 Вступна частина

Охорона праці – ряд систем розрахованих на догляд за життям та здоров'ям робітників у приміщені де працюють люди. До охорони праці стосуються процеси трудової діяльності, які включають багато заходів щодо захисту людської праці, а саме правові, лікувально-профілактичні, санітарно-гігієнічні, соціально-економічні та інші [31].

До функцій охорони праці входить саме проведення цих заходів, задля зниження впливу шкідливих факторів на організм робітників, шляхом застосування технік безпеки під час виконання роботи працівниками.

До техніки безпеки можна віднести ряд організаційних та технічних методів гарантування безпеки, що при виконанні необхідних для техніки безпеки заходів запобігають відсутністю небезпечних факторів, або зниження можливих випадків.

Робоче місце – місце постійного або тимчасового перебування працівника під час його трудової діяльності, в якому виконують виробничі завдання. Робоче місце є частиною виробничо-технологічної структури підприємства (організації), воно призначене для виконання частини технологічного (виробничого) процесу і визначається на основі трудових та інших діючих норм і нормативів.

Робоча зона – визначений простір, у якому розташовані робочі місця постійного або тимчасового перебування працівника під час його трудової діяльності, обмежений по висоті над рівнем підлоги або майданчика. До постійних відносяться робочі місця, на яких працює знаходиться більше 50% робочого часу за зміну або більше двох годин безперервно. Якщо робота 80 здійснюється в різних пунктах робочої зони, то постійним робочим місцем вважається вся робоча зона.

Умови праці – сукупність факторів виробничого середовища, що впливає на здоров'я і працездатність людини в процесі праці. Дослідження умов праці показали, що чинниками виробничого середовища в процесі праці є:

- санітарно-гігієнічна обстановка, яка визначає зовнішню середу в робочій зоні - мікроклімат, механічні коливання, випромінювання, температуру, освітлення та інші;
- психофізіологічні елементи: робоча поза, фізичне навантаження, нервовопсихологічну напругу та інші, які обумовлені самим процесом праці;
- естетичні елементи: оформлення виробничих приміщень, обладнання, робочого місця, робочого інструменту та інші;
- соціально-психологічні елементи, складові характеристику так званого психологічного клімату.

Згідно Конституції України (ч. 4 ст. 43) кожна працездатна людина має право на належні, безпечні і здорові умови праці. А тому, у відповідності до вимог ст. 153 Кодексу законів про працю України, ст. 6 та ч. 1 ст. 13 Закону України «Про охорону праці», роботодавець зобов’язаний створити на робочому місці в кожному структурному підрозділі умови праці відповідно до нормативно-правових актів, а також забезпечити додержання вимог законодавства щодо прав працівників у галузі охорони праці.

Отже, треба провести аналіз та виявити шкідливі та небезпечні фактори, які можуть впливати на працездатність, загрожувати здоров’ю чи життю працівників на робочих місцях у офісі, провести аналіз їх впливу на здоров’я, встановити граничнодопустимі рівні цих факторів на робочих місцях, тим самим визначити оптимальні умови необхідні для праці та навчання.

5.2 Аналіз шкідливих та небезпечних факторів в офісі фірми

Для аналізу шкідливих та небезпечних факторів необхідно розглянути характеристики умов праці робітників, а також можливі фактори, що можуть негативно вплинути на їхнє здоров’я.

Під умовами праці розуміють сукупність факторів та обставин трудового процесу і виробничого середовища, в якому здійснюється діяльність людини, що впливають на здоров’я та працездатність.

Під факторами трудового процесу розуміють основні його характеристики: важкість праці та напруженість праці. Важкість праці – це характеристика трудового процесу, яка відображає переважне навантаження на опорно-руховий апарат і функціональні системи організму (серцево-судинну, дихальну та інші, що забезпечують його діяльність). Напруженість праці – це характеристика трудового процесу, яка відображає навантаження переважно на центральну нервову систему, органи чуття, емоційну сферу працівника.

Під виробничим середовищем розуміють сукупність фізичних, хімічних, біологічних, психофізіологічних факторів на виробництві, що діють на людину. Усі ці фактори класифікують як небезпечні та шкідливі.

Небезпечні виробничі фактори – це ті, вплив яких на працівника призводить до травм, раптового погіршення здоров’я чи до смерті.

Шкідливі виробничі фактори – це ті, вплив яких на працівника може призвести до захворювання та зниження працездатності.

До фізичних небезпечних і шкідливих факторів належать:

– рухомі машини і механізми, рухомі частини виробничого обладнання, вироби, що пересуваються (матеріали, заготовки);

– підвищена запиленість і загазованість повітря робочої зони;

– підвищена чи знижена температура поверхонь обладнання, матеріалів, повітря робочої зони;

– підвищені рівні шуму, вібрації, ультразвуку, інфразвукових коливань;

– підвищений чи знижений барометричний тиск і його різкі зміни;

– підвищена чи знижена вологість, рухомість, іонізація повітря;

– підвищений рівень іонізуючих випромінювань, напруги в електромережі, статичної електрики, електромагнітних випромінювань, напруженості електричного і магнітного полів;

– відсутність чи брак природного світла, знижена контрастність, пряма і відбита блискотливість, підвищена пульсація світлового потоку;

– підвищені рівні ультрафіолетової та інфрачервоної радіації;

– гострі краї, шершавість на поверхні заготовок, інструментів і обладнання;

– розташування робочого місця на значній висоті відносно землі (підлоги).

До хімічних небезпечних і шкідливих виробничих факторів належать речовини, які за характером впливу на організм людини поділяються на токсичні, подразнюючі, сенсабілізуючі, концерогенні і мутагенні. Ці хімічні речовини впливають на репродуктивну функцію людини. За шляхами проникнення в організм людини вони поділяються на ті, що проникають через органи дихання, шлунково-кишковий тракт, шкіряний покрив і слизові оболонки.

До біологічних небезпечних і шкідливих виробничих факторів належать патогенні мікроорганізми (бактерії, віруси, рикетсії, спірохети, грибки) та продукти їх життєдіяльності, а також рослини та живі істоти.

До психофізіологічних небезпечних і шкідливих виробничих факторів належать фізичні (статичні і динамічні) та нервово-психічні перевантаження (розумове перенапруження, перенапруження аналізаторів, монотонність праці, емоційні перевантаження).

До небезпечних факторів стосується і гігієнічні умови праці робочих місць, що негативно впливають на здоров'я людей, які там працюють.

Гігієнічне оцінювання умов і характеру праці на робочих місцях виконується на основі гігієнічної класифікації праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу. Гігієнічна класифікація базується на принципі диференціації умов праці залежно від фактично визначених рівнів указаних факторів у порівнянні з санітарними нормами, правилами, гігієнічними нормами, а також з урахуванням можливого шкідливого впливу їх на стан здоров'я працівників.

Умови праці та принципи гігієнічної класифікації поділяються на чотири класи.

Оптимальні умови праці (1-й клас) – умови, за яких зберігається не лише здоров'я працівників, а й створюються передумови для підтримання високого рівня працездатності. Оптимальні гігієнічні нормативи виробничих факторів встановлені для мікроклімату і факторів трудового процесу. Для інших факторів за оптимальні умовно приймаються такі умови праці, за яких несприятливі фактори виробничого середовища не перевищують рівнів, прийнятих за безпечні для населення.

Допустимі умови праці (2-й клас) – характеризуються такими рівнями шкідливих виробничих факторів виробничого середовища і трудового процесу, які не перевищують встановлених гігієнічних нормативів, а можливі зміни функціонального стану організму відновлюються за час регламентованого відпочинку чи до початку наступної зміни та не чинять несприятливого впливу на стан здоров'я працівників та їх потомство в найближчому і віддаленому періодах.

Шкідливі умови праці (3-й клас) – характеризуються такими рівнями шкідливих виробничих факторів, які перевищують гігієнічні нормативи і здатні несприятливо впливати на організм працівника та на його нащадків. Шкідливі умови праці за ступенем перевищення гігієнічних нормативів і вираженості можливих змін в організмі працівників поділяються на чотири ступені.

Перший ступінь – умови праці характеризуються таким рівнем шкідливих факторів виробничого середовища і трудового процесу, які, як правило, зумовлюють функціональні зміни, що виходять за межі фізіологічних коливань (останні відновлюються при тривалішій, ніж початок наступної зміни, перерві контакту зі шкідливими факторами) та збільшують ризик погіршення здоров'я.

Другий ступінь – умови праці характеризуються такими рівнями шкідливих факторів виробничого середовища і трудового процесу, які здатні спричинювати стійкі функціональні порушення, призводять у більшості випадків до зростання виробничо-зумовленої захворюваності, появи окремих ознак або легких форм професійної патології (як правило, без втрати професійної працездатності), що виникають після тривалої експозиції (10 років і більше).

Третій ступінь умови праці характеризуються такими рівнями шкідливих факторів виробничого середовища і трудового процесу, які призводять,окрім зростання виробничо-зумовленої захворюваності, до розвитку професійних захворювань, як правило, легкого і середнього ступенів важкості (з утратою професійної працездатності в період трудової діяльності). Четвертий ступінь – умови праці характеризуються такими рівнями шкідливих факторів виробничого середовища і трудового процесу, які здатні призводити до значного зростання хронічної патології та рівнів захворюваності з тимчасовою втратою працездатності, а також важких форм професійних захворювань (з утратою загальної працездатності).

Небезпечні (екстремальні) умови праці (4-й клас) – умови праці характеризуються такими рівнями шкідливих факторів виробничого середовища і трудового процесу, вплив яких протягом робочої зміни (чи її частини) створює загрозу для життя, високий ризик виникнення важких форм професійних уражень.

Поширена думка, що офісні робітники працюють в приємній, безпечної обстановці. Хоча працювати в офісі не так небезпечно, як у багатьох інших

місцях, тут також існує ряд джерел ризику для життя і здоров'я. Деякі з них представляють для людей, які працюють в офісі серйозну небезпеку.

Для працівника несприятливими подіями внаслідок впливу умов праці є втома, стрес, захворювання (хвороба) або травма.

Втома – це фізіологічний стан організму, що виникає внаслідок надмірно інтенсивного або тривалої діяльності, що виявляється тимчасовим зниженням функціональних можливостей людського організму. Розрізняють фізичну, розумову і емоційну втому.

Фізична втома проявляється порушенням функції м'язів: зниженням сили, точності, узгодженості і ритмічності рухів; виникає при інтенсивній або тривалій фізичної діяльності.

Розумова втома проявляється зниженням продуктивності інтелектуальної праці, ослабленням уваги (труднощі зосередження), уповільненням мислення, зниженням показників розумової активності, інтересу до роботи; виникає при інтенсивній інтелектуальній діяльності.

Емоційна втома проявляється помітним зниженням емоційних реакцій під впливом надсильних або монотонних подразників (стресів).

Надмірне робоче навантаження протягом тривалого часу або недостатній час відпочинку може привести до хронічної втомиабо перевтоми. Розрізняють розумовий і психічний (душевний) перевтома. В умовах сучасного ритму праці і життя все частіше у працівників з'являється синдром хронічної втоми.

Стрес є серйозною проблемою соціально-психологічного характеру для багатьох організацій. Він може бути викликаний багатьма факторами, включаючи шум від переповнення приміщення людьми або працюючого обладнання, поганих відносин з начальством та / або колегами, збільшенням робочого навантаження і недостатньою свободою дій при виконанні роботи.

Іншим масово поширеним несприятливим наслідком праці є те або інше захворювання: нездужання, погане самопочуття, симптоми, які можуть

проходити бурхливо або відносно швидко минути («гострі» -з медичної термінології) або тривати (роками) або періодично загострюватися (хронічні).

Захворювання, спровоковані умовами праці, часто називають виробничозумовленими захворюваннями.

Специфічний вплив факторів, пов'язаний з конкретними виробничими чинниками, призводить до розвитку певних, викликаних цими факторами, захворювань. Оскільки вони викликані несприятливими умовами праці конкретних робочих місць конкретних професій, то їх називають професійними захворюваннями.

Захворювання кістково-м'язової системи та пошкодження м'яких тканин типу тендиніту виникають в результаті використання офісних меблів та обладнання, які не відповідають індивідуальним анатомічним особливостям. Тендинит може розвинутися в результаті повторюваних рухів окремих частин тіла. Так, наприклад, у службовців починають боліти пальці від тривалого писання або при роботі з дуже товстими папками, які доводиться постійно діставати із шаф і повертати на місце. Багато офісних робітників страждають від різноманітних RSI, типу захворювань зап'ястя і грудної клітини, через погану підгонки меблів і устаткування і відсутності перерв у друкуванні (на клавіатурі комп'ютера) або інших періодично повторюваних діях. Недосконала конструкція меблів і устаткування призводять також до порушення постави і здавлення нервів нижніх кінцівок, тому що багато офісних робітників велику частину часу 87 проводять сидячи. Всі ці фактори сприяють виникненню захворювань попереку і нижніх кінцівок в такій же мірі, як і тривале стояння на ногах.

Безперервне використання комп'ютерів і погане загальне освітлення викликає напругу зору, в результаті чого воно поступово погіршується, з'являються головні болі, печіння і відчуття втоми в очах. Регулювання рівня освітленості приміщення і контрастності зображення на екрані комп'ютера, а також часта перефокусування очей необхідні для попередження виникнення проблем із зором. Освітлення має відповідати характеру виконуваної роботи.

5.3 Розрахунок рівня освітлення в офісі фірми

Рівень освітленості впливає на працездатність і самопочуття людини. Державні норми це враховують і визначають, яким має бути освітлення приміщень, залежно від їх призначення. Відштовхуючись від цих норм, планують освітлення об'єктів, а отже, від них безпосередньо залежить і те, які світильники і скільки лампочок необхідно купити.

Стосовно ДБН (Державно будівельні норми) є конкретні показники якості освітлення приміщень різних призначень: житлових, офісних, складських, виробничих. Спираючись на ці дані, розраховують рівень освітленості, враховуючи площу приміщення, висоту стелі, відбиваючу здатність поверхонь, тип світильника та інші параметри. Стосовно необхідного нам показника в офісних приміщеннях, нормою освітлення буде 300 Лк.

Норми виникли після того як були враховані всі найбільш важливі особливості фізіологічних зорових процесів людини і фактори, які впливають на концентрацію, зорову напругу під час роботи, відпочинку (наприклад, контрастність, колір робочої поверхні).

Для розрахунку рівня освітленості приміщення зазвичай використовується спосіб, який передбачає використання даних про величину світлового потоку, який вимірюється люменами (Лм). Світловий потік, що випромінює лампа, який вказується на упаковці придбаної лампи.

Для визначення величини світлового потоку використовується формула: Норма освітлення * Площа * Коефіцієнт висоти стелі

При висоті стелі від 2,5 м до 2,7 м коефіцієнт дорівнює 1, при висоті 2,7- 3 метрів - 1,2, якщо висота становить від 3 до 3,5 метрів - 1,5, від 3,5 м до 4,5 м - 2. Порахуємо, яка величина світлового потоку необхідна вітальні площею 30 квадратних метрів з висотою стель 2,5 м:

150 * 30 * 1 = 4500 (Лм).

Оскільки різні лампочки випромінюють світло різної сабо, потрібна їх кількість залежить від того, що ми виберемо – світлодіодну, енергозберігаючу, галогенну або лампу розжарювання, а також від того, якої потужності вона буде.

Таблиця 5.1 – Приклад різних видів ламп і їх потужності

Лампочка	Потужність, Вт	Світловий потік, Лм	Потрібно лампочок, шт
Розжарювання	40	470	4500/470=9,5
Галогенна	32	470	4500/470=9,5
Енергозберігаюча	15	700	4500/700=6,4
Світлодіодна	10	750	4500/750=6

Отже, для освітлення вітальні площею 30 квадратних метрів необхідно 10 галогенних ламп потужністю 30 Вт, стільки ж більш потужних лампочок розжарювання, 7 енергозберігаючих по 15 Вт кожна або 6 світлодіодних 10-

89 ватних лампочок. Після підрахунків підсумок з десятковими знаками завжди округляємо в більшу сторону.

Більш складний спосіб розрахунку включає більше змінних:

N = (E * S) / (U * p * Fi * Кз),

де N – кількість світильників,

E – норма освітленості,

S – площа об'єкта,

Кз – коефіцієнт запасу (залежить від виду лампочки, ступеня забруднення приміщення),

U – коефіцієнт використання світлового потоку (залежить від висоти стін, площі кімнати, типу світильника, кольору покриття стелі, стін, підлоги),

p – кількість патронів світильника,

Fi – значення світлового потоку однієї лампи.

Наприклад, для вітальні з першого прикладу планується встановити точкові світильники. Колір покриття стелі - білий, стіни світлі, на підлозі - темно-коричневий ламінат. Для об'єкта показник Е становить 150 (дивимося в таблиці), S – 30, Кз – 1 (припускаємо, що купують світлодіодні лампи потужністю 10 Вт), U – 0,46, p – 1, Fi – 750.

Підставляємо дані:

N = (150 * 30) / (0,46 * 1 * 750 * 1) = 4500 / 345 = 13.

Виходячи зі розрахованої формули було знайдено оптимальне значення кількості світильників, тобто слід розмістити 13 світильників.

5.4 Розробка заходів щодо зменшення впливу шкідливих та небезпечних факторів

З метою запобігання або зменшення впливу шкідливих і небезпечних виробничих чинників під час підготовки робочих місць, або робочої зони, необхідно слідувати загальним вимогам, що встановлені законодавством Україні про працю.

Відповідно до законів України «Про охорону праці» умови праці на робочому місці, безпека технологічних процесів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам законодавства.

Більшість нормативів щодо умов праці офісних працівників встановлено на рівні державних стандартів. Відповідно до ч. 1 ст. 13 Закону про охорону праці роботодавець зобов'язаний створити на робочому місці в кожному структурному підрозділі умови праці відповідно до нормативно-правових актів.

Як відомо, тривала робота за комп'ютером та з документами при недостатньому рівні освітленості може призвести до значного

перенапруження зору, тому вимоги до освітлення є досить важливими. Додатково, окрім вже перелічених документів, вимоги до освітлення встановлено ДБН В.2.5-28-2006 «Природне і штучне освітлення», затвердженими наказом Мінрегіону від 15.05.2006 р. № 168.

Як вже зазначалося, відносно вікон робоче місце необхідно організовувати так, щоб природне світло було з лівого боку (п. 4.3 ДСанПіН 3.3.2.007-98). Робоче місце необхідно розміщувати таким чином, щоб уникнути попадання прямого світла в очі. Для забезпечення захисту і досягнення нормованих рівнів комп'ютерних випромінювань необхідне застосування приекранних фільтрів, локальних світлофільтрів (засобів індивідуального захисту очей) та інших засобів захисту, що пройшли випробування в акредитованих лабораторіях і мають щорічний гігієнічний сертифікат (п. 4.19 ДСанПіН 3.3.2.007-98).

Штучне освітлення приміщення має здійснюватись системою загального рівномірного освітлення (п. 3.2.2 ДСанПіН 3.3.2.007-98). У приміщеннях при переважній роботі з документами допускається використання системи комбінованого освітлення, тобто встановлення світильників місцевого освітлення додатково до загального.

Як джерела штучного освітлення необхідно використовувати люмінесцентні лампи. Згідно з п. 3.2.5 ДСанПіН 3.3.2.007-98 система загального освітлення має бути у вигляді суцільних або переривчатих ліній світильників, що розташовані збоку від робочих місць (зазвичай ліворуч) паралельно лінії зору працівників.

Коефіцієнт пульсації не повинен перевищувати 5 % (п. 3.2.14 ДСанПіН 3.3.2.007-98). Рівень освітленості на робочому столі в зоні розташування документів має бути в межах 300 – 500 лк. Світильники місцевого освітлення слід встановлювати таким чином, щоб не створювати відблисків на поверхні екрана, а освітленість екрана має не перевищувати 300 лк.

Для забезпечення нормованих значень освітленості у приміщеннях відповідно до п. 3.2.15 ДСанПіН 3.3.2.007-98 необхідно мити вікна і

світильники не рідше 2 разів на рік, а також своєчасно замінювати лампи, що перегоріли.

6 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

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

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

Відносини у галузі охорони навколишнього природного середовища в Україні регулюється Законом України № 1268-XII від 26.06.91р. «Про охорону навколишнього природного середовища» [32], а також розроблюваними відповідно до нього земельним, водним, лісовим законодавством, законодавством про надра, про охорону атмосферного повітря, про охорону і використання рослинного і тваринного світу та іншим спеціальним законодавством.

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

Завданням законодавства про охорону навколишнього природнього середовища є регулювання відносин у галузі охорони, використання і відтворення природних ресурсів, забезпечення екологічної безпеки, запобігання і ліквідації негативного впливу господарської та іншої діяльності та навколишнє природне середовище, збереження природних ресурсів, генетичного фонду живої природи, ландшафтів та інших природних

комплексів, унікальних територій та природних об’єктів, пов’язаних з історико-культурною спадщиною.

Охорона навколишнього природного середовища є системою державних і суспільних заходів, направлених на збереження, відтворювання і раціональне використання природних ресурсів і поліпшення полягання природного середовища, і є частиною прикладної екології.

В міру прискорення темпів науково-технічного прогресу дія людей на природу стає все більш могутньою. І в даний час воно все сумарно із дією природних чинників, що приводить до якісної зміни співвідношення сил між суспільством і природою. На сучасному етапі людство поставлено перед чинником виникнення в природі незворотних процесів, нових шляхів переміщення і перетворення енергії і речовини. В природу втручається все більше і більше нових речовин, чужих їй, часом сильно токсичними для організмів. Частина з них не включається в природний круговорот і нагромаджується в біосфері, що приводить до небажаних екологічних наслідків.

6.1 Забруднення навколишнього середовища в процесі використання комп’ютерної техніки

Основну загрозу для навколишнього середовища в межах галузі розробки ПЗ ставлять ПК, а саме електромагнітні випромінювання від них.

Дослідження останніх років показано, що електромагнітні випромінювання вищезгаданих електричних пристроїв містять торсіонову компоненту, котра несе інформацію про процеси, що відбуваються в тому чи іншому електронному пристрої.

Торсіонові поля мають високу проникаючу здатність і їх неможливо заекранувати.

Останніми роками при проектування, виготовленні і експлуатації технічних і систем управляння в різних сферах діяльності надзвичайно широко застосовуються персональні комп’ютери і всілякі комп’ютерні програми.

Робота з комп’ютерами програмістів, операторів і інших користувачів пов’язана з додатковими шкідниками і небезпечними чинниками, що негативно впливають на організм людини. Наприклад, шкідлива дія на зір надає не притримання стандартних візуальних ергономічних параметрів екрану, розмірів мінімального елементу відображення, мерехтіння зображення, відбивна здатність і ін. Працюючий комп’ютер створює електромагнітне поле, що шкідливо діє на організм людини. Це поле може виникати радіоперешкоди, тобто завантажити роботі радіо- і телеаппаратури, що призводить до зниження надійності технічної системи або системи управління, до збільшення риски виникнення аварійної ситуації і виробничому середовищі. Для забезпечення безпеки роботи з комп’ютером розроблені і повинні повсюди застосовуватися стандарти на монітори, вимоги до приміщення для експлуатації комп’ютерів і до організації і успадкування робочих місць.

6.2 Використання програмного забезпечення для вирішення екологічних проблем

Широко використовуються фахівцями-екологами програми розрахунку забруднення атмосфери та інші спеціалізовані екологічні програми на рівні окремого підприємства досить повно вирішують завдання, що стоять в цій області.

Розроблена фірмою «Інтеграл» комп'ютерна система розрахункового моніторингу стану якості повітряного басейну міста (регіону) «Еколог-Місто» призначена для автоматизації діяльності територіальних

природоохоронних органів, екологічних служб адміністрацій міст (регіонів), проведення зведених розрахунків забруднення атмосфери міст.

Результати зведених розрахунків забруднення атмосфери можуть бути використані з метою нормування викидів забруднюючих речовин і для вирішення інших завдань.

Система «Еколог-місто» складається з двох підсистем:

- підсистеми ведення банку даних;
- підсистеми отримання попередньої оцінки якості атмосферного повітря.

Взаємодія між підсистемами здійснюється як на рівні обміну інформацією, так і на рівні передачі управління.

Основним завданням підсистеми ведення банку даних є підготовка інформації про викиди шкідливих речовин в атмосферу та інших даних, необхідних для коректної роботи підсистеми отримання оцінок забруднення атмосфери.

Основними особливостями файлів даних, записаних в розробленому форматі є:

1) універсальність з точки зору можливості використання різних програмних засобів при створенні і перетворенні файлів. Текстові файли даних можуть бути створені і заповнені або за допомогою програм системи «Екологмісто» або за допомогою будь-якого текстового редактора;

2) сегментування інформації в файлах.

Використовується інформація наступних видів:

- про параметри джерел забруднення атмосфери;
- про інвентаризацію викидів забруднюючих речовин;
- про умови розсіювання домішок в атмосфері;
- про використовувані системах координат;
- про місцезнаходження на місцевості зон зі специфічними вимогами до якості атмосферного повітря;

- про узагальнених характеристиках міста, його районів, підприємств, їх майданчиків і цехів.

За рівнем описуваних об'єктів інформація сегментована на розділи, що містять опис:

- міста (регіону);
- району міста (регіону);
- підприємства;
- майданчики і цеху підприємства;
- джерела забруднення атмосфери, що належить конкретній ділянці (можливо, також і цеху) підприємства.

Підсистема ведення банку даних складається з чотирьох самостійних великих блоків, взаємодія між якими здійснюється на рівні обміну інформацією:

- блоку підготовки вихідної інформації;
- блоку ведення міського банку даних;
- блоків перетворення даних про джерела викидів з форматів, використовуваних в програмі УПРЗА "Еколог" та інших програмах в єдиний формат даних системи "Еколог-Місто";

Блок підготовки вихідної інформації надає користувачеві наступні можливості:

1) занести дані про джерела викидів підприємства або прийняти дані про джерела викидів підприємства в текстовому форматі з файлу на дискеті або на жорсткому диску;

2) провести перевірку коректності занесених даних;

3) записати занесені дані в текстовий формат для надання їх в цьому форматі на дискеті в територіальні органи охорони навколишнього середовища;

4) блок ведення міського банку даних дозволяє:

5) прийняти інформацію про джерела викидів підприємств з файлу в єдиному форматі, INT, або підготувати її безпосередньо;

6) переглянути інформацію у вигляді таблиць і при необхідності її доповнити;

7) переглянути взаємне розташування джерел на тлі топооснови місцевості як для окремого підприємства, так і для сукупності підприємств.

ВИСНОВКИ

В кваліфікаційній роботі вирішене важливе питання інженерії програмного забезпечення, а саме удосконалено нелінійну регресійну модель для оцінювання розміру web застосунків, що створюються мовою Java, на основі нормалізуючого перетворення десяткового логарифму і викидів регресії. Ця двохфакторна нелінійна регресійна модель у порівнянні з однофакторною нелінійною регресійною моделлю та існуючою нелінійною регресійною моделлю, яка була побудована для оцінювання розміру web застосунків, що створюються мовою Java, має більше значення для параметрів якості R^2 та *PRED*(0,25) та менше значення для параметра якості *MMRE*.

У процесі виконання кваліфікаційної роботи задачі, які були поставлені, виконані у повному обсязі:

– проаналізовано існуючі моделі для оцінювання розміру web застосунків;

– обґрунтувано необхідність удосконалення регресійної моделі для оцінювання розміру web застосунків, що створюються мовою Java;

– досліджено різні джерела з відкритим вихідним кодом (open source resources) та визначено web додатки, створені мовою Java;

– отримано (розраховано) метрики, які необхідні для дослідження;

– виконано передобробку отриманих даних, зокрема, перевірка їх на викиди;

– побудувано лінійну регресійну модель на основі обраного нормалізуючого перетворення для нормалізованих даних;

– побудувано нелінійну регресійну модель для оцінювання розміру web застосунків, що створюються мовою Java;

– порівняно отриману нелінійну регресійну модель з існуючими моделями за показниками якості;

– розроблено програму для оцінювання розміру web застосунків, що створюються мовою Java на основі побудованої нелінійної регресійної моделі.

Програмне забезпечення, розроблене в рамках кваліфікаційної роботи для оцінювання розміру web застосунків, що створюються мовою Java, дозволить скоротити час, пов’язаний з розрахунками.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Крупица А.В Актуальность применения АИС в работе интернет провайдера. *Гуманитарные научные исследования.* 2015. № 12. URL: http://human.snauka.ru/2015/12/13222 (дата звернення: 23.09.2022).

2. Що таке веб додаток? URL: http://www.ittexnoall.com/index.php/sozdanie-sajta/10-sozdanie-sajta/246-shcho-take-web-dodatok.html (дата звернення: 23.09.2022).

3. Разработка WEB-приложений. URL: http://clashdash.ru/directions/web-applications (дата звернення: 23.09.2022).

4. Živkovic A., Hericko M., Brumen B., Beloglavec S., Rozman I. The Impact of Details in the Class Diagram on Software Size Estimation. *Informatica.* 2005. Vol. 16, No. 2. P. 295–312.

5. Briand L.C., Morasca S., Basili V.R. Property Based Software Engineering Measurement. *IEEE Transaction on Software Engineering.* 2009. Vol. 22, no. 1. P. 68-86.

6. Hee Beng Kuan Tan, Yuan Zhao, Hongyu Zhang. Estimating LOC for information systems from their conceptual data models. In *Proceedings of the 28th International Conference on Software Engineering (ICSE '06)*, May 20-28, 2006, Shanghai, China, pp. 321-330. DOI: 10.1145/1134285.1134331

7. Prykhodko N.V., Prykhodko S.B. The non-linear regression model to estimate the software size of open source java-based systems. *Радіоелектроніка, інформатика, управління.* 2018. № 3. С.158-166. DOI https://doi.org/10.32838/2663-5941/2020.2-1/25

8. Приходько С.Б., Приходько Н.В., Смикодуб Т.Г. Чотирьохфакторна нелінійна регресійна модель для оцінювання розміру Java-застосунків з відкритим кодом. *Вчені записки ТНУ імені В.І. Вернадського. Серія: технічні науки.* 2020. Том 31 (70). Ч. 1. № 2. С.157-162.

9. Макарова Л.М., Приходько Н.В., Кудін О.О. Побудова нелінійної регресійної моделі для оцінювання розміру веб-додатків, реалізованих мовою Java. *Вісник Херсонського національного технічного університету.* 2019. Вип. 2 (69). С.145-153.

10. Makarova L.M., Andreeva A.S. Information technology for size estimation of web-applications implemented in Java. *Інновації в суднобудуванні та океанотехніці: матеріали IX Міжнародної науково-технічної конференції.* 2018. С. 405.

11. Котигробов С.В., Макарова Л.М. Двохфакторна нелінійна регресійна модель для оцінювання розміру web застосунків, що створюються мовою Java. *Матеріали V Всеукраїнської науково-практичної інтернет-конференції студентів, аспірантів та молодих вчених за тематикою «Сучасні комп'ютерні системи та мережі в управлінні»: збірка наукових праць* / Під редакцією А.А. Григорової. Херсон: Видавництво ФОП Вишемирський В.С. 2022. с. 139-140.

12. Веб-приложения – дань моде или будущее ПО? URL: http://article.techlabs.com.ua/50_17239.html (дата звернення: 29.09.2022).

13. App Download Data. URL: http://www.businessofapps.com/data/app-statistics/ (дата звернення: 29.09.2022).

14. Беллиньясо М. Разработка Web-приложений в среде ASP.NET. М.: «Диалектика», 2007. 640 с.

15. Briand L.C., Wieczorek I. Resource Estimation in Software Engineering. Encyclopedia of Software Engineering. John Wiley & Sons, Inc., 2002. 83 p.

16. Halsted M. H. Elements of software science (in Russian). Amsterdam: Elsevier North-Holland, 1977. 128 p.

17. Kaczmarek J., Kucharski M. Size and effort estimation for applications written in Java. *Information and SoftwareTechnology.* 2004. Vol. 46, Issue 9. P. 589–601.

18. Prykhodko N.V., Prykhodko S.B. Constructing the Nonlinear Regression Models on the Basis of Multivariate Normalizing Transformations. *Електронне моделювання.* 2018. Т. 40. № 6. С.99-108.

19. Магнус Я.Р., Катышев П.К., Пересецкий А.А. Эконометрика. М.: Дело, 2004. 576 с.

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

21. Prykhodko S., Prykhodko N., Makarova L., Pugachenko K. Detecting outliers in multivariate non-Gaussian data on the basis of normalizing transformations, in: *Proceedings of the 2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)*, Kyiv, Ukraine, 2017, 846-849. https://doi.org/10.1109/UKRCON.2017.8100366

22. Chatterjee S., Price B. Regression Analysis by Example. New York: John Wiley & Son, 1977. 228 p.

23. Гмурман В.Е. Теория вероятностей и математическая статистика. М.: Высшее образование, 2010. 479 с.

24. Приходько С.Б., Макарова Л.М., Пугаченко К.С. Методичні вказівки та завдання до виконання лабораторних робіт з дисципліни "Обробка експериментальних даних на комп`ютері". Миколаїв: НУК, 2018. 76 с.

25. The world's leading software development platform. URL: https://github.com/ (дата звернення: 27.07.2022).

26. Prykhodko S., Prykhodko N., Makarova L. Estimating the software size of open-source PHP-based systems using non-linear regression analysis. In *Proceedings of International Conference on Advanced Computer Information Technologies (ACIT 2018)*, June 1-3, 2018, Ceske Budejovice, Czech Republic, P. 199-202.

27. Моделювання UML. URL: http://ni.biz.ua/8/8_8/8_88488_modelirovaniya-UML.html (дата звернення: 26.09.2022).

28. Уніфікована мова моделювання. URL: https://bibliofond.ru/view.aspx?id=446563#1 (дата звернення: 25.09.2022).

29. Діаграма активності. URL: http://khpi-iip.mipk.kharkiv.edu/library/case/leon/gl7/gl7.html (дата звернення: 25.09.2022).

30. SQL чи NoSQL – ось в чому питання. URL: https://alternativescience.net/programming/242-sql-chy-nosql-os-v-chomu-pytannya/ (дата звернення: 10.10.2022).

31. Закон України «Про охорону праці» від 14.10.1992р № 2694-XII. (зі змінами від 19.08.2022). URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text

32. Закон України «Про охорону навколишнього природного середовища» від 26.06.91р № 1268-XII. (зі змінами від 10.07.2022). URL: https://zakon.rada.gov.ua/laws/main/1264-12#Text

ДОДАТОК А - ТЕХНІЧНЕ ЗАВДАННЯ

Вступ

Назва розроблюваного проекту: «Програмне забезпечення для оцінювання розмірів web-застосунків, що створюються мовою JAVA» , надалі програма. Галузь застосування – розробка web-застосунків.

1. Підстави для розробки

Розробка програмного забезпечення виконується на підставі завдання на кваліфікаційну роботу, виданого кафедрою ПЗАС НУК імені адмірала Макарова.

2. Призначення розробки

2.1 Експлуатаційне призначення

ПЗ дозволить автоматизувати та скоротити час розрахунків оцінювання розміру web-застосунків, що створюються мовою Java

2.2 Функціональне призначення

Функціональним призначенням програмного забезпечення є автоматизація розрахунків оцінювання розміру web-застосунків, що створюються мовою Java, покращення та легкість ведення даних, зберігання та використання інформації.

3. Вимоги до програмного забезпечення

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу виконуваних функцій

Програма повинна забезпечувати можливість виконання наведених нижче функцій:

1) Занесення даних.

2) Редагування даних.

3) Нормалізація даних.

4) Розрахунок верхньої та нижньої границь довірчого інтервалу.

5) Розрахунок верхньої та нижньої границь інтервалу прогнозування.

6) Розрахунок прогнозного значення розміру web-застосунку.

3.1.2 Вимоги до організації вхідних та вихідних даних

Вхідні та вихідні дані зберігаються у таблицях бази даних.

Вихідні дані повинні виводитися в відповідних місцях на створеній програмою формі.

3.2 Вимоги до надійності

При виникненні збоїв в роботі комп’ютера, відновлення нормальної роботи повинне проводитися після перезавантаження програмного продукту.

Для забезпечення надійності інформації повинна використовуватися СКБД, що забезпечує цілісність транзакцій і несе відповідальність за цілісність інформації.

У разі введення користувачем некоректної інформації програмний продукт повинен повідомити про помилку і надати можливість виправити її.

3.3 Вимоги до умов експлуатації

Необхідний рівень підготовки користувачів: мінімальні навики в користуванні комп’ютером.

3.4 Вимоги до складу та параметрів технічних засобів

Система повинна задовольняти вказаним вимогам на комп’ютері наступної мінімальної комплекції:

– CPU: Intel(R) Celeron(R) – 2.16 МГц;

– RAM: 2 GB;

– HDD: 10 GB вільного місця.

3.5 Вимоги до інформаційної та програмної сумістності

Для повноцінного функціонування системи необхідно використовувати ОС Windows версії не нижче 7.

Необхідна наявність встановленого Java Runtime Environment (JRE 1.8).

4. Вимоги до програмної документації

Програмна документація повинна містити:

– технічне завдання;

– текст програми;

– опис програми;

– інструкцію користувача;

– програму і методику випробувань ПЗ.

5. Стадії та етапи розробки

Стадії та етапи розробки представлені в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки програмного забезпечення

Етапи робіт	Контрольні точки
Обумовлення необхідності розробки	08.09.2022 - 12.09.2022
Розробка та затвердження технічного завдання	13.09.2022 - 20.09.2022
Розробка ескізного проекту	21.09.2022 - 21.10.2022
Затвердження ескізного проекту	22.10.2022 - 24.10.2022
Розробка технічного проекту	25.10.2022 - 15.11.2022
Затвердження технічного проекту	16.11.2022 - 18.11.2022
Розробка ПЗ	19.11.2022 - 29.11.2022
Розробка програмної документації	30.11.2022 - 10.12.2022
Випробування ПЗ	11.12.2022 - 14.12.2022

6 Порядок контролю та прийому

Контроль за аналізом та проектуванням кожної окремої частини програмного забезпечення відбувається на кожному етапі з урахуванням вимог, визначених у технічному завдані. Кожна стадія розробки повинна бути представлена в зазначені строки та узгоджена із замовником.

Прийом проводять відповідно до програми і методики випробувань і документують за допомогою протоколу проведення випробувань. У разі знаходження помилок підчас прийому програмного виробу складається акт про знайдені помилки, який підписуються представниками замовника і розробника і затверджуються керівником організації – замовника та організації – розробника. Розробник повинен на протязі не більше ніж 2 тижнів виправити зазначені помилки і оповістити замовника про повторне проведення перевірки.

ДОДАТОК Б - ОПИС ПРОГРАМИ

1 Загальні відомості

1.1 Позначення і найменування програми

Найменування: ПЗ для оцінювання розмірів web-застосунків, що створюються мовою JAVA.

1.2 Мови програмування

ПЗ написане на мові програмування Java.

2 Функціональне призначення

2.1 Класи вирішуваних завдань і призначення програми ПЗ повинно забезпечувати виконання наступних функцій:

1) Занесення даних.

2) Редагування даних.

3) Нормалізація даних.

4) Розрахунок верхньої та нижньої границь довірчого інтервалу.

5) Розрахунок верхньої та нижньої границь інтервалу прогнозування.

6) Розрахунок прогнозного значення розміру web-застосунку.

3 Використовувані технічні засоби

– CPU: Intel(R) Celeron(R) – 2.16 МГц;

– RAM: 2 GB;

– HDD: 10 GB вільного місця.

4 Вхідні і вихідні дані

Вхідні дані програми повинні знаходитись в окремому текстовому файлі. Вихідні дані програми мають відображатися на екрані користувача

ДОДАТОК В - ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ

1 Об’єкт випробувань

ПЗ для оцінювання розмірів web-застосунків, що створюються мовою JAVA.

2 Ціль випробувань

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

3 Вимоги до програми

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

Для проведення випробувань розглядали всі функції.

4 Вимоги до програмної документації

Для проведення випробувань надаються наступні документи:

– технічне завдання;

– опис програми;

– текст програми;

– інструкція користувача;

– програма та методика випробувань.

5 Засоби і порядок випробувань

Випробування програмного забезпечення повинні виконуватися в наступному порядку:

- виконуються інструкції до кожної функції;

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

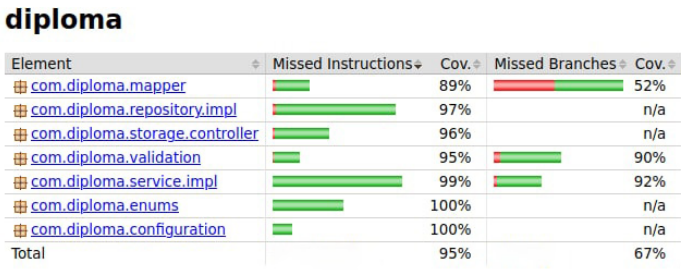
При виявлені помилок у роботі системи складається їх перелік і обговорюється термін їх виправлення розробником. Після цього замовник проводить повторне тестування в повному обсязі (можливе використання нових чи додаткових тестів).

6 Методи випробувань

Випробування програми виконується представником замовника в присутності представника розробника. Далі виконується перевірка роботи програми. Виконується перевірка всіх екранних форм програми, а також всіх допустимих для виконання операцій всередині екранних форм згідно до посібника користувача. Якщо програма демонструє адекватне функціонування згідно до вимог ТЗ та представленої програмної документації, то робота з розробки програми вважається виконаною, і виконується впровадження програми до експлуатації. В протилежному випадку програма має бути відправлено на доробку розробникові.

Також на кожному етапі додавання функціоналу були створені Unit тести - метод тестування програмного забезпечення, який полягає в окремому тестуванні кожного модуля коду програми. Модулем називають найменшу частину програми, яка може бути протестованою. У процедурному програмуванні модулем вважають окрему функцію або процедуру.

Тестами було покрито 95% відсотків програми (рисунок В.1).



diploma

Element	Missed Instructions	Cov.	Missed Branches	Cov.
com.diploma.mapper		89%		52%
com.diploma.repository.impl		97%		n/a
com.diploma.storage.controller		96%		n/a
com.diploma.validation		95%		90%
com.diploma.service.impl		99%		92%
com.diploma.enums		100%		n/a
com.diploma.configuration		100%		n/a
Total		95%		67%

Рисунок В.1 – Покриття ПЗ тестами

ДОДАТОК Г - ІНСТРУКЦІЯ КОРИСТУВАЧА

Після завантаження файлу JavaAppAnalyzer.jar - на екрані з'являється форма для відображення початкового стану програми для оцінювання розмірів web-застосунків, що створюються мовою JAVA (рисунок Г.1). Форма була створена на основі макету з рисунку 3.4.

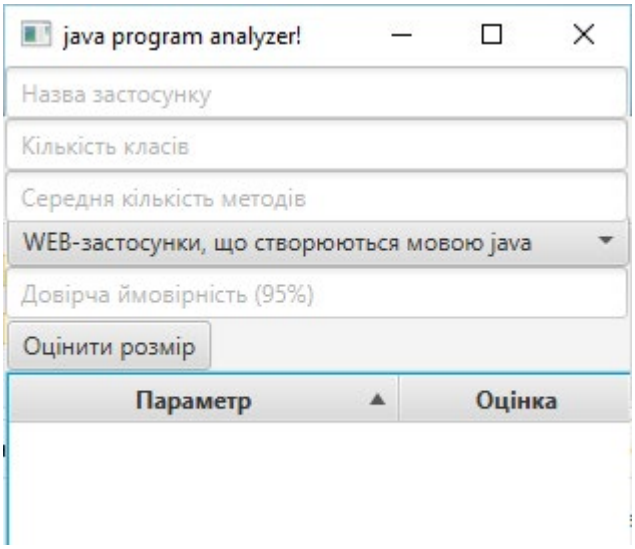


Рисунок Г.1 – Форма для відображення початкового стану програми

Інтерфейс ПЗ було написано так, щоби зробити інтерфейс користувача, який спрощує (самозрозумілий), ефективний і приємний (зручний для користувача) таким чином, щоби забезпечити бажаний результат. Це, зазвичай, означає, що користувач повинен застосовувати щонайменші зусилля для досягнення очікуваного результату, а також, щоби ПЗ зменшувало небажані результати для людини.

Після відкриття програми потрібно ввести дані на основі яких буде виконуватись аналіз. Дані вводяться послідовно:

1. Вести назву застосунку.
2. Ввести кількість класів.
3. Ввести середню кількість методів.
4. Вибрати модель.
5. Натисніть кнопку "Оцінити розмір".

Після натискання «Оцінити розмір» - ПЗ провалідує дані та надрукує результат, який буде видно в таблиці.

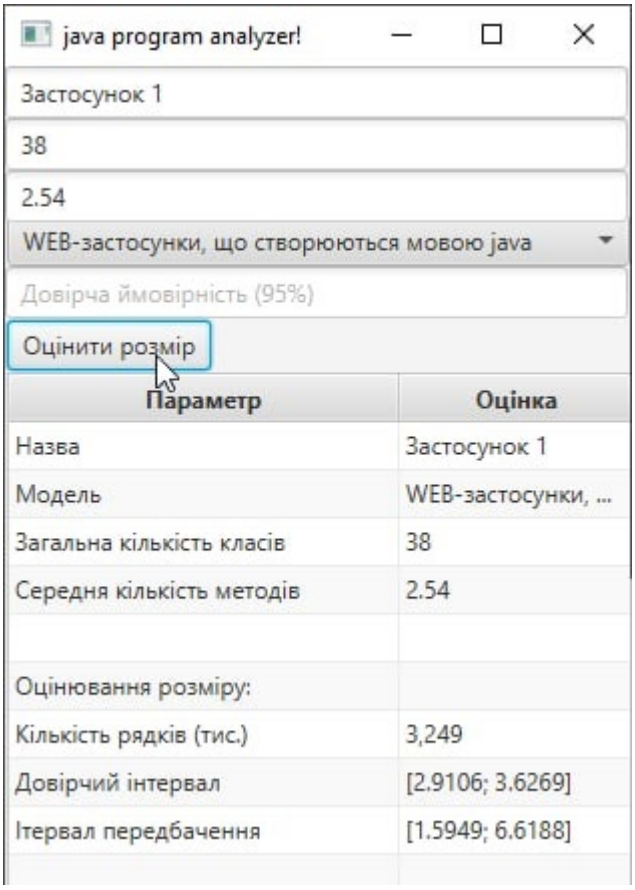


Рисунок Г.2 – Відображення результатів обчислень

ДОДАТОК Д - ТЕКСТ ПРОГРАМИ

```
public class DPController {
    @FXML
    protected void onButtonClick() {
        dataTable.setItems(data.list);
        models.setItems(FXCollections.observableList(data.models));
        models.getSelectionModel().selectFirst();
        appName.setText(data.appName);
        hours.setText(String.valueOf(data.hours));
        modelsCount.setText(String.valueOf(data.modelsCount));
        percent.setText(String.valueOf(data.percent));
    }

    @FXML
    private TextField appName;
    @FXML
    private TextField hours;
    @FXML
    private ChoiceBox<String> models;
    @FXML
    private TextField modelsCount;
    @FXML
    private TextField percent;
    @FXML
    private TableView<Row> dataTable;
    @FXML
    private TableColumn<Row, String> param;
    @FXML
    private TableColumn<Row, String> value;

    @FXML
    private void initialize() {
        param.setCellValueFactory(cellData -> new
ReadOnlyStringWrapper(cellData.getValue().getFieldName()));
        value.setCellValueFactory(cellData -> new
ReadOnlyStringWrapper(cellData.getValue().getFieldValue()));
    }

    public void start() {
        models.setItems(FXCollections.observableList(List.of("Модель")));
        models.getSelectionModel().selectFirst();

        // dataTable.setItems(Data.initDataTable);
    }
}
@Data
@JsonTypeInfo(include = JsonTypeInfo.As.WRAPPER_OBJECT, use = JsonTypeInfo.Id.CUSTOM, property
= "error", visible = true)
```

```
@JsonTypeIdResolver(LowerCaseClassNameResolver.class)
public class Error {

    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "dd-MM-yyyy hh:mm:ss")
    private LocalDateTime timestamp;
    private String message;
    private String debugMessage;
    private List<ApiSubError> subErrors;

    private Error() {
        timestamp = LocalDateTime.now();
    }

    Error(Throwable ex) {
        this();
        this.message = "Unexpected error";
        this.debugMessage = ex.getLocalizedMessage();
    }

    public Error(String message) {
        this();
        this.message = message;
    }

    Error(String message, Throwable ex) {
        this();
        this.message = message;
        this.debugMessage = ex.getLocalizedMessage();
    }

    public void addSubError(ApiSubError subError) {
        if (subErrors == null) {
            subErrors = new ArrayList<>();
        }
        subErrors.add(subError);
    }

    private void addValidationError(String object, String field, Object rejectedValue, String
message) {
        addSubError(new ApiValidationError(object, field, rejectedValue, message));
    }

    private void addValidationError(String object, String message) {
        addSubError(new ApiValidationError(object, message));
    }

    private void addValidationError(FieldError fieldError) {
        this.addValidationError(
                fieldError.getObjectName(),
                fieldError.getField(),
```

```
                fieldError.getRejectedValue(),
                fieldError.getDefaultMessage());
    }

    void addValidationErrors(List<FieldError> fieldErrors) {
        fieldErrors.forEach(this::addValidationError);
    }

    private void addValidationError(ObjectError objectError) {
        this.addValidationError(
                objectError.getObjectName(),
                objectError.getDefaultMessage());
    }

    void addValidationError(List<ObjectError> globalErrors) {
        globalErrors.forEach(this::addValidationError);
    }


    private void addValidationError(ConstraintViolation<?> cv) {
        this.addValidationError(
                cv.getRootBeanClass().getSimpleName(),
                ((PathImpl) cv.getPropertyPath()).getLeafNode().asString(),
                cv.getInvalidValue(),
                cv.getMessage());
    }

    void addValidationErrors(Set<ConstraintViolation<?>> constraintViolations) {
        constraintViolations.forEach(this::addValidationError);
    }
}
public abstract class SpringJavaFXApplication extends Application {
    protected ConfigurableApplicationContext applicationContext;

    @SuppressWarnings("unused")
    public static void launch(Class<? extends Application> appClass, String... args) {
        Application.launch(appClass, args);
    }

    @SuppressWarnings("unused")
    public static void launch(Class<? extends Application> appClass, Class<? extends
Preloader> preloaderClass,
            String... args) {
        System.setProperty("javafx.preloader", preloaderClass.getName());
        Application.launch(appClass, args);
    }

    @SuppressWarnings("unused")
    public static void launch(String... args) {
        Application.launch(args);
```

```
    }

    @Override
    public void init() {
        Parameters parameters = getParameters();
        SpringApplication application = new SpringApplication(this.getClass());

        application.setBannerMode(Banner.Mode.OFF);
        application.setHeadless(false);

        applicationContext = application.run(parameters.getRaw().toArray(new String[0]));
    }

    @Override
    public void start(Stage primaryStage) throws Exception {
        ViewManager viewManager = applicationContext.getBean(ViewManager.class);
        viewManager.registerPrimaryStage(primaryStage);
    }

    @Override
    public void stop() throws Exception {
        applicationContext.close();
        System.exit(0);
    }
}

@Slf4j
@EqualsAndHashCode
public class ViewManagerImpl implements ViewManager {
    public static final String PRIMARY_STAGE_PROPERTY = "primaryStage";
    public static final String POLICY_PROPERTY = "policy";

    private final ConfigurableApplicationContext applicationContext;
    private final List<Window> windows = new ArrayList<>();
    private final Property<Stage> primaryStage = new SimpleObjectProperty<>(this,
PRIMARY_STAGE_PROPERTY);
    private final Property<ViewManagerPolicy> policy = new SimpleObjectProperty<>(this,
POLICY_PROPERTY, ViewManagerPolicy.CLOSEABLE);

    //region Constructors

    public ViewManagerImpl(ConfigurableApplicationContext applicationContext) {
        this.applicationContext = applicationContext;
    }

    //endregion

    //region Properties

    @Override
```

```
public Optional<Stage> getPrimaryStage() {
    return Optional.ofNullable(primaryStage.getValue());
}

@Override
public ReadOnlyProperty<Stage> primaryStageProperty() {
    return primaryStage;
}

@Override
public Optional<Stage> getStage(String name) {
    return windows.stream()
            .filter(e -> e.getStage().getTitle().equalsIgnoreCase(name))
            .findFirst()
            .map(Window::getStage);
}

@Override
public ViewManagerPolicy getPolicy() {
    return policy.getValue();
}

@Override
public Property<ViewManagerPolicy> policyProperty() {
    return policy;
}

@Override
public void setPolicy(ViewManagerPolicy policy) {
    this.policy.setValue(policy);
}

@Override
public int getTotalWindows() {
    return windows.size();
}

//endregion

//region Methods

@Override
public void registerPrimaryStage(Stage primaryStage) {
    Assert.notNull(primaryStage, "primaryStage cannot be null");
    if (getPrimaryStage().isPresent()) {
        log.warn("Ignoring primary stage register as one has already been registered");
        return;
    }

    this.primaryStage.setValue(primaryStage);
```

```
        addWindowView(primaryStage, primaryStage.getScene(), true);
    }

    @Override
    public void addWindowView(Stage window, Scene view) {
        Assert.notNull(window, "window cannot be null");
        Assert.notNull(view, "view cannot be null");
        addWindowView(window, view, false);
    }

    //endregion

    //region Functions

    private void addWindowView(Stage window, Scene view, boolean isPrimaryStage) {
        window.setOnHiding(onWindowClosingEventHandler());
        windows.add(new Window(window, view, isPrimaryStage));
        log.debug("Currently showing " + getTotalWindows() + " window(s)");
    }

    private EventHandler<WindowEvent> onWindowClosingEventHandler() {
        return event -> {
            Stage stage = (Stage) event.getSource();
            Window window = this.windows.stream()
                    .filter(e -> e.getStage() == stage)
                    .findFirst()
                    .orElseThrow(() -> new StageNotFoundException(stage.getTitle()));

            this.windows.remove(window);
            log.debug("Currently showing " + getTotalWindows() + " window(s)");

            if (policy.getValue() == ViewManagerPolicy.CLOSEABLE) {
                if (window.isPrimaryWindow()) {
                    log.debug("Application closing, primary window is closed");
                    exitApplication();
                } else if (this.windows.size() == 0) {
                    log.debug("All windows closed, exiting application");
                    exitApplication();
                }
            }
        };
    }

    private void exitApplication() {
        try {
            Platform.exit();
        } catch (Exception ex) {
            log.error("Failed to stop application gracefully, " + ex.getMessage(), ex);
            System.exit(1);
        }
```

```
    }

    //endregion

    @Value
    @AllArgsConstructor
    private static class Window {
        private final Stage stage;
        private Scene scene;
        private final boolean primaryWindow;
    }
}

@Repository
public class ProjectDaoImpl implements ProjectDao {
    Logger logger = LoggerFactory.getLogger(getClass());

    @Autowired
    JdbcTemplate jdbcTemplate;


    @Override
    public void insertProject(Project p) {
        int rows = jdbcTemplate.update("INSERT INTO PROJECT (PROJECT_ID) VALUES (?)",
                p.getProjectId());
        logger.info(rows + " rows inserted into PROJECT");
    }

    @Override
    public void deleteProject(Project p) {
        int rows = jdbcTemplate.update("DELETE FROM PROJECT WHERE PROJECT_ID = ?",
p.getProjectId());
        logger.info(rows + " rows deleted from PROJECT");
    }

    @Override
    public Project getProjectById(String id) {
        List<Map<String, Object>> rowsList = jdbcTemplate.queryForList("SELECT * FROM PROJECT
WHERE PROJECT_ID = ?", id);

        return new Project(String.valueOf(rowsList.get(0).get("PROJECT_ID")),
getRecentFiles(id));
    }

    @Override
    public Boolean projectExistsById(String id) {
        try {
            Map<String, Object> matchProject = jdbcTemplate.queryForMap("SELECT * FROM PROJECT
WHERE PROJECT_ID = ?", id);
            if (matchProject.size() > 0) {
```

```
                return Boolean.TRUE;
            }
        } catch (IncorrectResultSizeDataAccessException e) {
            return Boolean.FALSE;
        }
        return Boolean.FALSE;

    }

    @Override
    public List<String> getRecentFiles(String id) {
        List<Map<String, Object>> selectedRows = jdbcTemplate.queryForList("SELECT * FROM
PROJECT_FILES WHERE " +
                "PROJECT_ID = ?", id);
        List<String> recentFiles = new ArrayList<>();
        for (Map map : selectedRows) {
            recentFiles.add(String.valueOf(map.get("RECENT_FILE")));
        }

        return recentFiles;
    }

    @Override
    public void insertRecentFile(String file, String id) {
        jdbcTemplate.update("INSERT INTO PROJECT_FILES (PROJECT_ID, RECENT_FILE) VALUES
(?,?)", id, file);

    }
}
@FxmlView("/fxml/workbookview.fxml")
public class WorkbookController {

    protected int numSheets;
    private final Logger logger = LoggerFactory.getLogger(WorkbookController.class);
    private final FxWeaver fxWeaver;
    private XSSFWorkbook currentWorkbook;

    /* References the default sheetview */
    private final FxControllerAndView<SpreadsheetController, SpreadsheetView>
sheetControlView;
    private final List<FxControllerAndView<SpreadsheetController, SpreadsheetView>>
sheetControls = new ArrayList<>();

    @Value("${sheet-buddy.start.title}")
    String defaultTitle;
    @FXML
    protected TabPane tabPane;
    @FXML
    protected Tab tab1;
```

```
    VBox rootNode;

    @Autowired
    public WorkbookController(FxWeaver fxWeaver,

@SuppressWarnings("SpringJavaInjectionPointsAutowiringInspection")
FxControllerAndView<SpreadsheetController,
                                    SpreadsheetView> sheetControlView) {
        this.fxWeaver = fxWeaver;
        this.sheetControlView = sheetControlView;
        sheetControls.add(sheetControlView);
    }

    @FXML
    public void initialize() {
        /* Initialize blank workbook */
        currentWorkbook = XSSFWorkbookFactory.createWorkbook();
        sheetControlView.getController().poiSheet =
currentWorkbook.createSheet(tab1.getText());
        sheetControlView.getController().sheetNumber = 1;
    }

    protected void updateWorkbookView(XSSFWorkbook workbook) {
        try {
            currentWorkbook.close();
            currentWorkbook = workbook;
            numSheets = currentWorkbook.getNumberOfSheets();
            List<GridBase> workbookGrid = WbUtil.mapWorkbookGrid(workbook);
            sheetControls.clear();
            tabPane.getTabs().clear();

            for (int i = 0; i < workbookGrid.size(); i++) {

                Tab tab = new Tab(currentWorkbook.getSheetName(i));
                SpreadsheetController.turnOffDefaultInit = true;

                FxControllerAndView<SpreadsheetController, SpreadsheetView>
sheetControllerView =
                        fxWeaver.load(SpreadsheetController.class);

                sheetControls.add(sheetControllerView);
                int finalI = i;
                sheetControllerView.getView().ifPresent(ss -> {
                    ss.setGrid(workbookGrid.get(finalI));
                });
                tab.setContent(sheetControllerView.getController().ssView);
                this.tabPane.getTabs().add(tab);

            }
            this.rootNode.getChildren().removeIf(node -> node instanceof TabPane);
```

```
            rootNode.getChildren().add(tabPane);

        } catch (IOException ioException) {
            ioException.printStackTrace();
            logger.error("Exception thrown while trying to close the current Workbook. -> " +
ioException.getMessage());

        }

    }


    protected XSSFWorkbook getCurrentWorkbook() {
        return currentWorkbook;
    }

    /**
     * Adds a new blank spreadsheet tab to the workbook view
     * @param rootNode VBox containing the currently visible TabPane.
     */
    protected void addSpreadsheet(VBox rootNode) {
        String tabTitle = defaultTitle.concat(String.valueOf(numSheets + 1));
        Tab tab = new Tab(tabTitle);

        FxControllerAndView<SpreadsheetController, SpreadsheetView> sheetControlView =
                fxWeaver.load(SpreadsheetController.class);
        tab.setContent(sheetControlView.getController().ssView);

        sheetControls.add(sheetControlView);

        numSheets += 1;
        System.out.println("Number of current sheets is " + numSheets);
        sheetControlView.getController().sheetNumber = numSheets;

        this.tabPane.getTabs().add(tab);
        rootNode.getChildren().set(rootNode.getChildren().size() - 1, tabPane);

    }

    void close() throws IOException {
        //Perform close logic
        currentWorkbook.close();
    }
}
@Component
@FxmlView("/fxml/main.fxml")
public class ViewController {

    private final Logger logger = LoggerFactory.getLogger(ViewController.class);
    private final FxWeaver fxWeaver;
```

```
    private Project project;

    private FileService fileService;
    private ProjectService projectService;

    @FXML
    protected Spinner<String> cellTypeSpinner;

    @FXML
    protected Menu recentFilesMenu;


    @Value("${sheet-buddy.about.page}")
    private String aboutPageUri;

    @Value("${sheet-buddy.issues.page}")
    private String issuesPageUri;


    @FXML
    VBox rootNode;

    @FXML
    MenuBar mainMenu;

    private final FxControllerAndView<WorkbookController, TabPane> workbookControlView;

    @Autowired
    public ViewController(FxWeaver fxWeaver,
                          @SuppressWarnings("SpringJavaInjectionPointsAutowiringInspection")
FxControllerAndView<WorkbookController,
                                  TabPane> workbookControlView) {
        this.fxWeaver = fxWeaver;
        this.workbookControlView = workbookControlView;


    }

    @FXML
    public void initialize() {
        /* Initialize the Project */
        project = new Project();
        if (projectService.projectExistsById(project.getProjectId())) {
            project = projectService.getProjectById(project.getProjectId());
        }

        workbookControlView.getController().rootNode = this.rootNode;

        /* Initialize recent files menu */
        recentFilesMenu = MenuUtil.initRecentFileMenu(recentFilesMenu,
project.getRecentFiles());
```

```
        SpinnerValueFactory.ListSpinnerValueFactory<String> spinnerValueFactory =
                new
SpinnerValueFactory.ListSpinnerValueFactory<>(CellFormatUtil.getSupportedCellFormats());
        cellTypeSpinner.setValueFactory(spinnerValueFactory);

    }
    @FXML
    protected void openAboutPage(ActionEvent actionEvent) {
        fxWeaver.getBean(HostServices.class).showDocument(aboutPageUri);
    }

    @FXML
    protected void openWorkBook(ActionEvent event) {
        File wbFile = DialogUtil.showFilePrompt("Choose the workbook to open", ".xlsx",
project.getWorkingDirectoryPath());

        if (Objects.nonNull(wbFile)) {

            logger.info(".xlsx file picked to open -> " + wbFile.getName());

            String directory = wbFile.getParent();
            if (directory != null) {
                project.setWorkingDirectoryPath(directory);
            }

            try {
                XSSFWorkbook workbook = fileService.getWorkbook(wbFile);
                workbookControlView.getController().updateWorkbookView(workbook);

                project.setMostRecentFile(wbFile.getAbsolutePath());
                fileService.setRecentFile(wbFile.getAbsolutePath(), project.getProjectId());

            } catch (Exception e) {
                e.printStackTrace();
                logger.error("Error opening the WorkBook from file: " + wbFile.getName() + "
Exception is " +
                        "instanceOf: " + e.toString());
                DialogUtil.showSimpleAlert("Error opening the WorkBook from file: " +
wbFile.getName(), Alert.AlertType.ERROR);
            }
        }
    }

    @FXML
    protected void exitRequested(ActionEvent actionEvent) {
        try {
            workbookControlView.getController().close();

        } catch (IOException ioe) {
```

```
            ioe.printStackTrace();
        }
        if (!projectService.projectExistsById(project.getProjectId())) {
            projectService.insertProject(project);
        }
        project.persistPreferences();
        fileService.setRecentFile(project.getMostRecentFile(), project.getProjectId());

        try {
            fxWeaver.getBean(Application.class).stop();
        } catch (Exception ex) {
            ex.printStackTrace();
            logger.error("Error exiting when calling JavaFxApplication.stop() method");
            DialogUtil.showAlert("Something went wrong: " + ex.getLocalizedMessage(),
"Background Process " +
                                 "Running Error",
                         "Alert!", Alert.AlertType.ERROR);
        }
    }

    /**
     * @param actionEvent fire event from MenuItem
     */
    @FXML
    protected void openIssuesPage(ActionEvent actionEvent) {
        fxWeaver.getBean(HostServices.class).showDocument(issuesPageUri);
    }

    @FXML
    protected void createNewWorkbook(ActionEvent actionEvent) {
        File f = DialogUtil.showSaveFilePrompt("Create New Workbook", ".xlsx", "",
                StageStyle.UTILITY);
        if (Objects.nonNull(f)) {
            try {
                Workbook wb = WbUtil.createBlankWorkbook(f);
                project.setMostRecentFile(f.getAbsolutePath());
            } catch (Exception exc) {
                exc.printStackTrace();
                DialogUtil.showSimpleAlert("An error has occurred", Alert.AlertType.ERROR);
            }
        }
    }

    @FXML
    protected void closeWorkbook(ActionEvent actionEvent) {
        // TODO: Define

    }

    @FXML
```

```
    protected void saveWorkbook(ActionEvent actionEvent) {
        // TODO: Define
    }

    @FXML
    protected void saveWorkbookAs(ActionEvent actionEvent) {
        // TODO: Define

    }

    @FXML
    protected void openPreferences(ActionEvent actionEvent) {
        FxControllerAndView<SettingsController, AnchorPane> settingsControlView =
                fxWeaver.load(SettingsController.class);
        settingsControlView.getController().show();

    }

    @FXML
    protected void newSpreadsheet(ActionEvent actionEvent) {
        logger.info("adding new ssView");
        workbookControlView.getController().addSpreadsheet(rootNode);

    }

    @Autowired
    public void setProjectService(ProjectService projectService) {
        this.projectService = projectService;
    }

    @Autowired
    public void setFileService(FileService fileService) {
        this.fileService = fileService;
    }
}
@Component
public class WbUtil {

    public static void saveWorkbook(Workbook wb, Project project) throws IOException {

        try (OutputStream fileOut = new FileOutputStream(project.getMostRecentFile())) {
            wb.write(fileOut);
        }
        wb.close();

    }

    public static Workbook createBlankWorkbook(File file) throws Exception {
        Workbook wb = new XSSFWorkbook();
```

```
        try (FileOutputStream fOs = new FileOutputStream(file)) {
            wb.createSheet("New Sheet");
            wb.write(fOs);
            DialogUtil.showInfoAlert("Successful Operation", "You created a new workbook with
one spreadsheet", "New Sheet", false);
            return wb;
        } catch (Exception ex) {
            DialogUtil.showSimpleAlert("Error during writing your new file",
Alert.AlertType.ERROR);
            ex.printStackTrace();
        }
        throw new Exception("Failed to create new Workbook");
    }

    public static void validateSheetName(String name) throws IllegalArgumentException {
        try {
            WorkbookUtil.validateSheetName(name);
        } catch (IllegalArgumentException e) {
            e.printStackTrace();
            DialogUtil.showInfoAlert(e.getMessage(), true);
        }
    }

    public static String createSafeSheetName(String name) {
        return WorkbookUtil.createSafeSheetName(name);
    }

    public static List<GridBase> mapWorkbookGrid(XSSFWorkbook workbook) {
        List<GridBase> sheets = new LinkedList<>();
        for (int i = 0; i < workbook.getNumberOfSheets(); i++) {
            XSSFSheet sheet = workbook.getSheetAt(i);
            FormulaEvaluator evaluator =
workbook.getCreationHelper().createFormulaEvaluator();

            GridBase grid = SsUtil.mapSheetToGrid(sheet, evaluator);
            sheets.add(grid);
        }
        return sheets;
    }
}
```