МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

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

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

КВАЛІФІКАЦІЙНАРОБОТА

на здобуття ступеня вищої освіти «магістр»

на тему: Математична модель для перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java, та створення програми для її реалізації

Виконала: студентка 6151мз групи
Смикодуб Т.Г.
(підпис, ПІБ)

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

Миколаїв – 2021 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

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

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

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

Студенту _____
(Прізвище, ім'я, по батькові)

1. Тема роботи: Математична модель для перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java, та створення програми для її реалізації

Керівник роботи Приходько С.Б.

Затверджені наказом ректора № 1239уч від «13» 10 2021 р.

2. Термін подання роботи: 06.12.2021 р.

3. Вихідні дані по роботі: _____

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

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

5. Перелік презентаційних матеріалів _____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
	Трушлякова А.Б.		
	Гурець Н.В.		
	Гурець Н.В.		

7. Дата видачі завдання 13.10.2021 р.

К А Л Е Н Д А Р Н И Й П Л А Н

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

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

Студент _____ _____
(підпис) (ПІБ)

Керівник роботи _____ _____
(підпис) (ПІБ)

РЕФЕРАТ

Смикодуб Тетяна Георгіївна

«Математична модель для перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java, та створення програми для її реалізації»

Кваліфікаційна робота на здобуття ступеня вищої освіти магістр зі спеціальності 121 – «Інженерія програмного забезпечення» (ОП «Інженерія програмного забезпечення»). Національний університет кораблебудування імені адмірала Макарова. Миколаїв, 2021 р.

Обсяг роботи: 123 стор., 17 табл., 19 рис., 52 використаних джерел, 5 додатків.

Актуальність теми роботи: За статистикою біля 78% компаній широко використовують програмне забезпечення з відкритим вихідним кодом. Згідно звіту Coverity Scan-Open Source Report 2017, аналіз ПЗ дозволив виявити аномалії та дефекти у багатьох найважливіших проектах із відкритим кодом. Тому удосконалення існуючих моделей для перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java, є актуальним задачею та представляє науково-практичний інтерес.

Мета та завдання дослідження. Метою є підвищення достовірності визначення аномалій у взаємозв’язках між частинами застосунків, що розроблені мовою Java, з точки зору об’єктно-орієнтовного проектування за рахунок побудови трансформованого еліпсу передбачення. **Завдання дослідження:** провести аналіз існуючих засобів та моделей перевірки взаємозв'язків Java-застосунків; удосконалити математичну модель для перевірки взаємозв'язків Java-застосунків за рахунок побудови рівняння трансформованого еліпсу передбачення для нормалізованих даних; розробити ПЗ для перевірки взаємозв'язків застосунків, реалізованих мовою Java.

Об’єктом дослідження є процес перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java.

Предмет дослідження: математичні моделі для перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java.

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

Наукова новизна одержаних результатів: полягає в удосконаленні існуючої математичної моделі для перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java, за рахунок використання трансформованого еліпсу передбачення, побудованого на основі двовимірного нормалізуючого перетворення Джонсона сім’ї S_b. Це дозволяє підвищити достовірність визначення аномалій у взаємозв’язках між частинами застосунків.

Практичне значення отриманих результатів. Розроблене ПЗ для перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java, дозволяє визначити аномалії у взаємозв'язках між частинами застосунків, забезпечує зберігання результатів визначення, а також надає користувачу швидкий доступ до попередніх результатів.

Апробація результатів досліджень: результати досліджень пройшли апробацію на ІІ Всеукраїнської науково-практичної Інтернет конференції «Інформаційні технології: моделі, алгоритми, системи» (м. Миколаїв, 26 – 28 жовтня 2021 р.).

Публікації: результати роботи викладено у 1 науковій праці – тезах конференції.

Ключові слова: JAVA, ЗАСТОСУНОК, МЕТРИКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ТРАНСФОРМОВАНИЙ ЕЛІПС ПЕРЕДБАЧЕННЯ, НОРМАЛІЗУЮЧЕ ПЕРЕТВОРЕННЯ, ВИКИД.

ABSTRACT

Smykodub Tetiana Georgiyivna

«A mathematical model for checking interconnections between parts of the applications developed in Java and creating the software for its implementation»

The qualification work for the degree of higher education Master's degree in specialty 121 - "Software Engineering" (EP "Software Engineering"). Admiral Makarov National University of Shipbuilding. Mykolaiv, 2021

The qualification work is presented on the 123 pages of typewritten text, contains 17 tables, 19 figures, 5 appendices and 52 references.

Relevance of the topic of the work. According to statistics, about 78% of companies widely use Open Source Software. According to the Coverity Scan-Open Source Report 2017, the software analysis revealed anomalies and defects in many of the most important open source projects. Therefore, the improvement of existing models for checking the relationships between parts of the applications developed in Java is an urgent task and is of scientific and practical interest.

The purpose and objectives of the study. The aim is to increase the reliability of the definition of anomalies in the relationships between the parts of applications developed in Java, in terms of object-oriented design by building a transformed prediction ellipse. **Objectives of the study**: to analyze the existing tools and models for verifying the relationship of Java-applications; to improve the mathematical model for checking the relationships of Java-applications by constructing the equation of the transformed prediction ellipse for normalized data; develop software for checking the relationships of applications developed in Java.

The object of the study is the process of checking the relationships between parts of applications developed in Java.

The subject of the study is mathematical models for checking interconnections between parts of the applications developed in Java

Research methods. Methods of probability theory, mathematical statistics, and regression analysis were used to solve the tasks.

The scientific novelty of the obtained results is to improve the existing mathematical model for checking the relationships between parts of applications developed in Java, using a transformed prediction ellipse based on the bivariate normalizing Johnson transformation of the S_b family. This increases the reliability of detecting anomalies in the relationships between parts of the application.

The practical significance of the results obtained. The developed software for checking the relationships between parts of applications developed in Java, allows to identify anomalies in the relationships between parts of the application, provides storage of the results of the definition, and provides the user with quick access to previous results.

Approbation of research results: research results were tested at the II All-Ukrainian scientific-practical Internet conference "Information technologies: models, algorithms, systems" (Mykolaiv, October 26 - 28, 2021).

Publications: results of the work are presented in 1 scientific paper – conference abstracts.

Keywords: JAVA, APPLICATION, SOFTWARE METRIC, TRANSFORMED PREDICTION ELLIPSE, NORMALIZING TRANSFORMATION, OUTLIER.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ВВ	Випадкова величина
ЕД	Емпіричні дані
ПЗ	Програмне забезпечення
СК	Метрики Чідамбера та Кемерера
GUI	Graphical User Interface
OSS	Open Source Software

ВСТУП

Актуальність теми. Згідно стандарту ISO/IEC/IEEE 12207:2017 [1] вимірювання метрик програмного забезпечення (ПЗ) є найважливішою частиною інженерії вимірювання ПЗ. Сьогодні 78% компаній широко використовують програмне забезпечення з відкритим вихідним кодом (OSS), а компоненти з відкритим кодом містяться в більш ніж половині всього запатентованого програмного забезпечення [2]. Згідно звіту Coverity Scan (Coverity Scan-Open Source Report 2017) [3], аналіз ПЗ дозволив розробникам виправити понад 600000 дефектів у деяких із найважливіших проектів із відкритим кодом. За останній рік мова програмування Java зберігає позиції лідерства згідно рейтингу TIOBE [4]. Також Java знаходиться серед трійки найбільш популярних мов програмування, з якими працюють користувачі платформи GitHub. Тому удосконалення існуючих моделей для перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java, є актуальним завданням.

На теперішній час існує велика кількість досліджень та наукових праць [5], які присвячені аналізу метрик програмного забезпечення. Слід зазначити, що більшість існуючих моделей для перевірки зв’язності між частинами застосунків не враховують кореляцію між метриками ПЗ [6]. За критеріями Мардіа було встановлено, що емпіричні дані метрик CBO та RFC, які відображають відношення між частинами застосунку, мають негаусівських закон розподілу [7]. Тому побудова моделі на основі рівняння еліпсу передбачення виявилася недоцільною без нормалізації даних [8]. Для нормалізації двовимірних даних пропонується застосування двовимірного перетворення Джонсона [9-12].

Проекти з відкритим вихідним кодом різноманітні за розміром, кількістю компонентів та тривалістю розробки. Тому підвищення достовірності визначення аномалій у взаємозв’язках між частинами Java-

застосунків за рахунок побудови моделі на основі рівняння трансформованого еліпсу передбачення, а також розробка програми для її реалізації є актуальним та має практичну цінність.

Мета та завдання дослідження. Метою роботи є підвищення достовірності визначення аномалій у взаємозв’язках між частинами застосунків, що розроблені мовою Java, з точки зору об’єктно-орієнтовного проектування за рахунок побудови трансформованого еліпсу передбачення.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати існуючі моделі перевірки взаємозв'язків Java-застосунків, порівняти їх;
- обґрунтувати необхідність удосконалення математичної моделі для перевірки взаємозв'язків Java-застосунків;
- дослідити різні джерела з відкритим вихідним кодом, реалізовані мовою Java, які можуть бути використані для побудови моделі;
- отримати необхідні метрики з кожного проекту;
- нормалізувати отримані емпіричні дані, використовуючи двовимірне нормалізуюче перетворення Джонсона [13];
- перевірити вихідні емпіричні дані на викиди;
- побудувати рівняння трансформованого еліпсу передбачення для нормалізованих даних;
- удосконалити математичну модель для перевірки взаємозв'язків Java-застосунків [14];
- розробити програму для перевірки взаємозв'язків застосунків, реалізованих мовою Java.

Об’єктом дослідження є процес перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java.

Предмет дослідження: математичні моделі для перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java.

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

Методи теорії ймовірностей, математичної статистики та регресійного аналізу використовувалися для побудови моделі на основі рівняння трансформованого еліпсу передбачення для перевірки взаємозв'язків між частинами Java-застосунків.

Наукова новизна одержаних результатів полягає в удосконаленні існуючої математичної моделі для перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java, за рахунок використання трансформованого еліпсу передбачення, побудованого на основі двовимірного нормалізуючого перетворення Джонсона сім'ї S_b. Це дозволяє підвищити достовірність визначення аномалій у взаємозв'язках між частинами застосунків.

Практичне значення отриманих результатів. Розроблене ПЗ для перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java, дозволяє визначити аномалії у взаємозв'язках між частинами застосунків, забезпечує зберігання результатів визначення, а також надає користувачу швидкий доступ до попередніх результатів.

Особистий внесок здобувача. Кваліфікаційна робота є самостійно виконаною працею. Усі результати, викладені у роботі, отримані автором особисто. У праці, яка опублікована у співавторстві з керівником кваліфікаційної роботи [14], здобувачеві належить: удосконалення математичної моделі для перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java.

Апробація результатів досліджень: результати досліджень пройшли апробацію на II Всеукраїнської науково-практичної Інтернет конференції «Інформаційні технології: моделі, алгоритми, системи» (м. Миколаїв, 26 – 28 жовтня 2021 р.).

Публікації. Основні результати кваліфікаційної роботи викладено у 1 науковій праці – тезах конференції.

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ТА МОДЕЛЕЙ ДЛЯ ПЕРЕВІРКИ ВЗАЄМОЗВ'ЯЗКІВ МІЖ ЧАСТИНАМИ ЗАСТОСУНКІВ, ЩО РОЗРОБЛЕНІ МОВОЮ JAVA

1.1 Java-застосунки з відкритим вихідним кодом

Станом на листопад 2021 мова програмування Java зберігає позиції лідерства згідно рейтингу TIOBE з 10,72% використання [4]. Також Java знаходиться серед трійки найбільш популярних мов програмування, з якими працюють користувачі платформи GitHub (платформи для проектів з відкритим вихідним кодом (Open Source Software)).

Програмне забезпечення з відкритим кодом (OSS) швидко набирає популярність у корпоративному світі як практична альтернатива дорогому приватному програмному забезпеченню. Сьогодні 78% компаній використовують OSS, а компоненти з відкритим кодом містяться в більш ніж половині всього запатентованого програмного забезпечення [2]. Програмне забезпечення з OSS надає багато послуг і продуктів для компаній, промисловості, урядів та освітніх організацій, включаючи The White House, GM, CNN, Stanford HC, BestBuy та Umm Al-Qura University. Обґрунтування просте: OSS знижує витрати на розробку, скорочує час виходу на ринок, підвищує продуктивність розробників і прискорює інновації.

Open Source Software Development – це свого роду розподілена база розробки програмного забезпечення з використанням техніки експертної оцінки, а команда розробників розподілена по всьому світу в різних часових поясах. Існує багато успішних OSS, таких як Apache, PHP, Nginx, MySQL і MariaDB. Успіх цих проектів пояснюється багатьма ключовими факторами успіху, наведемо деякі з них: розробник є фактичним користувачем, модульна архітектура [15], існування спільнот, які підтримують розробку системи [16] тощо.

Отже, до переваг Open Source застосунків можна віднести такі [17]:

– Доступність. Абсолютна більшість Open Source застосунків повністю безкоштовні для будь-якого кінцевого користувача, у тому числі для комерційних компаній.

– Гнучкість. Відкритий код дозволяє користувачам самостійно вносити будь-які зміни та доопрацьовувати програми для своїх конкретних цілей та потреб. Також більшість такого ПЗ є кросплатформним – сумісним з різними пристроями та сімействами ОС.

– Якість та безпека. Open Source застосунки регулярно оновлюються і допрацьовуються, причому необмеженою групою програмістів, а масштабною спільнотою користувачів з усього світу. Відкритість їх коду дозволяє всім охочим знаходити та виправляти помилки та вразливості в OSS, що робить його якіснішим, надійнішим та безпечнішим багатьох пропрієтарних аналогів.

– Масштабованість. Ще однією важливою перевагою застосунків з відкритим кодом є його можливість легко масштабуватися під запити компанії-клієнта, а також реєструвати в ньому необмежену кількість користувачів.

– Відкритість. При виборі OSS користувачі не прив'язуються до конкретного розробника та не залежать від його послуг або оновлень. За бажанням вони в будь-який момент можуть вибрати інший подібний продукт із аналогічним вихідним кодом.

Однак існують і недоліки такого підходу при створенні ПЗ: згідно звіту Coverity Scan аналіз ПЗ дозволив розробникам виправити понад 600000 дефектів у деяких із найважливіших проектів із відкритим кодом [3]. Проблема якості коду може бути особливо важливою для великих організацій, які розповсюджують продукти OSS серед своїх клієнтів. Таким організаціям необхідно інвестувати в процедури та системи для управління використанням OSS та включати передовий досвід розробки програмного забезпечення у життєвий цикл OSS. Дослідження показують, що

забезпечення якості програмного забезпечення зазвичай перевищує 30% загальної вартості проекту [18]. Також потрібні засоби для виправлення виявлених проблем та забезпечення швидкого та широкого розгортання виправлень. Отже удосконалення існуючих моделей, що аналізують взаємозв'язки між частинами застосунків, є досить актуальним.

На теперішній час існують дослідження та наукові праці, в яких виявляються та аналізуються причини неуспіху деяких проектів з відкритим вихідним кодом [19]. Існують деякі характеристики проектів OSS, які сприяють невдалому результату: відсутність формальних засобів, тобто відсутність планування проекту, недотримання обєктно-орієнтовного підходу та поганий архітектурний дизайн.

1.2 Метрики програмного забезпечення

Метрики ПЗ кількісно визначають різні властивості програмних застосунків та програмних процесів у вигляді чисельного відображення. Мета полягає у виведенні одного або декількох значень особливостей ПЗ, що дає змогу порівнювати ці значення з подібними проектами, зі специфічними стандартами, які притаманні даній компанії. З отриманих результатів можна прийти до висновку щодо якості ПЗ та всього програмного процесу, а також, якщо необхідно, подальших заходів.

Часто метрики ПЗ порівнюють з методами та моделями для оцінювання витрат і зусиль, такими як Function Point, COCOMO або метод Delphi.

В стандарті IEEE 1061 термін «метрика програмного забезпечення» визначається як показник якості ПЗ, і тлумачиться таким чином: метрика якості програмного забезпечення - це функція, яка відображає програмний блок в числовому значенні [20].

Метрики програмного продукту враховують його характеристики. Розрізняють динамічні і статичні метрики для зручності використання.

Динамічні метрики ПЗ використовуються для оцінювання продуктивності та надійності ПЗ. Прикладами є необхідний час виконання або кількість помилок, що виникли під час виконання.

Статичні метрики ПЗ включають показники дизайну, самої програми або документації. Вони також враховують такі аспекти, як складність і простота обслуговування.

Відмінності між процедурною і об'єктно-орієнтованою розробкою програмного забезпечення враховуються шляхом подальшої диференціації в звичайні та об'єктно-орієнтовані метрики продукту [22]. Об’єктно-орієнтовані метрики повинні мати можливість зосередитися на визначенні методів та даних як інтегрованого об’єкта [23]. Значення об’єктно-орієнтованих мір слід використовувати для оцінювання таких критеріїв:

– Ефективність – чи ефективно розроблені програмні модулі?

– Складність – чи можна було б ефективніше використовувати програмні модулі для зменшення архітектурної складності?

– Зрозумілість – чи збільшує дизайн психологічну складність?

– Можливість повторного використання. Чи підтримує якість дизайну можливе повторне використання?

– Тестованість/придатність до обслуговування – чи підтримує програмна одиниця модульне тестування?

1.3 Існуючі моделі для перевірки взаємозв'язків між частинами застосунків

Використання об’єктно-орієнтованого підходу не гарантує розробку високоякісного програмного забезпечення, а також не дозволяє уникнути помилок, що допускають програмісти як на етапі розробки, так і на етапі обслуговування.

Тому в літературі пропонуються різні об’єктно-орієнтовані метрики як засіб визначення того, чи відповідає досліджуване програмне забезпечення бажаними властивостями об’єктно-орієнтованого проектування, відповідність яким дозволить покращити якість програмного забезпечення. Найбільш відомі були запропоновані Лі та Генрі [24], Бріанд та ін. [25], Чідамбер та Кемерер (далі СК) [26] та Абреу та ін. [27].

Лі і Генрі проаналізували дві комерційні системи, які називаються UIMS і QUES. Вони виявили, що показники СК пояснюють додаткову дисперсію в зусиллях по технічному обслуговуванню, крім тієї, яку пояснюють традиційні показники розміру [28]. Чідамбер і Кемерер також проаналізували емпіричні дані від двох комерційних організацій та запропонували модель, за допомогою якої показники можуть бути використані для управління зусиллями з розробки ОО [26]. R. Subramanyam та M.S. Krishnan [29] стверджують, що використання СК набору може допомогти користувачам зрозуміти складність дизайну, виявити недоліки в проекті та передбачити певні результати проекту та зовнішні якості програмного забезпечення, такі як дефекти програмного забезпечення, тестування та зусилля з обслуговування.

Отже, застосування об’єктно-орієнтованих метрик дозволить вдосконалити:

- визначення якості існуючого ПЗ або його розробки / експлуатації;
- прогнозування якості ПЗ або його розробки / експлуатації у майбутньому;
- підвищення якості ПЗ;
- оцінювання вартості майбутніх проектів;
- оцінювання продуктивності застосування нових засобів і методів;
- прогноз майбутніх потреб у кадрах;
- передбачення і скорочення майбутніх потреб у технічному обслуговуванні.

визначення якості існуючого програмного забезпечення або його розробки/функціонування;

Метрики СК можуть застосовуватися на різних етапах проектування та впровадження (кодування) життєвого циклу розробки програмного забезпечення, наприклад, їх можна застосувати до діаграми класів (під час пізнішої фази проектування) [30] і до вихідного коду.

Надамо визначення та теоретичні основи метрик СК [26, 31]:

1) Зв’язаність між об’єктами (CBO, Coupling Between Objects): це кількість інших класів, до яких він пов’язаний. CBO відноситься до поняття, що об'єкт пов'язаний з іншим об'єктом, якщо один з них діє на інший, тобто методи одного використовують методи або змінні екземпляра іншого. Оскільки об’єкти одного класу мають однакові властивості, два класи з’єднуються, коли методи, оголошені в одному класі, використовують методи або змінні екземпляра, визначені іншим класом. Надмірне з’єднання між класами об’єктів шкодить модульному дизайну та запобігає повторному використанню. Чим незалежніший клас, тим легше його повторно використовувати в іншій програмі. Щоб покращити модульність та сприяти інкапсуляції, міжоб’єктні класи пар мають бути зведені до мінімуму. Чим більше число пар, тим вище чутливість до змін інших частин конструкції, а отже, обслуговування складніше. Вимірювання зв'язку корисно, щоб визначити, наскільки складним може бути тестування різних частин проекту. Чим вище зв’язок між об’єктними класами, тим більш складним має бути тестування

2) Відгук класу (RFC, the Response for Class): це кількість методів класу та кількість методів, викликаних цими методами класу. Він вимірює ступінь зв'язку між класами системи. Визначення метрики RFC (1.1):

$$RFC = | RS |, \tag{1.1}$$

де RS – набір відповідей для класу.

Набір відповідей для класу можна виразити так:

$$RS = \{ M \} \cup_{all\, i} \{ R_i \}, \quad (1.2)$$

де $\{ R_i \}$ – набір методів, викликаних методом *i*, $\{ M \}$ – набір усіх методів у класі.

RFC – це набір методів, які потенційно можуть бути виконані у відповідь на повідомлення, отримане об’єктом цього класу. Мощність цього набору є мірою атрибутів об'єктів у класі. Оскільки набір включає методи, викликані поза класом, він також є мірою потенційного зв’язку між класом та іншими класами. Якщо у відповідь на повідомлення можна викликати велику кількість методів, тестування та налагодження класу ускладнюються, оскільки це вимагає більшого рівня розуміння з боку тестувальника. Чим більша кількість методів, які можна викликати з класу, тим більша складність класу. Знання значень міри допоможе правильно розподілити час тестування.

3) Відсутність згуртованості в методах (LCOM, Lack of Cohesion in Methods): це кількість пар методів, які не мають спільної змінної екземпляра (подібність яких дорівнює 0), мінус кількість пар методів, які мають спільні змінні екземпляра класу (подібність яких не дорівнює нулю).

Чим більше подібних методів, тим більш згуртований клас, що узгоджується з традиційними уявленнями про згуртованість, що вимірюють взаємозв’язок між частинами проекту. Якщо жоден із методів класу не відображає поведінку екземпляра, тобто не використовує жодних змінних екземпляра, вони не мають подібності, і значення LCOM для класу буде нульовим. Значення LCOM визначає відносну різнорідність методів у класі. Менша кількість пар, що не перетинаються, означає більшу схожість методів. LCOM тісно пов'язаний зі змінними екземпляра і методами класу, і тому є мірою атрибутів класу. Згуртованість методів у класі є бажаною, оскільки вона сприяє інкапсуляції. Відсутність згуртованості означає, що

класи слід розділити на більше підкласів. Низька згуртованість збільшує складність, тим самим збільшуючи ймовірність помилок під час процесу розробки.

4) Вага методів на клас (WMC, Weighted Methods Per Class): міра представляє суму складності всіх методів класу. Якщо складності всіх методів є одиницями, то WMC – це кількість методів класу. Кількість методів і складність задіяних методів є показником того, скільки часу та зусиль потрібно для розробки та підтримки класу. Чим більша кількість методів у класі, тим більший потенційний вплив на нащадків, оскільки нащадки успадкують усі методи, визначені в класі. Класи з великою кількістю методів, ймовірно, будуть більш специфічними для застосування, що обмежує можливість повторного використання.

5) Глибина дерева наслідування (DIT, Depth of Inheritance Tree): вимірює номер рівня для класу в дереві успадкування. DIT кореневого класу в дереві успадкування дорівнює нулю. DIT є мірою того, скільки класів-предків потенційно можуть вплинути на цей клас. точки зору. Чим глибше клас знаходиться в ієрархії, тим більшу кількість методів він, ймовірно, успадкує, що ускладнює прогнозування його поведінки. Більш глибокі дерева формують більшу складність проектування, оскільки задіяно більше методів та класів. Чим глибше певний клас знаходиться в ієрархії, тим більший потенціал повторного використання успадкованих методів.

6) Кількість нащадків (NOC, Number of Children): вимірює кількість прямих підкласів класу (нащадків). Якщо підкласи залежать від свого суперкласу (методів або змінних екземплярів), будь-які зміни в суперкласі можуть вплинути на підкласи, а отже, тим важче підтримувати суперклас. NOC відноситься до поняття обсягу властивостей. Це міра того, скільки підкласів успадкують методи батьківського класу. Чим більше нащадків, тим більше повторне використання, оскільки успадкування є формою повторного використання. Чим більше нащадків, тим більша ймовірність хибного абстрагування батьківського класу. Якщо NOC класу має велике значення, це

може бути випадком неправильного використання підкласу. NOC дає уявлення про потенційний вплив класу на дизайн. Якщо клас має велику кількість нащадків, може знадобитися більше часу на тестування методів цього класу.

Для перевірки взаємозв’язків між частинами ПЗ або відносин між класами ПЗ в [29] запропоновано використовувати разом дві метрики: CBO та RFC. Але на практиці, як правило, ці метрики аналізуються окремо без їх кореляції, а прийнятні їх значення визначаються лише для класу. Так для класу, значення CBO між 1 та 5 вважається добрим, а рекомендоване значення RFC складає від 0 до 50. Тому виникає потреба у математичній моделі, яка би дозволяла визначати припустимі значення CBO та RFC з урахуванням їх кореляції для застосунку в цілому [14].

1.4 Способи удосконалення математичної моделі для перевірки взаємозв'язків між частинами застосунків

Вихідні дані для перевірки взаємозв'язків між частинами ПЗ, як правило, не мають нормального розподілу, що в результаті призводить до хибних результатів у розрахунках або негативно впливає на достовірність отриманих результатів. Щоб уникнути цієї проблеми, перед тим, як будувати математичну модель, потрібно нормалізувати вихідні дані.

Для нормалізації негаусівських даних можуть бути використані перетворення на основі десяткового або натурального логарифму, перетворення Box-Cox, перетворення Джонсона та інші. У даній роботі в якості нормалізуючого перетворення буде використовуватись одномірне та двовимірне чотирьохпараметричне перетворення Джонсона, та проаналізовані отримані результати. Гіпотеза: двовимірне чотирьохпараметричне перетворення Джонсона надає кращі результати у

порівнянні з іншими перетвореннями (центрування за середнім значенням та одномірне перетворення Джонсона).

Сьогодні в основі статистичних методів аналізу двовимірних даних лежить еліпс [32, 33]. Проте відомі статистичні методи (наприклад, виявлення викидів, що базуються на еліпсі передбачення) використовуються при припущенні, що дані мають розподіл Гаусу. Але це припущення справедливе тільки в поодиноких випадках. Використання нормалізуючих перетворень дозволяє перейти до побудови рівняння еліпсу передбачення для нормалізованих даних, потім шляхом застосування відповідного зворотнього перетворення перейти до побудови трансформованого еліпсу передбачення [12].

В загальному вигляді математична модель може бути представлена рівнянням еліпсу передбачення за формулою (1.3):

$$(Z-\overline{m}_Z)^T S^{-1}(Z-\overline{m}_Z)=\frac{2\left(N^2-1\right)}{N(N-2)}F_{2,N-2,\alpha}, \qquad (1.3)$$

де $F_{2,N-2,\alpha}$ - квантіль F розподілу;

$\overline{m}_Z$ - вектор середніх;

$\overline{\mathrm{m}}_{\mathrm{Z}}=\left(\overline{m}_{z_1},\overline{m}_{z_2}\right)^T$;

α - ймовірність;

S - матриця коваріації.

Матриця коваріації представлена (1.4):

$$S=\begin{pmatrix} S_{Z_1}^2 & S_{Z_2Z_1} \\ S_{Z_1Z_2} & S_{Z_1}^2 \end{pmatrix}. \qquad (1.4)$$

Ліва частина рівняння (1.3) має вигляд:

$$\frac{S_{Z_1}^2 S_{Z_2}^2}{S_{Z_1}^2 S_{Z_2}^2 - S_{Z_1 Z_2}^2}\left[\frac{\left(Z_1-\overline{m}_{Z_1}\right)^2}{S_{Z_1}^2}+\frac{\left(Z_2-\overline{m}_{Z_2}\right)^2}{S_{Z_2}^2}-\frac{2S_{Z_1Z_2}\left(Z_1-\overline{m}_{Z_1}\right)\left(Z_2-\overline{m}_{Z_2}\right)}{S_{Z_1}^2 S_{Z_2}^2}\right]= \\ =\frac{2\left(N^2-1\right)}{N\left(N-2\right)}F_{2,N-2,\alpha} \quad . \qquad (1.5)$$

Рівняння (1.5) перегрупуємо у більш зручний для геометричного моделювання вигляд:

$$\frac{1}{S_{Z_1}^2}\left(Z_1-\overline{m}_{Z_1}\right)^2-\frac{2S_{Z_1Z_2}\left(Z_2-\overline{m}_{Z_2}\right)}{S_{Z_1}^2 S_{Z_2}^2}\left(Z_1-\overline{m}_{Z_1}\right)+\frac{\left(Z_2-\overline{m}_{Z_2}\right)^2}{S_{Z_2}^2}= \\ -\frac{2\left(N^2-1\right)}{N\left(N-2\right)}F_{2,N-2,\alpha}\frac{S_{Z_1}^2 S_{Z_2}^2 - S_{Z_1Z_2}^2}{S_{Z_1}^2 S_{Z_2}^2}=0 \quad . \qquad (1.6)$$

Однак при побудові рівняння еліпсу передбачення виникає ряд труднощів [10]:

– якщо випадкові величини, що входять до математичної моделі, не підпорядковуються нормальному закону розподілу, це призводить до неможливості побудувати еліпс передбачення;

– якщо випадкова величина корелюється одночасно з іншими випадковими величинами, що зазвичай і буває при вирішенні практичних завдань, отримуємо недостовірні результати.

Гіпотеза: застосування двовимірного перетворення Джонсона дозволить усунути вищенаведені проблеми.

1.5 Перевірка вихідних емпіричних даних на викиди

Для отримання значень викидів у вихідних даних використаємо квадрат відстані Махаланобіса. Квадрат відстані Махаланобіса MSD для двовимірних випадкових даних визначається за формулою (1.7):

$$d_i^2 = (\mathrm{Z}_i - \overline{m}_Z)^T S^{-1} (\mathrm{Z}_i - \overline{m}_Z), \tag{1.7}$$

де $i = 1,2,\ldots,N$;

$\overline{m}_Z$ - вектор середніх;

S - матриця коваріації

Матриця коваріації визначається за формулою (1.8):

$$S = \frac{1}{N} \sum_{i=1}^{N} (\mathrm{Z}_i - \overline{m}_Z)(\mathrm{Z}_i - \overline{m}_Z)^T. \tag{1.8}$$

Після визначення d_i^2 треба використати або критерій Пірсона χ^2, або критерій Фішера *F*. При використанні критерію Фішера *F* треба визначити тестову статистику T_S (Test statistic) [32]. T_S для d_i^2 обчислюється за формулою (1.9) :

$$Ts = \frac{(N-m)Nd_i^2}{(N^2-1)m}. \tag{1.9}$$

T_S має апроксимований розподіл *F* з m та $N - m$ ступенями свободи.

У даній роботі будемо використовувати критерій Фішера *F*. Тестова статистика для квадрату відстані Махаланобіса порівнюється з квантилем розподілу *F*, який позначається як $F_{m,N-m,\alpha}$. Тут α – це рівень значущості, який у нашому випадку дорівнюватиме 0,005. Точки, для яких значення T_S, розраховане за формулою (1.9), буде більше, ніж квантиль розподілу *F* вважаються викидами, і значення максимального видаляється з набору даних.

Після усунення викиду отриманий новий набір значень знову нормалізується та для нього знову знаходиться квадрат відстані Махаланобіса [10] за формулою (1.8) та тестова статистика за формулою (1.9). Процедура повторюється до тих пір, поки всі значення T_S не будуть менше або дорівнюватимуть квантилю розподілу *F*.

2 УДОСКОНАЛЕННЯ МАТЕМАТИЧНОЇ МОДЕЛІ ДЛЯ ПЕРЕВІРКИ ВЗАЄМОЗВ'ЯЗКІВ МІЖ ЧАСТИНАМИ ЗАСТОСУНКІВ, ЩО РОЗРОБЛЕНІ МОВОЮ JAVA

2.1 Розробка удосконаленої математичної моделі для перевірки взаємозв'язків між частинами застосунків із застосуванням нормалізуючих перетворень

Для моделювання ВВ у двомірному просторі даних на сьогоднішній день застосовують рівняння еліпсу передбачення [33], але його застосування обмежене двомірним нормальним розподілом. У випадку негаусівської ВВ побудова рівняння еліпсу передбачення без припущення про нормальність ВВ може бути ускладненою або неможливою. Також використання такого припущення може істотно спотворити результати. Тому перед тим, як будувати рівняння еліпсу передбачення, потрібно ВВ перевірити на нормальність за допомогою критерію Мардіа (Mardia's test), якій базується на багатовимірних асиметрії та ексцесі [34], а потім нормалізувати вихідні ЕД.

Отже, двовимірний розподіл даних метрик RFC та CBO для Java-застосунків не є гаусівським. Так, було перевірені двовимірні дані метрик RFC та CBO з 46 Java-застосунків з відкритим кодом, які розміщені на сайті GitHub (https://github.com) на нормальність за допомогою критерію Мардіа. Тому використаємо двовимірне перетворення Джонсона для нормалізації негаусівських даних.

В загальному випадку двовимірне нормалізуюче перетворення негаусівського вектора випадкових даних до гаусівського має вигляд (2.1):

$$Z = \psi(X). \tag{2.1}$$

У роботі [13] наводиться багатовимірне нормалізуюче перетворення Джонсона (2.2):

$$Z = \gamma + \eta h\left[\lambda^{-1}(X - \varphi)\right] \sim N_m(0_m, S), \tag{2.2}$$

де γ, η, φ та λ параметри перетворення Джонсона, що визначаються за (2.3).

$$\begin{gathered}\gamma = (\gamma_1, \gamma_2)^T;\ \eta = diag(\eta_1, \eta_2);\ \varphi = (\varphi_1, \varphi_2)^T; \\ \lambda = diag(\lambda_1, \lambda_2);\ h[(y_1, y_2)] = \{h_1(y_1), h_2(y_2)\}^T,\end{gathered} \tag{2.3}$$

де $h_i(\cdot)$ - це одна з функцій перетворення сім'ї розподілів Джонсона (2.4)-(2.6)

$$h_1(x, \varphi, \lambda) = \ln(\tilde{x}), x > \varphi; \tag{2.4}$$

$$h_2(x, \varphi, \lambda) = \ln\left(\frac{\tilde{x}}{1 - \tilde{x}}\right), \varphi < x < \varphi + \lambda; \tag{2.5}$$

$$h_3(x, \varphi, \lambda) = Arsh(\tilde{x}), -\infty \le x \le +\infty. \tag{2.6}$$

Сім'ї функцій h_1 відповідає логарифмічно нормальний розподіл S_L Джонсона, сім'ї функцій h_2 відповідає сім'я розподілів S_B Джонсона, сім'ї функцій h_3 відповідає сім'я розподілів S_U Джонсона, $\tilde{x} = \frac{x - \varphi}{\lambda}$.

Перетворення (2.1) має зворотне перетворення (2.7):

$$x = \varphi + \lambda h^{-1}(z, \gamma, \eta);\ \eta > 0; -\infty < \gamma < \infty; \lambda > 0; -\infty < \varphi < \infty, \tag{2.7}$$

де x – випадкова величина з розподілом Джонсона;

γ, η, φ, λ – параметри перетворення Джонсона;

z – нормально розподілена випадкова величина;

h^{-1} – функції певної сім'ї (2.8) –(2.10):

$$h_1^{-1}(z,\gamma,\eta) = e^{\zeta}; \tag{2.8}$$

$$h_2^{-1}(z,\gamma,\eta) = \frac{1}{1+e^{-\zeta}}; \tag{2.9}$$

$$h_3^{-1}(z,\gamma,\eta) = \frac{e^{\zeta}-e^{-\zeta}}{2}. \tag{2.10}$$

Функція h_1^{-1} – для сім'ї S_L Джонсона, функція h_2^{-1} – для сім'ї S_B Джонсона, функція h_3^{-1} – для сім'ї S_U Джонсона, $\zeta = \frac{z-\gamma}{\eta}$.

При виборі конкретної сім'ї розподілу Джонсона користуються виходячи із значень квадрата асиметрії A^2 і ексцесу $\varepsilon(A^2)$ вихідної вибірки [35]. Для автоматизації розрахунків та при розробці ПЗ зручніше використовувати аналітичну залежність (2.11), наведену в [36]:

$$\begin{aligned}\varepsilon(A^2) = {} & 3{,}59\cdot 10^{-6}A^8 - 4{,}8805\cdot 10^{-4}A^6 + 4{,}1655\cdot 10^{-2}A^4 + \\ & + 1{,}8203A^2 + 2{,}9658.\end{aligned} \tag{2.11}$$

Якщо результат обчислення (A^2, ε) знаходиться біля лінії S_L, то можна використовувати розподіл з сім'ї S_L. Якщо результат обчислення (A^2, ε) знаходиться вище лінії S_L, то можна використовувати розподіл з сім'ї S_U, а якщо нижче лінії S_L до лінії критичної області – то розподіл з сім'ї S_B. Якщо ж результат обчислення потрапляє у критичну область, то використовувати для апроксимації сім'ї розподілів Джонсона не можна [36].

Отже трансформований еліпс передбачення будується наступним чином. Спочатку за допомогою взаємо-зворотного нормалізуючого перетворення двовимірні негаусівські дані нормалізують таким чином, щоб розподіл нормалізованих даних був наближений до гаусівського. Далі для

нормалізованих даних будують еліпс передбачення. І нарешті, за еліпсом передбачення для нормалізованих даних із використанням зворотного до нормалізуючого перетворення отримують трансформований еліпс передбачення.

Якщо зворотне перетворення (2.7) представити у вигляді (2.12), тоді рівняння для побудови трансформованого еліпса передбачення буде мати вигляд (2.13):

$$X = \psi^{-1}(Z); \tag{2.12}$$

$$\begin{aligned} &\frac{\left(\psi_1(X_1)-m_{Z_1}\right)^2}{S_{Z_1}^2}+\frac{\left(\psi_2(X_2)-m_{Z_2}\right)^2}{S_{Z_2}^2}- \\ &-\frac{2S_{Z_1Z_2}\left(\psi_1(X_1)-m_{Z_1}\right)\left(\psi_2(X_2)-m_{Z_2}\right)}{S_{Z_1}^2S_{Z_2}^2}= \\ &=\frac{2\left(N^2-1\right)\left(S_{Z_1}^2S_{Z_2}^2-S_{Z_1Z_2}^2\right)}{N(N-2)S_{Z_1}^2S_{Z_2}^2}F_{2,N-2,\alpha}. \end{aligned} \tag{2.13}$$

Рівняння (2.13) застосовується для побудови трансформованого еліпсу передбачення у випадку двовимірних даних з негаусівських розподілом.

2.2 Оцінка адекватності математичної моделі

В двовимірному статистичному аналізі в загалі передбачається нормальність ВВ. Отже оцінювання багатовимірної нормальності є важливою проблемою. Деякі автори пропонують графічний метод для рішення цієї проблеми [37], але з точки зору розрахункової вартості та простоти в цій роботі зосередимося на обчисленні багатовимірних асиметрії та ексцесі. Згідно критерію Мардіа [34] обчислюємо багатовимірну асиметрію за формулою (2.14) та багатовимірний ексцесі за формулою (2.15):

$$\beta_1 = \frac{1}{N^2} \cdot \sum_{i,j=1}^{N} \{(Z_i - \bar{Z})^T \cdot S_N^{-1} (Z_j - \bar{Z})\}^3 ; \quad (2.14)$$

$$\beta_2 = \frac{1}{N} \cdot \sum_{i=1}^{N} \{(Z_i - \bar{Z})^T \cdot S_N^{-1} (Z_i - \bar{Z})\}^2 , \quad (2.15)$$

де $\bar{Z}$ - вектор середніх;

S - матриця коваріації

Якщо проводиться статистичний аналіз на немалих вибірках, у випадку гаусівського двовимірного розподілу $\beta_1 = 0$ та $\beta_2 = m(m+2) = 8$, де m=2 ступеням свободи.

Перевірка на адекватність математичної моделі пропонується проводити емпірично, шляхом аналізу отриманих значень.

2.3 Побудова удосконаленої математичної моделі для перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java

Для побудови удосконаленої математичної моделі із застосуванням нормалізуючих двовимірних перетворень в якості вихідних ЕД були розглянуті дані метрик RFC та CBO з 46 Java-застосунків з відкритим кодом, які розміщені на сайті GitHub (https://github.com). Збір та статистичне обчислення даних було виконано за допомогою ресурсу GitHub, наступним чином, спочатку виконувалась фільтрація проектів за ключовими словами Java, Framework, потім треба було скопіювати URL-адресу проекту, та завантажити проект через check out from Version Control з опцією Git платформи IntelliJ IDEA. Обчислення об’єктно-орієнтованих метрик стосовно

якості розробки ПЗ було виконано за допомогою інструменту CK (https://github.com/mauricioaniche/ck).

В якості досліджувальних метрик було обрано RFC та CBO. Метрики RFC і CBO фіксують ступінь зв'язку між класами шляхом підрахунку пар між класами та методів, зовнішніх для даного класу. Крім цього використання цих показників сприяє виявленню можливих недоліків проектування або порушення об'єктно-орієнтовних принципів. В якості випадкової величини *y* взята метрика CBO, випадкової величини *x* – RFC.

Перше, що потрібно зробити – перевірити отримані дані на нормальність за допомогою критерію Мардіа (Mardia's test), якій базується на багатовимірних асиметрії та ексцесі. Нормальні дані не повинні мати викидів, отже, якщо вони є, то це може бути причиною того, що дані не розподіляються за нормальним законом. Якщо викиди будуть знайдені, потрібно видалити їх з аналізу та провести повторну перевірку. Лише після того можна переходити до наступного кроку аналізу даних [38].

Отримані ЕД для вибірок Y та X не відповідають нормальному закону розподілу: $\beta_1 = 2{,}48$ та $\beta_2 = 10{,}98$

Аналіз даних на викиди було отримано за допомогою квадрата відстані Махаланобіса (Mahalanobis squared distance), формули (1.7), (1.8). У таблиці 2.1 наведено фрагмент даних, значення вихідних ЕД *X* та *Y*, нормалізовані значення Z_X та Z_Y, значення T_S для d_i^2 на першій ітерації, для якої F=5,99, m=2, α=0,005.

Таблиця 2.1 – Фрагмент даних

№ п/п	Y	X	Z_Y	Z_X	T_S для d_i^2
1	2	3	4	5	6
4	10059	15890	0,583326397	0,679485	0,360397
5	79670	136233	2,643978012	2,632002	3,459675
6	2163	2882	-0,22667833	-0,27496	0,131738

Продовження таблиці 2.1

1	2	3	4	5	6
7	20009	22694	0,991645402	0,894263	0,637503
8	4410	5423	0,139581138	0,06995	0,159897
9	2198	2510	-0,21852759	-0,34958	0,502711
10	8	15	-2,64260212	-2,61188	3,251478
11	200	292	-1,40171197	-1,47397	1,245832
12	397	1729	-1,07063297	-0,5496	4,383569
13	132	785	-1,59752876	-0,96738	6,677527
14	199	436	-1,4041	-1,27163	0,898328
15	229	236	-1,33699945	-1,57913	2,616703
16	1264	2796	-0,49751146	-0,29134	0,594446
17	4950	7664	0,200094826	0,261925	0,07134
18	2517	3320	-0,14955534	-0,19831	0,115267
19	19373	28402	0,97106218	1,035405	0,604709
20	24807	26446	1,134031252	0,989902	0,94474
21	207	339	-1,38530783	-1,39924	0,964301
22	11940	18112	0,68043216	0,757253	0,37122
23	22276	27904	1,061473747	1,024066	0,587654
24	7319	8376	0,408464913	0,311731	0,302933

На першій ітерації отримали значення викиду проект № 13 PassBox, на наступній ітерації отримали значення викиду проект № 12 Metro_systems. Видаляємо їх з загального набору ЕД та повторюємо перевірку. Загалом за дві ітерації було видалено два проекта. Скоригована таким чином вибірка ЕД для подальшого аналізу склала 44 набора даних.

Згідно формулі (2.11) визначаємо сім'ю розподілів Джонсона та виконуємо нормалізацію даних за допомогою перетворення Джонсона. Згідно розрахунків для ВВ X отримаємо сім'ю S_B Джонсона, для ВВ Y також

визначили перетворення сім'ї S_B Джонсона. Параметри двовимірного перетворення Джонсона для ВВ X та Y одержані за формулами (2.2-2.3) та наведені в табл. 2.2.

Таблиця 2.2 – Параметри двовимірного перетворення Джонсона для ВВ X та Y

Параметр	ВВ Y	ВВ X
γ	1,61212	1,89932
η	0,4975	0,5358
φ	89692,712	170967,483
λ	-9,318	-22,692

Отримані нормалізовані вибірки Z_X та Z_Y відповідають нормальному закону розподілу: $\beta_1 = 0{,}07$ та $\beta_2 = 7{,}608212$ для довірчої ймовірності 0,995.

Для нормального розподілу даних будуємо рівняння еліпсу передбачення за формулою (1.6), фрагмент отриманих значень для побудови еліпсу передбачення наведено в табл. 2.3.

Таблиця 2.3 – Фрагмент даних границь еліпсу передбачення

№ п/п	Z_Y	Z_X	*Верхня границя*	*Нижня границя*
1	2	3	4	5
4	0,583326397	0,679485	1,083905	0,265675
5	2,643978012	2,632002	2,893767	2,33387
6	-0,22667833	-0,27496	0,142497	-0,68862
7	0,991645402	0,894263	1,291474	0,484697
8	0,139581138	0,06995	0,486196	-0,34726
9	-0,21852759	-0,34958	0,067627	-0,76195
10	-2,64260212	-2,61188	-2,31131	-2,87636

Продовження таблиці 2.3

1	2	3	4	5
11	-1,40171197	-1,47397	-1,08454	-1,84302
12	-1,07063297	-0,5496	-0,87366	-1,65204
13	-1,59752876	-0,96738	-1,19486	-1,94159
14	-1,4041	-1,27163	0,126079	-0,70473
15	-1,33699945	-1,57913	0,675791	-0,15556
16	-0,49751146	-0,29134	0,219219	-0,6131
17	0,200094826	0,261925	1,426966	0,629537
18	-0,14955534	-0,19831	1,383365	0,582761
19	0,97106218	1,035405	-1,00645	-1,77269
20	1,134031252	0,989902	1,159252	0,34479
21	-1,38530783	-1,39924	1,416108	0,617873
22	0,68043216	0,757253	0,72478	-0,10562
23	1,061473747	1,024066	1,083905	0,265675
24	0,408464913	0,311731	2,893767	2,33387

На рис. 2.1 зображено на вісі абсцис значення RFC, а на вісі ординат – значення CBO, як бачимо в середині еліпсу передбачення знаходяться усі нормалізовані дані.

Тобто, усі викиди на етапі первинної обробки даних були видалені правильно. Аналіз отриманих даних дозволяє зробити висновок, що найбільш популярні фреймворки на мові Java, утворюють нормалізований розподіл даних. Цей результат дає можливість стверджувати, що згідно метрик RFC і CBO з урахуванням їх взаємозв’язку, два проекти з первинного набору даних мають високу ступінь зв’язку між класами, тобто якщо у відповідь на повідомлення можна викликати велику кількість методів. Тестування та налагодження такого класу ускладнюються. Чим більша кількість методів, які можна викликати з класу, тим більша складність класу.

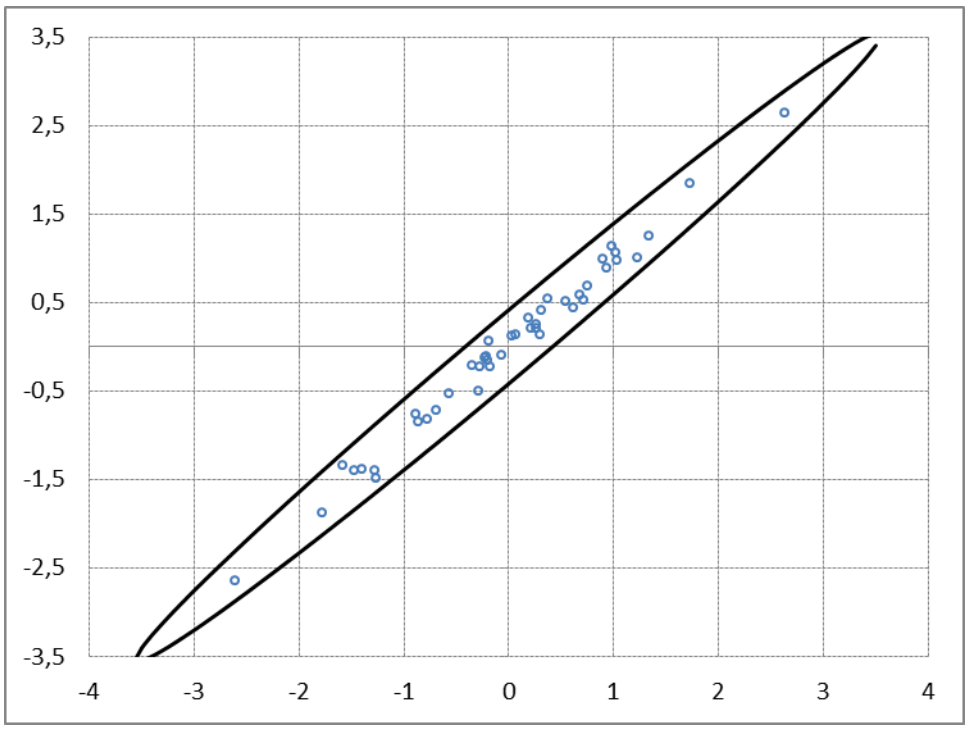

Рисунок 2.1 – Нормалізовані дані за двовимірним перетворенням, рівняння еліпсу передбачення

Застосуємо зворотне перетворення (2.9) та формулу (2.13) для побудови рівняння трансформованого еліпсу передбачення для двовимірних негаусівських даних. Фрагмент отриманих значень для побудови трансформованого еліпсу передбачення наведено в табл. 2.4.

Таблиця 2.4 – Фрагмент даних границь трансформованого еліпсу передбачення

№ п/п	*Y*	*X*	*Верхня границя*	*Нижня границя*
1	2	3	4	5
4	10059	15890	23039,691	7715,54268
5	79670	136233	83343,2263	118343,064
6	2163	2882	4434,68994	1331,85525
7	20009	22694	30865,3049	11362,9106
8	4410	5423	8441,52335	2520,79204
9	2198	2510	3840,44005	1159,80635

Продовження таблиці 2.4

1	2	3	4	5
10	8	15	24,38134	0,31762365
11	200	292	385,846512	135,485041
14	199	436	593,052915	203,134046
15	229	236	307,535291	108,925942
16	1264	2796	4297,35558	1292,0414
17	4950	7664	11843,7142	3591,77276
18	2517	3320	5133,20403	1535,02054
19	19373	28402	36586,8322	14594,7418
20	24807	26446	34704,5676	13469,1763
21	207	339	452,661505	157,650821
22	11940	18112	25727,2083	8882,68296
23	22276	27904	36114,9801	14306,3592
24	7319	8376	12893,8138	3936,64935

Вихідні ЕД та рівняння трансформованого еліпсу передбачення зображено на рис. 2.2, де на вісі абсцис відкладено інтегровані значення RFC за проектами, а на вісі ординат - інтегровані значення CBO.

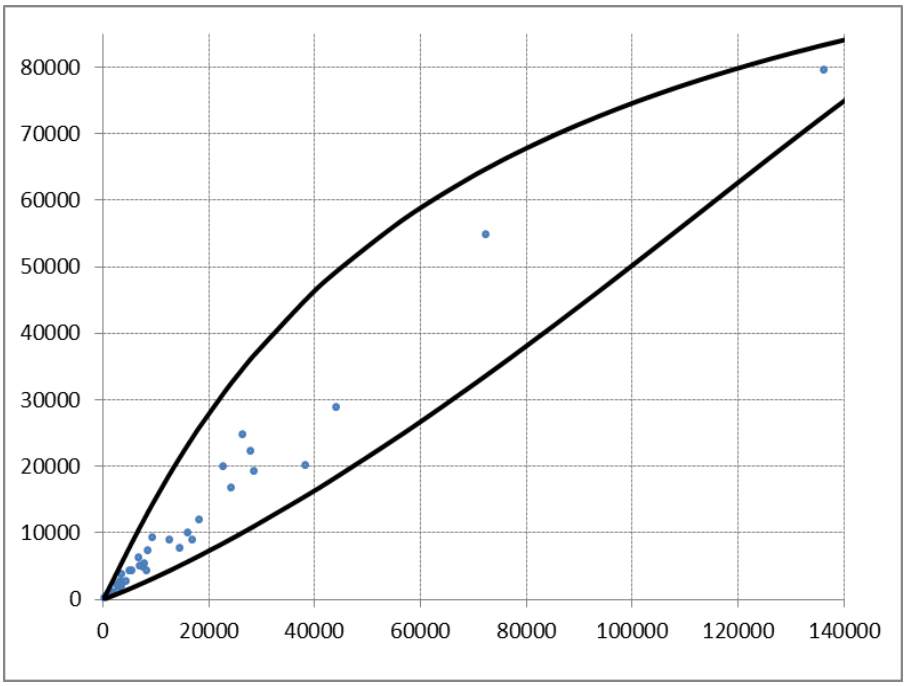

Рисунок 2.2 – Вихідні ЕД, рівняння трансформованого еліпсу передбачення, побудованого на основі двовимірного нормалізуючого перетворення.

Виходячи з аналізу рис. 2.2, ЕД знаходяться в середині інтервалу прогнозування, аналогічно з рис. 2.1 для нормалізованих даних. Тобто, усі викиди на етапі первинної обробки даних були видалені правильно.

Для порівняння на цьому наборі даних була проведена нормалізація центруванням за середнім значенням та діленням на середньоквадратичне відхилення, в наслідок перетворення нормалізований розподіл даних отримати не вдалося, усунення викидів теж не привело до нормалізації даних, тому побудова еліпсу передбачення на негаусівських даних не має сенсу.

Також для порівняння на цьому ВВ була проведена нормалізація за одновимірним перетворенням Джонсона сім'ї S_B. В результаті перетворення отримані ЕД для вибірок Y та X не відповідають нормальному закону розподілу: $\beta_1 = 9{,}32$ та $\beta_2 = 11{,}13$. Також аналіз даних на викиди було отримано за допомогою квадрата відстані Махаланобіса (Mahalanobis squared distance), формули (1.7), (1.8). При значенні довірчої ймовірності рівної 0,995 викидів не було знайдено, як наслідок не можливість отримати нормальний розподіл даних. Тому пропонується зменшити значення довірчої ймовірності до 0,95.

У таблиці 2.5 наведено фрагмент даних, значення вихідних ЕД X та Y, нормалізовані значення Z_X та Z_Y, значення T_S для d_i^2 на першій ітерації, для якої F=3,21, m=2, α=0,05.

Таблиця 2.5 – Фрагмент даних

№ п/п	Y	X	Z_Y	Z_X	T_S для d_i^2
1	2	3	4	5	6
4	10059	15890	0,709926279	0,83591	0,492123
5	79670	136233	2,664939597	2,17949	5,434642
6	2163	2882	-0,15549008	-0,19933	0,04364

Продовження таблиці 2.5

1	2	3	4	5	6
7	20009	22694	1,14387503	1,054439	0,598971
8	4410	5423	0,235277183	0,181653	0,04181
9	2198	2510	-0,14682658	-0,28211	0,256941
10	8	15	-2,15651384	-2,52843	4,580177
11	200	292	-1,34702833	-1,5102	1,383101
12	397	1729	-1,03005115	-0,50411	3,890296
13	132	785	-1,52081455	-0,96543	4,720474
14	199	436	-1,3492188	-1,29565	0,903514
15	229	236	-1,28707631	-1,61944	2,532057
16	1264	2796	-0,44189745	-0,2175	0,733641
17	4950	7664	0,300016786	0,391383	0,169908
18	2517	3320	-0,07343828	-0,11438	0,0262
19	19373	28402	1,122140347	1,192518	0,673299
20	24807	26446	1,29361611	1,148555	0,87427
21	207	339	-1,33193818	-1,43156	1,092636
22	11940	18112	0,813542622	0,916055	0,484823
23	22276	27904	1,217452323	1,181615	0,633459
24	7319	8376	0,522995601	0,445372	0,153513

На першій ітерації отримали значення чотирьох викидів проект № 5 Elasticsearch, проект № 10 KBC--Kaun-Banega-Crorepati, проект № 12 Metro_systems та № 13 PassBox. Видаляємо їх з загального набору ЕД та повторюємо перевірку. Скоригована таким чином вибірка ЕД для подальшого аналізу склала 42 набора даних.

Згідно формулі (2.11) визначаємо сім'ю розподілів Джонсона та виконуємо нормалізацію даних за допомогою одновимірного перетворення Джонсона. Згідно розрахунків для ВВ X отримаємо сім'ю S_B Джонсона, для

ВВ Y також визначили перетворення сім'ї S_B Джонсона. Параметри одновимірного перетворення Джонсона для ВВ X та Y наведені в табл. 2.6.

Таблиця 2.6 – Параметри одновимірного перетворення Джонсона для ВВ X та Y

Параметр	ВВ Y	ВВ X
γ	4,0738	3,2883
η	0,6023	0,6046
φ	2656146,6433	989017,3043
λ	16,5816	72,8086

Отримані нормалізовані вибірки Z_X та Z_Y відповідають нормальному закону розподілу: $\beta_1 = 0{,}143$ та $\beta_2 = 8{,}673427$ для довірчої ймовірності 0,995.

Для нормального розподілу даних будуємо рівняння еліпсу передбачення за формулою (1.6), фрагмент отриманих значень для побудови еліпсу передбачення наведено в табл. 2.7.

Таблиця 2.7 – Фрагмент даних границь трансформованого еліпсу передбачення

№ п/п	*Y*	*X*	*Верхня границя*	*Нижня границя*
1	2	3	4	5
4	10059	15890	30296,942	8224,23994
6	2163	2882	5409,01062	1180,7153
7	20009	22694	39246,1091	12255,7933
8	4410	5423	11023,52	2433,21885
9	2198	2510	4590,51213	1006,75977
11	200	292	222,787018	56,8149503
14	199	436	435,930808	111,185213
15	229	236	148,381389	36,2401983
16	1264	2796	5219,17124	1140,23928

Продовження таблиці 2.7

1	2	3	4	5
17	4950	7664	15761,9022	3605,56319
18	2517	3320	6379,55144	1389,12064
19	19373	28402	45156,1427	15711,0133
20	24807	26446	43268,8959	14522,0796
21	207	339	289,33118	74,3464214
22	11940	18112	33485,7319	9526,34716
23	22276	27904	44688,1904	15407,9272
24	7319	8376	17201,9691	3988,03089

Вихідні ЕД та рівняння трансформованого еліпсу передбачення, побудованого на основі одновимірного нормалізуючого перетворення, зображено на рис. 2.2. Поснемо рис. 2.2: на вісі абсцис відкладено інтегровані значення RFC за проектами, а на вісі ординат - інтегровані значення CBO.

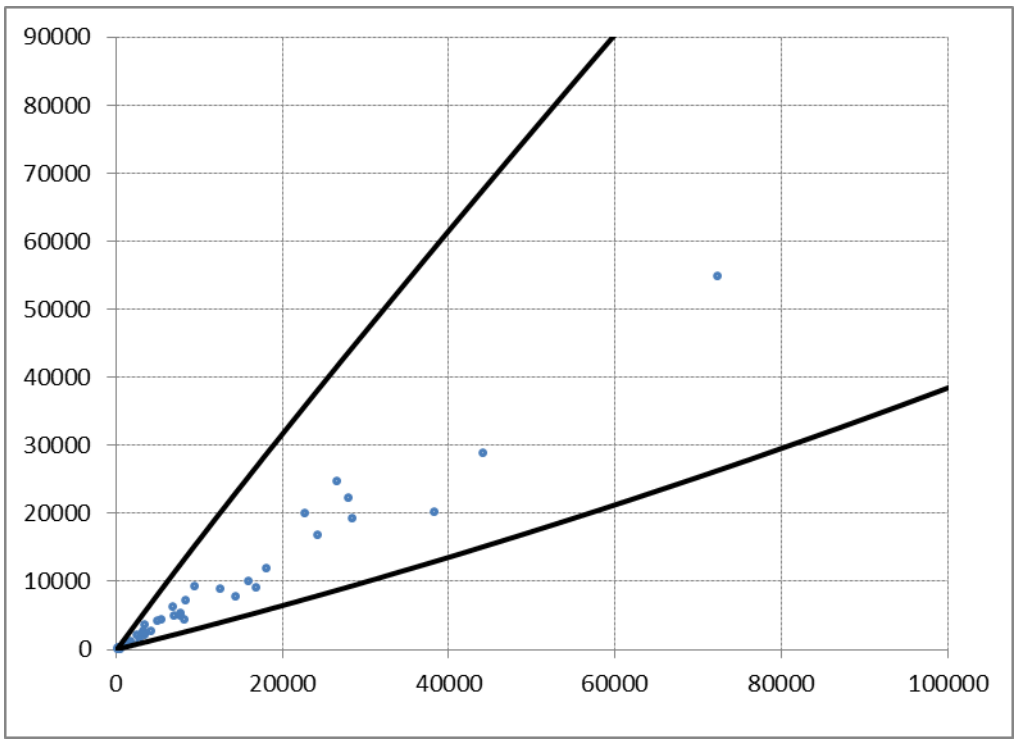

Рисунок 2.3 – Вихідні ЕД, рівняння трансформованого еліпсу передбачення, побудованого на основі одновимірного нормалізуючого перетворення.

Визначення аномалій у взаємозв’язках між частинами застосунків.

Для визначення аномалій у взаємозв’язках між частинами застосунків з точки зору об’єктно-орієнтовного проектування застосуємо математичну модель за рахунок побудови трансформованого еліпсу передбачення на основі двовимірного перетворення Джонсона сім’ї S_B.

В якості тестового значення візьмемо проект Projects зі інтегрованими значеннями метрик CBO 21343 та RFC 57891. Тестове значення проекту та рівняння трансформованого еліпсу передбачення, побудованого на основі двовимірного перетворення Джонсона сім’ї S_B, представлено на рис. 2.4. Пояснення до рис.2.4: вісь абсцис – значення RFC, вісь ординат – значення CBO, проект Projects зображено у вигляді червоного ромба. Значення тестового проекту виходить за межі трансформованого еліпсу передбачення, і це є сигналом для подальшого аналізу тестового проекту. Додатковий аналіз метрик класів CBO та RFC для кожного класу окремо виявив, що опосередковані значення перевищують коефіцієнт 4, а це свідчить про високу зв’язність між класами.

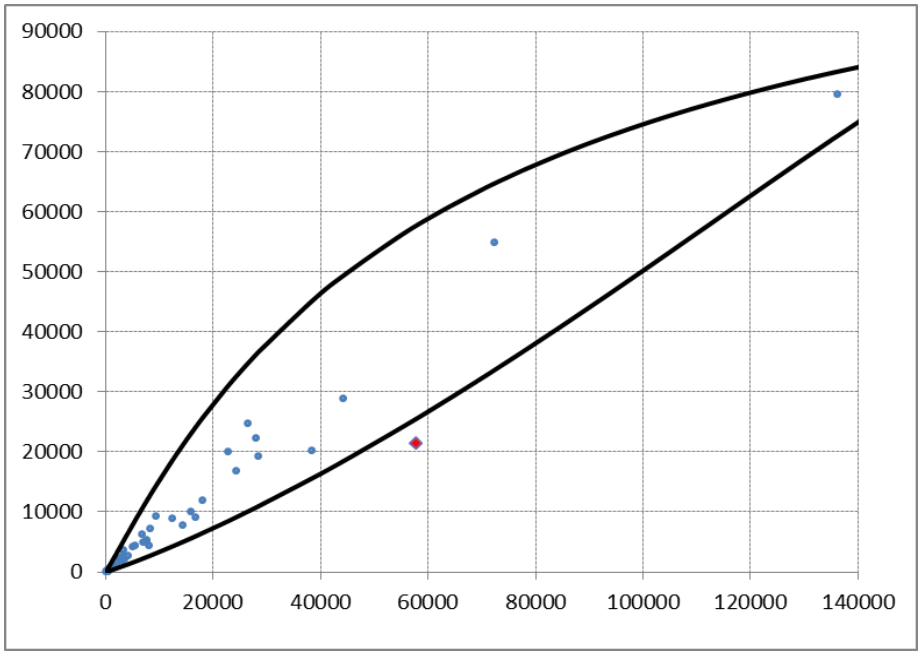

Рисунок 2.4 – Модель1 для перевірки взаємозв'язків між частинами на наявність аномалій.

Для порівняльного аналізу побудуємо модель для перевірки взаємозв’язків між за стосунками за рахунок трансформованого еліпсу передбачення, побудованого на основі одновимірного перетворення Джонсона сім’ї S_B. Перевіримо спроможність виявлення аномалій для Projects зі інтегрованими значеннями метрик CBO 21343 та RFC 57891. Тестове значення проекту та рівняння представлено на рис. 2.5. Пояснення до рис.2.5: вісь абсцис – значення RFC, вісь ординат – значення CBO, проект Projects зображено у вигляді червоного ромба. Отже перевірка на основі другої моделі не виявила аномалій, хоча Projects має високу зв’язність між класами.

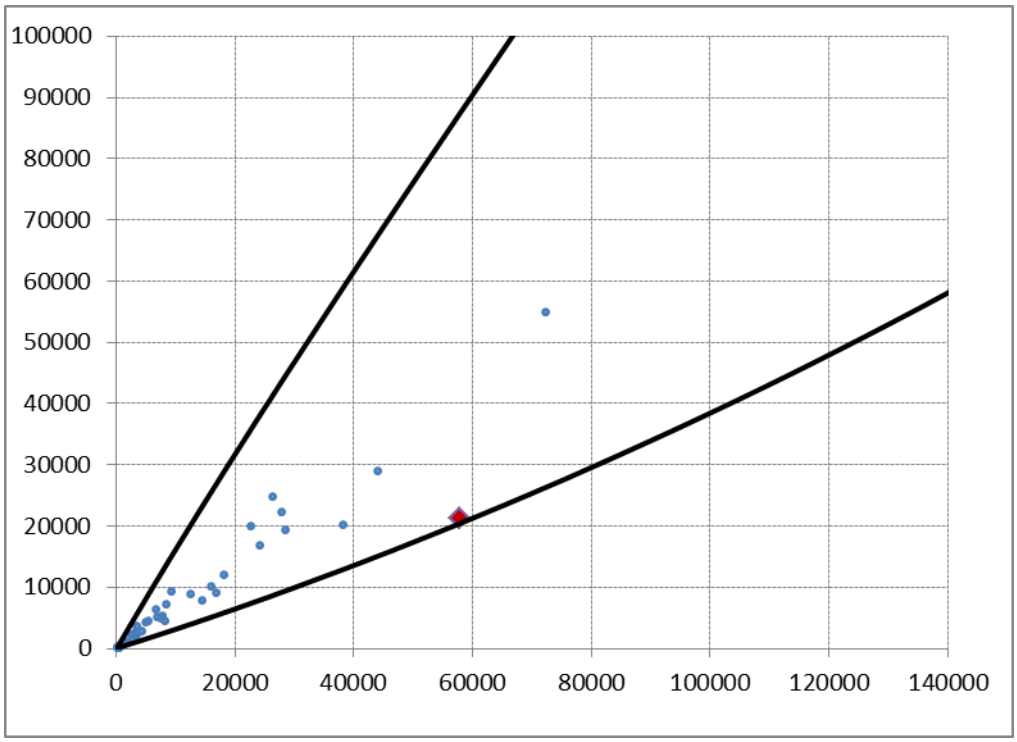

Рисунок 2.4 – Модель2 для перевірки взаємозв'язків між частинами на наявність аномалій.

Висновок:

Виходячи з аналізу отриманих результатів, можемо стверджувати, що двовимірне перетворення Джонсона сім'ї S_B призводить до нормалізації даних з кращим значенням за критерієм Мардіа. Застосування двовимірного перетворення Джонсона, дозволяє врахувати кореляцію метрик CBO та RFC,

що призводить до більш достовірних результатів. Кількість викидів при двовимірному перетворенні Джонсона менша, ніж при одновимірному перетворенні Джонсона. При двовимірному перетворенні кількість викидів склала 2 проекти, ці проекти мають високу ступінь зв’язку між класами, що в свою чергу сигналізує про недоліки в проектуванні Java-застосунку. При одновимірному перетворенні кількість викидів склала 4 проекти.

3 ПРОЕКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПЕРЕВІРКИ ВЗАЄМОЗВ'ЯЗКІВ МІЖ ЧАСТИНАМИ ЗАСТОСУНКІВ, ЩО РОЗРОБЛЕНІ МОВОЮ JAVA

3.1 Ескізний проект програмного забезпечення

3.1.1 Моделювання як засіб представлення процесу розробки ПЗ

Моделювання є важливою складовою частиною процесу розробки програмних застосунків. В сучасній практиці проектування ПЗ використовують візуальні моделі, що являють собою засоби опису, проектування та документування архітектури ПЗ. Серед візуальних моделей найбільш відомою є мова UML (Unified Modeling Language). UML діаграми можуть бути перетворені у вихідний код (пряме перетворення) і навпаки вихідний код може бути перетворений в діаграми (зворотне перетворення). У деяких випадках пряме перетворення може здійснюватися автоматично за допомогою програм конвертерів.

Стандарт UML версії 1.1, прийнятий OMG 1997 р., містить структурні (structural) моделі та моделі поведінки (behavioral) [39].

Структурні моделі містять наступний набір діаграм [40]:

– діаграми класів (class diagrams) – для моделювання статичної структури класів системи та зв'язків між ними;

– діаграми компонентів (component diagrams) – для моделювання ієрархії компонентів (підсистем) системи;

– діаграми розміщення (deployment diagrams) – для моделювання фізичної архітектури системи.

Моделі поведінки містять наступний набір діаграм [40]:

– діаграми варіантів використання (use case diagrams) – для моделювання функціональних вимог до системи (у вигляді сценаріїв взаємодії користувачів із системою);

– діаграми взаємодії (interaction diagrams);

– діаграми послідовності (sequence diagrams) та кооперативні діаграми (collaboration diagrams) – для моделювання процесу обміну повідомленнями між об'єктами;

– діаграми станів (statechart diagrams) – для моделювання поведінки об'єктів системи при переході з одного стану до іншого;

– діаграми діяльності (activity diagrams) - для моделювання поведінки системи у межах різних варіантів використання, чи потоків управління.

3.1.2 Розробка діаграма варіантів використання

Розробка діаграми варіантів використання (Use Case Diagram) дозволить визначити та сформулювати загальні вимоги до функціональної поведінки системи. Діаграма варіантів використання – це послідовність дій (транзакцій), що виконує система у відповідь на подію, що ініціює дійова особа. Елементи діаграми варіантів використання: прецедент, дійова особа, відносини між акторами та варіантами використання [39].

Дійову особу, що користується програмним забезпечення, позначимо «Користувач». Опис дійової особи наведено в таблиці 3.1.

Таблиця 3.1 – Опис суб’єктів взаємодії

Дійова особа	Перелік варіантів використання
Користувач	Користувач має наступні функції: введення даних, розрахунок параметрів одновимірного перетворення Джонсона, розрахунок параметрів двовимірного перетворення Джонсона, перевірка даних на нормальність, нормалізація даних, побудова еліпсу передбачення, побудова трансформованого еліпсу передбачення, оцінювання якості.

Короткий опис варіантів використання надано в таблиці 3.2.

Таблиця 3.2 – Короткий опис варіантів використання

Дійова особа	Назва	Короткий опис
1	2	3
Користувач	Введення даних	Варіант використання дозволяє користувачеві ввести набір даних
Користувач	Розрахунок параметрів для вихідних вибірок	Варіант використання відповідає за розрахунок параметрів для вихідних вибірок
Користувач	Розрахунок параметрів одновимірного перетворення Джонсона	Варіант використання відповідає за розрахунок параметрів одновимірного перетворення Джонсона
Користувач	Розрахунок параметрів двовимірного перетворення Джонсона	Варіант використання відповідає за розрахунок параметрів двовимірного перетворення Джонсона
Користувач	Перевірка даних на нормальність	Варіант використання виконує перевірку даних на нормальність

Продовження таблиці 3.2

1	2	3
Користувач	Нормалізація даних	Варіант використання виконує нормалізацію внесених даних
Користувач	Побудова еліпсу передбачення	Варіант використання відповідає за побудову рівняння еліпсу передбачення
Користувач	Побудова трансформованого еліпсу передбачення	Варіант використання відповідає за побудову рівняння трансформованого еліпсу передбачення
Користувач	Оцінювання якості	Варіант використання виконує оцінювання метрик якості

Діаграма варіантів використання надана на рис. 3.1, на ній виділено 1 актора та відповідні прецеденти.

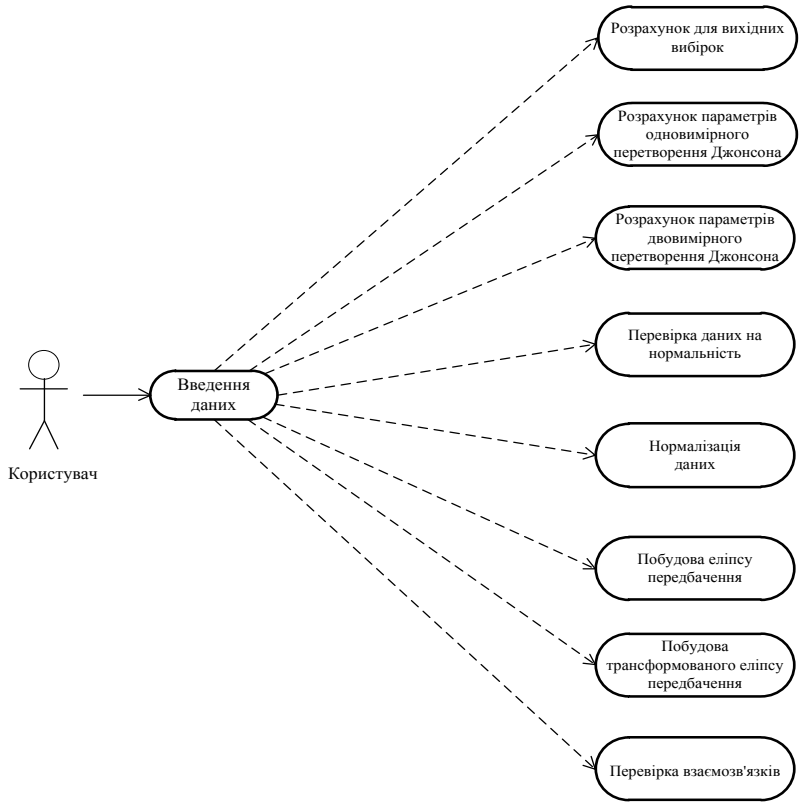

Рисунок 3.1 – Діаграма варіантів використання ПЗ

3.1.3 Розробка специфікації варіантів використання

Потік подій варіанту використання має такі складові [41]:

- короткий опис;
- передумови;
- основний потік подій;
- альтернативний потік подій;
- постумови.

Опис варіантів використання ПЗ наведено у табл. 3.3.

Таблиця 3.3 – Опис варіантів використання ПЗ

Опис	Складова
1	2
Введення даних	
Короткий опис	Варіант використання дозволяє дійовій особі завантажити файл зі статистичними даними.
Передумови	Користувач має попередньо запустити систему та підготувати файл зі статистичними даними.
Основний потік подій	1. Система виводить початкову форму для завантаження файлу. 2. Користувач обирає файл та підтверджує операцію; 3. Система перевіряє допустимість формату файлу; 4. Якщо формат вірний, тоді статистичні дані зчитуються.

Продовження табл. 3.3

1	2
Альтернативний потік подій	1. Система виводить початкову форму для завантаження файлу. 2. Користувач обирає файл та підтверджує операцію; 3. Система перевіряє допустимість формату файлу; 4. Якщо формат хибний, надається повідомлення про невдалу операцію; 5. Надання можливості повторного завантаження файлу.
Постумови	Дані зчитуються з файлу і з ними виконуються подальші операції.
Розрахунок параметрів для вихідних вибірок.	
Короткий опис	Варіант використання розраховує основні параметри для вихідних вибірок: (кількість значень вибірки; вибіркове середнє; дисперсія; середньоквадратичне відхилення; асиметрія; квадрат асиметрії;ексцес; мінімальне значення; максимальне значення).
Передумови	Користувач вибрав правильний формат файлу та натиснув кнопку «Обчислити»
Основний потік подій	1. Система розраховує параметри; 2. Система виводить розраховані параметри;
Альтернативний потік подій	-
Постумови	Виведення інформації, запис інформації у файл
Перевірка даних на нормальність.	
Короткий опис	Варіант використання забезпечує перевірку набору даних на нормальність розподілу

Продовження табл. 3.3

1	2
Передумови	Функція буде доступна після успішного завантаження даних з файлу.
Основний потік подій	1. Користувач натиснув кнопку «Перевірити на нормальність» 2. Система аналізує нормальність розподілу; 3. Система виводить значення багатовимірних асиметрії та ексцесу.
Постумови	-
Нормалізація даних.	
Короткий опис	Варіант використання забезпечує для перевірки набору даних на нормальність розподілу
Передумови	Функція буде доступна після успішного завантаження даних з файлу.
Основний потік подій	1. Користувач вибрав режим «Нормалізація даних»; 2. Користувач активував нормалізація даних за одновимірним перетворенням; 3. Система нормалізує дані; 4. Система виводить отриманий результат.
Альтернативний потік подій	1. Користувач вибрав режим «Нормалізація даних»; 2. Користувач активував нормалізація даних за двовимірним перетворенням; 3. Система нормалізує дані; Система виводить отриманий результат.
Постумови	-
Розрахунок параметрів одновимірного перетворення Джонсона.	
Короткий опис	Варіант використання виконує пошук оптимальних параметрів одновимірного перетворення.

Продовження табл. 3.3

1	2
Передумови	Функція буде доступна після успішного завантаження даних з файлу.
Основний потік подій	1. Користувач вибрав режим «Одновимірне перетворення Джонсона»; 2. Користувач вводить з клавіатури параметри γ, η, ϕ, λ для початкового наближення; 3. Користувач натискає кнопку «Обчислити»; 4. Система обробляє інформацію, обчислює оптимальні параметри γ, η, ϕ, λ. 5. Система виводить параметри нормалізації γ, η, ϕ, λ.
Альтернативний потік подій	-
Постумови	-
Розрахунок параметрів двовимірного перетворення Джонсона.	
Короткий опис	Варіант використання виконує пошук оптимальних параметрів двовимірного перетворення.
Передумови	Функція буде доступна після успішного завантаження даних з файлу.
Основний потік подій	1. Користувач вибрав режим «Двовимірне перетворення Джонсона»; 2. Користувач вводить з клавіатури параметри γ, η, ϕ, λ для початкового наближення; 3. Користувач натискає кнопку «Обчислити»; 4. Система обробляє інформацію, обчислює оптимальні параметри γ, η, ϕ, λ. 5. Система виводить параметри нормалізації γ, η, ϕ, λ.

Продовження табл. 3.3

1	2
Альтернативний потік подій	-
Постумови	-
Побудова еліпсу передбачення.	
Короткий опис	Варіант використання відповідає за побудову рівняння еліпсу передбачення.
Передумови	Цей варіант використання доступний тільки для нормалізованих даних
Основний потік подій	1. Користувач вибрав режим «Побудова еліпсу передбачення»; 2. Користувач натиснув кнопку «Обчислити» на вкладці «Побудова еліпсу передбачення»; 3. Система розраховує коефіцієнти рівняння; 4. Система виводить отримані дані; 5. Користувач натиснув кнопку «Побудувати» на вкладці «Побудова еліпсу передбачення»; 6. Система будує еліпс передбачення.
Постумови	-
Побудова трансформованого еліпсу передбачення.	
Короткий опис	Варіант використання відповідає за побудову рівняння трансформованого еліпсу передбачення.
Передумови	Цей варіант використання доступний тільки для нормалізованих даних

Продовження табл. 3.3

1	2
Основний потік подій	1. Користувач вибрав режим «Побудова трансформованого еліпсу передбачення»; 2. Користувач натиснув кнопку «Обчислити» на вкладці «Побудова трансформованого еліпсу передбачення»; 3. Система розраховує коефіцієнти рівняння; 4. Система виводить отримані дані; 5. Користувач натиснув кнопку «Побудувати» на вкладці «Побудова трансформованого еліпсу передбачення»; 6. Система будує трансформованого еліпс передбачення.
Постумови	-

3.1.4 Побудова діаграми діяльності

Діаграми діяльності використовуються для опису потоку подій, перерахованих у випадках використання. Діаграма діяльності UML показує розбивку конкретної діяльності на кілька компонентів. В даному випадку поняття «діяльність» — це конкретизація певної поведінки у вигляді паралельного та узгодженого послідовного виконання різних підпорядкованих елементів — вкладених видів діяльності та різноманітних дій, у поєднанні з потоками від виходів одного вузла до входів іншого [42].

На рисунках 3.2 – 3.4 зображено декілька діаграм діяльності побудованих на основі варіантів використання, описаних в таблицях 3.1-3.3.

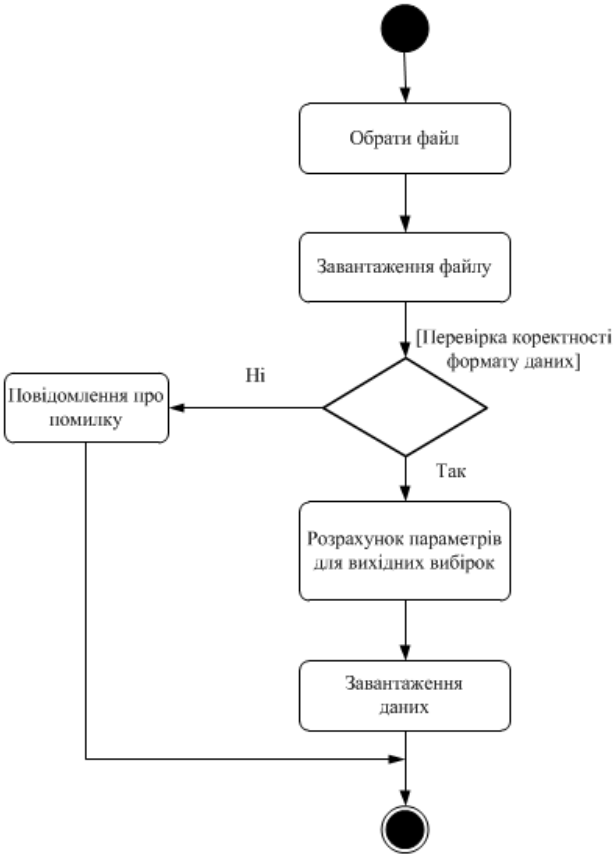

Рисунок 3.2 – Діаграма діяльності для варіанту використання «Введення даних»

Рисунок 3.3 – Діаграма діяльності для варіанту використання «Перевірка даних на нормальність»

Рисунок 3.4 – Діаграма діяльності для варіанту використання «Побудова еліпсу передбачення»

3.1.5 Побудова концептуальної моделі даних

Концептуальна модель - модель предметної області, що складається з переліку взаємопов'язаних понять, що використовуються для опису цієї галузі, поряд з властивостями і характеристиками [43], класифікацією цих понять, типами, ситуаціями, характеристиками в цій області та закономірностями процесів у ній. При побудові концептуальної моделі необхідно визначити: сутності, атрибути сутностей та відносини між сутностями. Концептуальні діаграми рівня зображують прямокутники, атрибути у вигляді еліпсів, а зв’язки у вигляді ромбів.

При розробці ескізного проекту програмного забезпечення для перевірки взаємозв'язків між частинами застосунків, що розроблені мовою

Java, було визначено, що програмне забезпечення такого типу структури не потребує функціональності бази даних. Усі вхідні дані програма буде отримувати за допомогою завантаження файлу та зчитування з нього.

Результати обчислень будуть експортуватися у вихідний файл CSV-формату (Character-separated values).

CSV – це текстовий файл, який містить інформацію. Кожен рядок є окремим рядком таблиці, а стовпці відокремлюються один від одного спеціальними символами - роздільниками (наприклад, комами). Останнім часом роздільником може бути не тільки кома, а й інші символи (пробіл, крапка з комою, табуляція тощо) [44.].

Перевага цього формату полягає в тому, що хоча файл CVS є звичайним текстовим файлом, його можна використовувати як вхідний файл до будь-якої бази даних, зазвичай використовується для передачі даних між базами даних і редакторами електронних таблиць.

3.1.6 Побудова прототипу інтерфейсу користувача

Інтерфейс користувача є засобом зручної взаємодії користувача з програмною системою. В цьому підрозділі представлені схеми екранних форм, систем меню, діалогів і т.д. для проектування зовнішнього інтерфейсу. Розробимо проект інтерфейсу користувача для перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java.

Прототипи деяких екранних форм представлено на рисунках 3.5 – 3.7. Для відображення результатів нормалізації даних було розроблено ескіз екранної форми, що представлено на рисунку 3.5.

Для реалізації діалогу с користувачем та відображення результатів аналізу даних на наявність викидів було розроблено ескіз екранної форми, що представлено на рисунку 3.6.

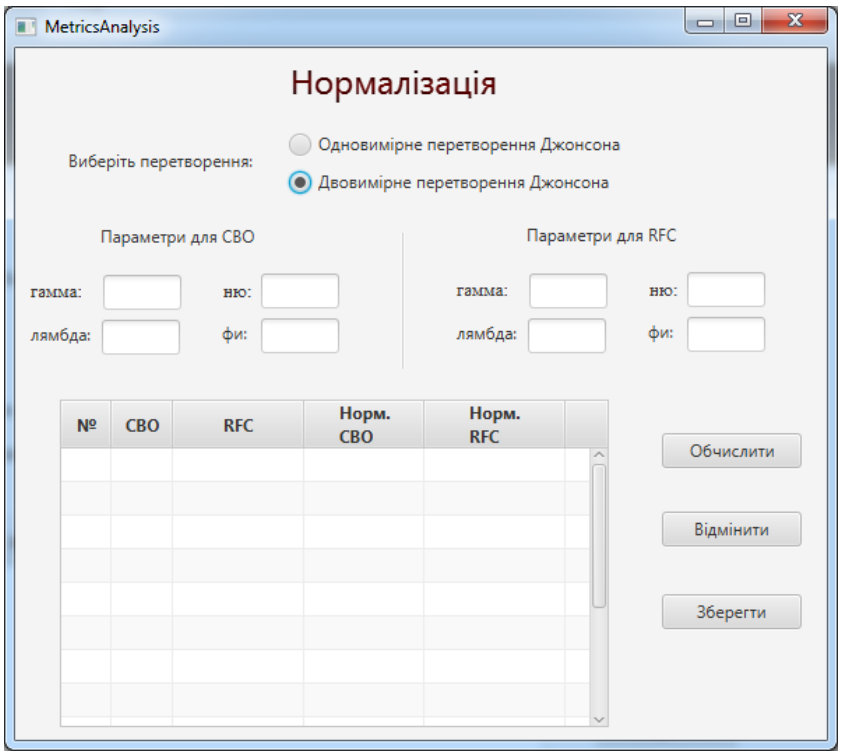

Рисунок 3.5 – Ескіз форми «Нормалізація даних»

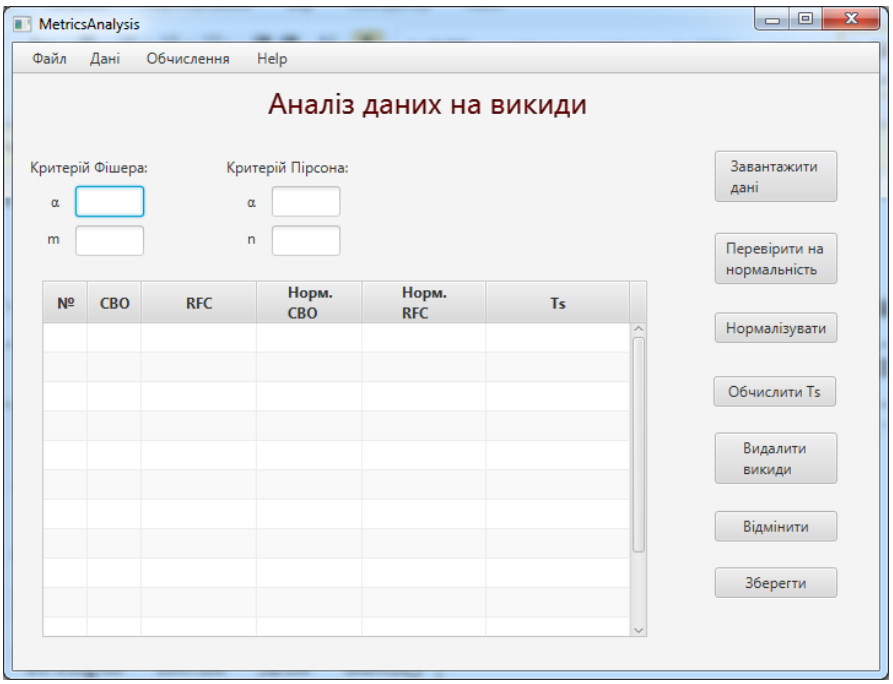

Рисунок 3.6 – Ескіз форми «Аналіз даних на наявність викидів»

Ескіз екранної форми для перегляду результатів розрахунку для перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java, зображений на рисунку 3.7.

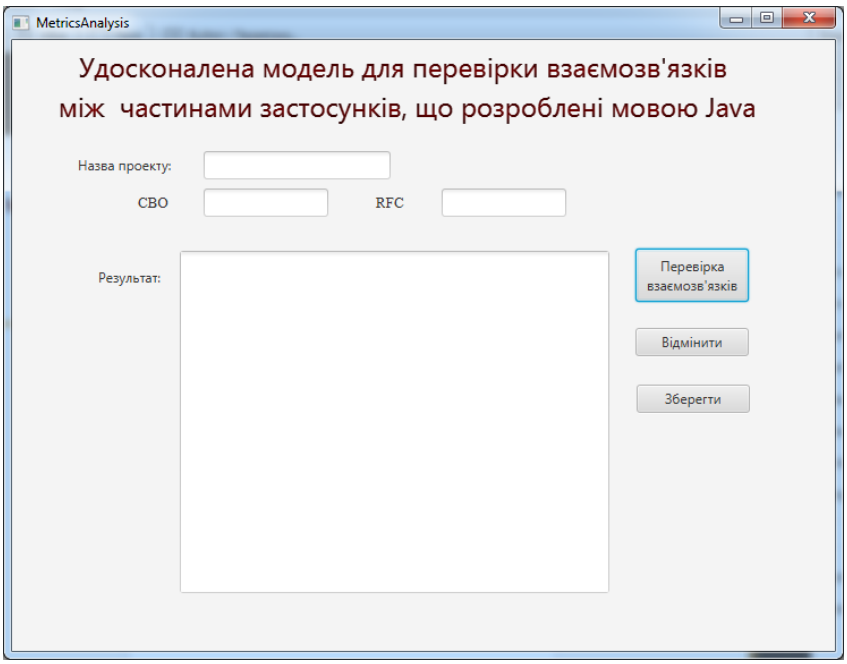

Рисунок 3.7 – Ескіз екранної форми для перегляду результатів розрахунку для перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java.

3.2 Технічний проект програмного забезпечення для перевірки взаємозв'язків між частинами застосунків

3.2.1 Розробка статичної моделі ПЗ

Результати ескізного проектування ПЗ для перевірки взаємозв'язків між частинами Java-застосунків необхідно довести до технічного рівня. Для цього розробимо діаграму класів. Діаграма класів – статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення. Діаграма класів може містити пакети та вкладені пакети [45]. Діаграма класів ПЗ, що розробляється, представлена на рисунку 3.8.

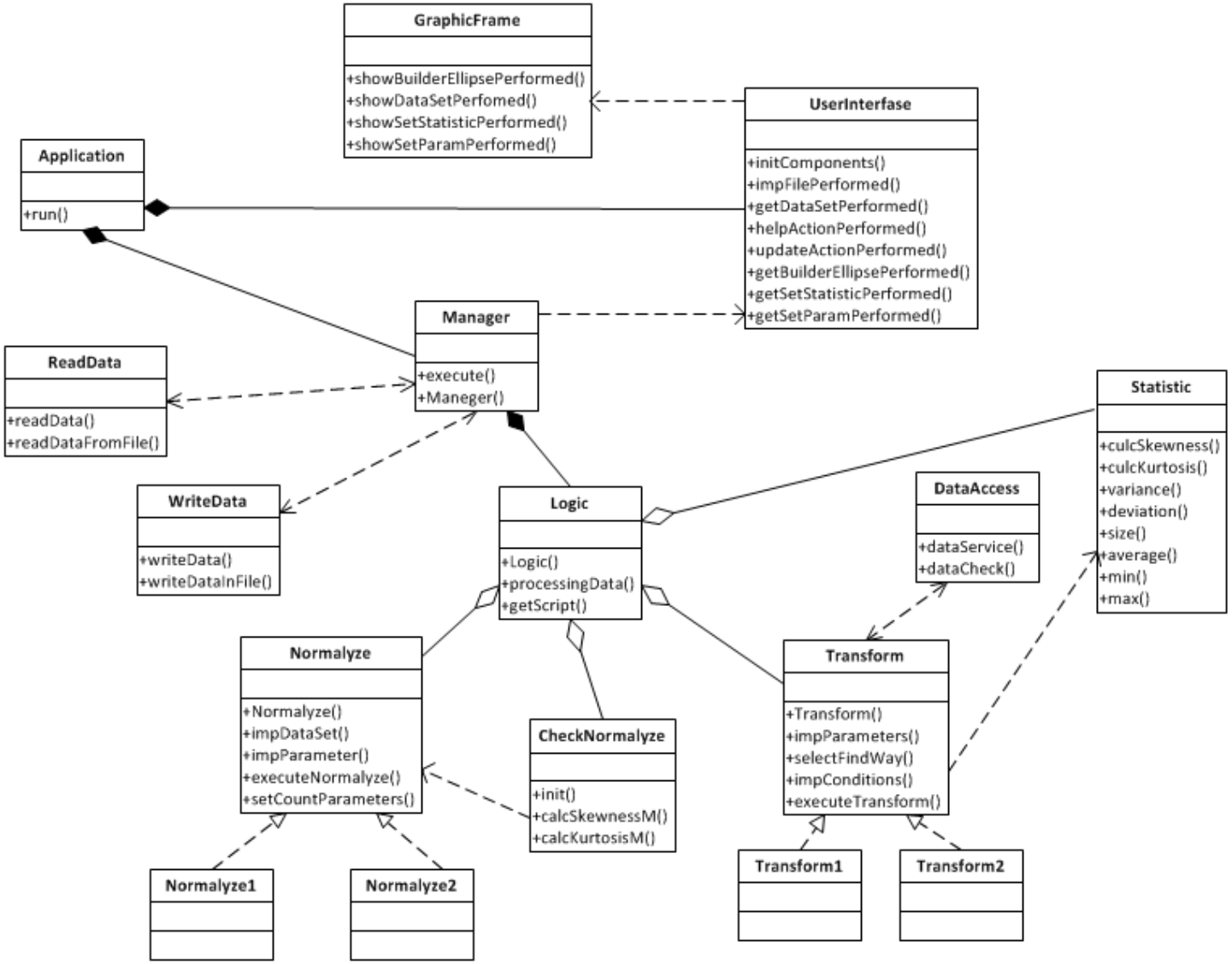

Рисунок 3.8 – Діаграма класів ПЗ

Розглянемо класи, що вказані на діаграмі класів.

Клас Application – головний клас, який містить точку входу в програму (функцію main). Він створює об’єкти класів Manager та UserInterface.

Клас Manager – організує взаємодію між класами, які відповідають за імпорт даних в систему, обробку даних, взаємодію з користувачем. Функції класу Manager:

- зчитування даних з файлу;
- користуючись послугами інших класів здійснювати обробку даних, збереження та видачу необхідних даних, нормалізацію даних, пошук параметрів перетворення Джонсона, обчислення рівняння еліпсу передбачення та обчислення рівняння трансформованого еліпсу передбачення.

Клас UserInterface – відображає інтерфейс користувача. Функція цього компоненту – здійснювати взаємодію з користувачем. Цей клас також здійснює побудову еліпсу передбачення та побудову трансформованого еліпсу передбачення, для чого використовує клас GraphicFrame.

Класи ReadData та WriteData реалізують введення виведення даних.

Клас DataAccess відповідає за трансформацію з набором даних. За допомогою цього класу здійснюється перевірка на викиди, та оновлення набору даних.

У класі Logic реалізовано алгоритм приведення ЕД до нормалізованого набору даних. Основні функції класу (користуючись послугами інших класів): виконати нормалізацію даних, пошук параметрів перетворення Джонсона, обчислення рівняння еліпсу передбачення та обчислення рівняння трансформованого еліпсу передбачення; потім передати отримані результати до класу Manager.

Клас Transform1 реалізовує інтерфейс Transform, за допомогою якого обчислює параметри для одновимірного перетворення Джонсона.

Клас Transform2 реалізовує інтерфейс Transform, за допомогою якого обчислює параметри для двовимірного перетворення Джонсона.

Клас Statistic відповідає за обчислення наступних значень: розмір вибірки; вибіркове середнє; дисперсія; середньоквадратичне відхилення; асиметрія; квадрат асиметрії;ексцес; мінімальне значення; максимальне значення.

Клас BuilderEllipse1 реалізовує інтерфейс BuilderEllipse, за допомогою якого виконує побудову рівняння еліпсу передбачення.

Клас BuilderEllipse2 реалізовує інтерфейс BuilderEllipse, за допомогою якого виконує побудову рівняння трансформованого еліпсу передбачення.

3.2.2 Розробка динамічної моделі ПЗ

Динамічна модель програми може бути представлена у вигляді діаграми послідовності. У роботі представлені діаграми послідовності по деяким варіантам використання.

Діаграма послідовності (sequence diagram) – діаграма, на якій показані взаємодії об’єктів, впорядковані за часом їх прояву [46].

На рис. 3.9 наведено діаграму послідовності на основі раніше представлених варіантів використання.

Згідно варіанту використання «Введення файлу» користувач спочатку вибирає файл з емпіричними даними метрик CBO та RFC. Система перевіряє розширення файлу на допустимість та повертає результат. Потім згідно варіанту використання «Перевірка даних на нормальність» система аналізує дані. Якщо отримали негаусівських розподіл даних, тоді згідно варіанту використання «Нормалізація даних» система нормалізує ЕД за вказаним користувачем перетворенням. Потім здійснюється перевірка на наявність викидів та видалення викидів з вибірки даних.

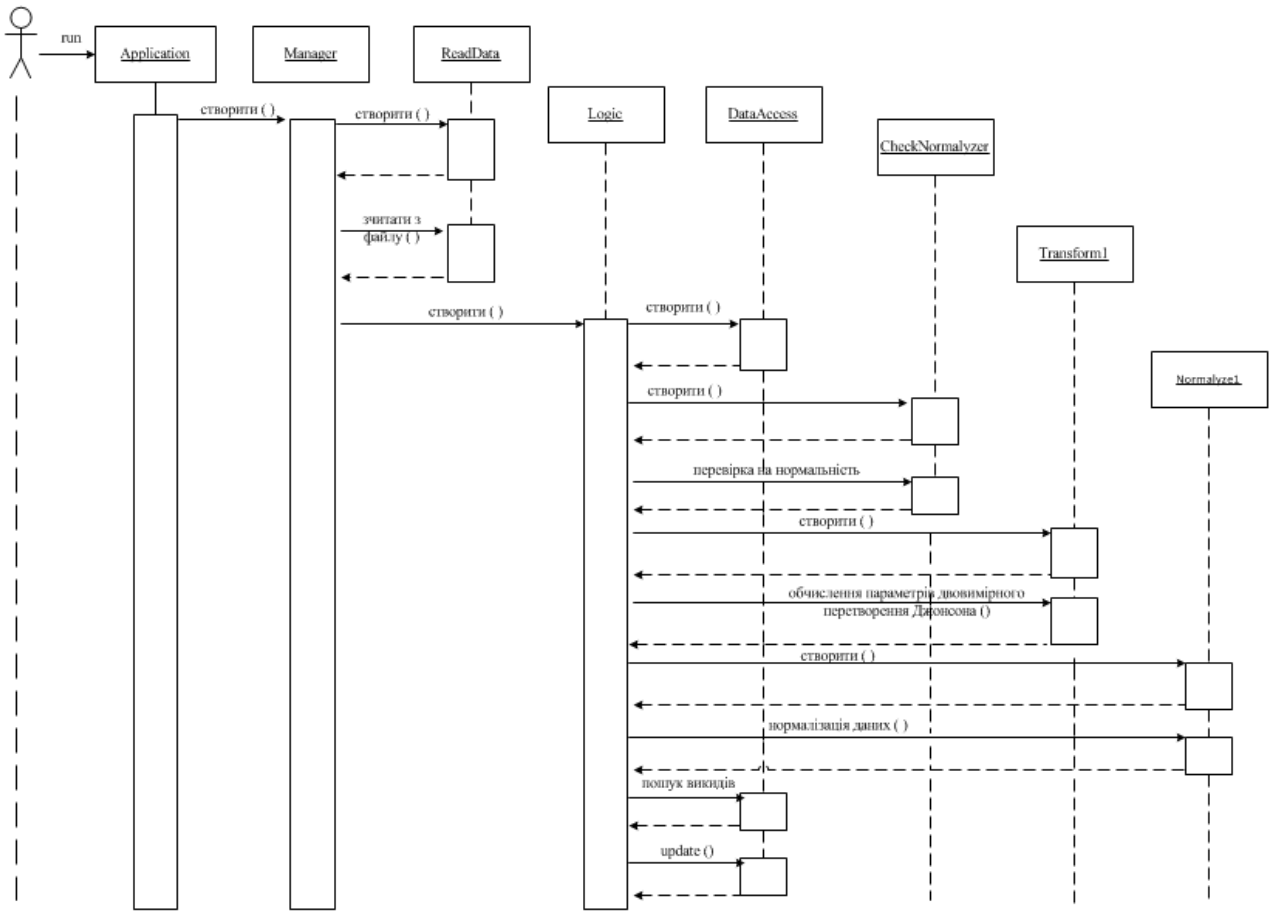

Рисунок 3.9 – Динамічна модель програмного забезпечення

3.3 Робочий проект програмного забезпечення для перевірки взаємозв'язків між частинами застосунків

3.3.1 Обґрунтування вибору мови програмування та засобів розробки

Оскільки аналізуються взаємозв’язок між частинами Java-застосунків, то в якості мови програмування логічно вибрати мову Java. Як відомо Java програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи. Основна перевага використання байт-коду це портативність [47]. Тим не менш, додаткові витрати на інтерпретацію означають, що інтерпретовані програми будуть

майже завжди працювати повільніше, ніж скомпільовані у машинний код, і саме тому Java одержала репутацію «повільної» мови. Проте, цей розрив суттєво скоротився після введення декількох методів оптимізації у сучасних реалізаціях JVM.

Основні можливості Java [47]:

- автоматичне управління пам’яттю;
- розширені можливості обробки виключних ситуацій;
- багатий набір засобів фільтрації вводу/виводу;
- набір стандартних колекцій;
- присутність класів, які дозволяють виконувати HTTP-запити і обробляти відповіді;
- вбудовані в мову засоби створення багатопоточних додатків;
- уніфікований доступ до баз даних:
    - на рівні окремих SQL-запитів – на основі JDBC, SQLJ;
    - на рівні концепції об’єктів, які мають здатність до збереження в базі даних – на основі Java Data Objects і Java Persistence API;
- підтримка шаблонів;
- паралельне виконання програм.

В якості середовища розробки оберемо платформу IntelliJ IDEA. Для розробки інтерфейсу користувача будемо використовувати бібліотеку JavaFX. Для побудови графіків будемо використовувати бібліотеку JFreeChart.

Програмне забезпечення буде розроблюватися та тестуватися в операційній системі Window 10, проте розроблювана підсистема є кросплатфоменною, її можна розгорнути на всіх платформах, які підтримують JVM.

3.3.2 Тестування програмного забезпечення

Для реалізації програмного забезпечення було обрано мову програмування Java. Текст програмного забезпечення наведено в додатку Г.

Розробка програмного забезпечення була виконана зі застосуванням методології TDD (Test Driven Development). Реалізація модульного тестування була виконана на основі фреймворка JUnit5. Були розроблені модульні тести до класів пакетів logic та utils, текст деяких тестових класів наведено в додатку Г.

В табл.3.4 представлена спрощена форма Test Cases для вимог до графічного інтерфейсу (GUI requirements).

Вимоги:

Користувач вводить дані вручну на етапі оцінювання ПЗ, а саме на етапі обчислення параметрів двовимірного перетворення Джонсона. Він повинен ввести чотири параметри для початкового наближення.

Таблиця 3.4 – Спрощена форма Test Cases до форми «Обчислення параметрів двовимірного перетворення Джонсона»

Шаг	Дія	Очікуваний результат
1	2	3
1	Відкрити вкладку «Двовимірне перетворення Джонсона» у додатку MetricsAnalysis. Чотири поля для вводу тексту та 2 кнопки повинні бути доступні.	Список елементів присутній.

Продовження таблиці 3.4

1	2	3
2	В поле «Гамма» введено символ «Й».	Виникає повідомлення про недопустимість вводу символьних даних
3	В поле «Гамма» введено символ «6,566».	Число відображається в полі «Гамма»
4	В поле «Ню» введено символ «Z».	Виникає повідомлення про недопустимість вводу символьних даних
5	В поле «Ню» введено символ «0,566».	Число відображається в полі «Ню»
6	В поле «Фі» введено символ «Zdfd».	Виникає повідомлення про недопустимість вводу символьних даних
7	В поле «Фі» введено символ «-8,566».	Число відображається в полі «Фі»
8	В поле «Лямбда» введено символ «!».	Виникає повідомлення про недопустимість вводу символьних даних
9	В поле «Лямбда» введено символ «5».	Число відображається в полі «Лямбда»

Висновок: за результатами тестування вимог до класів, що реалізують варіант використання «Двовимірне перетворення Джонсона», додаток працює правильно. Тестування інших класів виконується аналогічно.

3.3.3 Випробування програмного забезпечення

Згідно вимог наведених в Технічному завданні (див. Додаток А) виконано повне функціональне тестування та тестування на відмову ПЗ. Випробування ПЗ було виконано згідно технічних умов, які заявлені в Технічному завданні.

Всі виявлені недоліки ПЗ були зафіксовані в протоколах та усунуті розробником до моменту впровадження додатку.

Протокол тестування.

1. Проведено перевірку ПЗ на відповідність технічному завданню:

– перевірка роботи функції «Введення даних».

Після вибору файлу з емпіричними даними вони були успішно завантажені в додаток.

– перевірка роботи функції «Розрахунок параметрів для вихідних вибірок»

Після завантаження даних, обчислення параметрів було успішно виконано.

– перевірка роботи функції «Перевірка даних на нормальність»

Після обчислення критеріїв Мардіа, додаток успішно визначив, чи підпорядковуються двовимірний простір даних нормальному закону розподілу.

– перевірка роботи функції «Нормалізація даних»

Додаток успішно виконав нормалізацію даних за двовимірним перетворенням Джонсона.

– перевірка роботи функції «Розрахунок параметрів двовимірного перетворення Джонсона»

Додаток успішно виконав обчислення параметрів для двовимірного перетворення Джонсона.

– перевірка роботи функції «Побудова еліпсу передбачення»

Додаток успішно виконав обчислення рівняння еліпсу передбачення.

– перевірка роботи функції «Побудова трансформованого еліпсу передбачення»

Додаток успішно виконав обчислення трансформованого рівняння еліпсу передбачення.

– перевірка роботи функції «Перевірка взаємозв’язків між частинами застосунків»

Додаток успішно визначив відношення введених даних згідно моделі.

4 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПЕРЕВІРКИ ВЗАЄМОЗВ'ЯЗКІВ МІЖ ЧАСТИНАМИ ЗАСТОСУНКІВ, ЩО РОЗРОБЛЕНІ МОВОЮ JAVA

Результатом кваліфікаційної роботи є додаток для перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java.

Перелік завдань, що були вирішені:

− виконано аналіз існуючих моделей для перевірки взаємозв'язків між частинами застосунків, визначено постановку задачі;

− виконано збір даних по проектам з відкритим вихідним кодом на ресурсі GitHub;

− застосовано двовимірне перетворення Джонсона для нормалізації емпіричних даних;

− виконано перевірку даних на нормальність на основі критеріїв Мардіа;

− побудовано рівняння еліпсу передбачення;

− побудовано рівняння трансформованого еліпсу передбачення;

Також кваліфікаційна робота містить:

– технічне завдання (Додаток А);

– опис програми (Додаток Б)

– текст програми (Додаток В);

– інструкція користувача (Додаток Г);

– програма та методика випробувань ПЗ (Додаток Д).

Додаток був розроблений на мові Java, в якості середовища розробки обрано платформа IntelliJ IDEA.

Програмне забезпечення було протестоване, також були виконані випробування згідно технічного завдання. Результати підтвердили відповідність технічному завданню.

Як результат розробки, на рисунку 4.1 зображено вікно програмного

забезпечення «Удосконалена модель для перегляду результатів розрахунку для перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java».

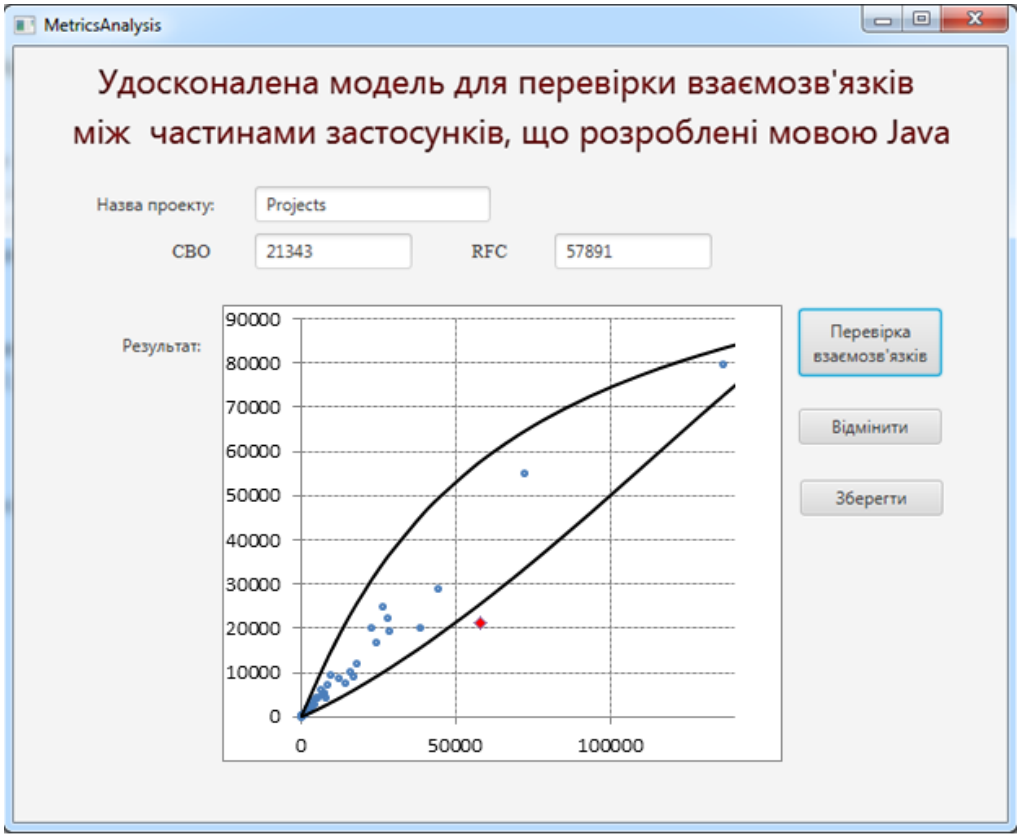

Рисунок 4.1 – Головне вікно програмного забезпечення «Удосконалена модель для перегляду результатів розрахунку для перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java».

5 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ПЕРЕВІРКИ ВЗАЄМОЗВ’ЯЗКІВ МІЖ ЧАСТИНАМИ ЗАСТОСУНКІВ, ЩО РОЗРОБЛЕНІ МОВОЮ JAVA

5.1 Опис програмного забезпечення

Успіх програмних проектів залежить від якості даних, на основі яких приймаються рішення під час стратегічного та тактичного планування. Останнім часом для зменшення ризиків керування проектами по розробці ПЗ використовуються різні методології розробки.

Одним із найважливіших факторів, що впливають на прийняття рішення по пріоритезації задач, включення задач до backlog і т.д. є оцінювання часу виконання робіт.

Основними моделями, які використовуються для перевірки взаємозв’язків між частинами застосунків, є метрики ПЗ, які розглядаються як засіб визначення того, чи відповідає досліджуване програмне забезпечення бажаними властивостями об’єктно-орієнтованого проектування. Це метрики Лі/Генрі, Бріанда, Чідамбера/Кемерера та ін., однак всі ці моделі не завжди адекватно враховують взаємозв’язки між частинами застосунків, оскільки не враховують кореляцію між даними. Тому задача удосконалення методів перевірки взаємозв'язків між частинами застосунків, реалізованих мовою Java, є актуальною.

У зв’язку з вузькою спеціалізацією питання не існує готового ПЗ, яке б в повній мірі вирішувало всі необхідні задачі. Впровадження ПЗ для перевірки взаємозв'язків між частинами застосунків, реалізованих мовою Java, перш за все націлене на зменшення витрат та підвищення ефективності роботи.

Створення ПЗ вимагає одноразових витрат на розробку, придбання необхідних технічних засобів і поточних витрат на функціонування.

Економія від функціонування ПЗ визначається з урахуванням витрат на його експлуатацію. Відношення цієї економії до витрат на створення ПЗ характеризує економічну ефективність капітальних вкладень. Економічні показники визначаються за діючими на момент розрахунку оптовими цінами, тарифами і ставками заробітної плати.

5.2 Розрахунок витрат на створення та експлуатацію програмного забезпечення

Витрати на розробку системи складаються з витрат на заробітну плату розробника, на амортизацію комп`ютера [48], 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 15000 грн.. Вартість сучасного комп`ютера складає 20000 грн. (приведена вартість машини на базі Intel Core i5). При вартості кіловат-години електроенергії рівної 1,68 грн., розраховується вартість розробки програми. Витрати на допоміжні матеріали приведені в табл. 5.1

Таблиця 5.1 – Витрати на допоміжні матеріали

Пункти витрат	Сума грн.
Папір	90,00
Заправлення картриджа до принтера	130,00
Література	300,00
Непередбачені витрати	50,00
Разом матеріали і комплектуючі вироби	570,00

Вартість програми розрахуємо по формулі (5.1):

$$Спр = (Взп + Всф + Взг + Ве) * Т + Вм, \quad (5.1)$$

де: Взп –заробітна плата обслуговуючого персоналу, грн.;

Всф – відрахування в соціальні фонди складає 38% від заробітної плати (ВЗ-венний збір, НДФЛ – податок на доходи фізичних осіб , ЄСВ- єдиний соціальний внесок), грн.;

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

Ве – витрати на електроенергію;

Т – тривалість розробки, міс.

Вм – витрати на основні і допоміжні матеріали.

Ве при споживанні потужності 500 Вт, тривалості роботи на місяць, рівної 21*8 = 168 годин, вартості кіловат-години електроенергії 1,68 грн.

Ве = 168 * 1,68 * 0,5 = 141,12 грн.

Загальні витрати на розробку системи наведені в табл. 5.2.

Таблиця 5.2 – Витрати на розробку системи

Найменування витрат	Одиниця виміру	Кількість
Тривалість розробки	Міс	3
Заробітна плата	Грн.	15000,00
Відрахування в соціальні	Грн.	5700,00
Загальногосподарські витрати	Грн.	960,00
Витрати на допоміжні	Грн.	570,00
Витрати на електроенергію	Грн.	141,12

Відповідно до формули 4.1 вартість програми:

Спр = (15000,00 + 5700,00+ 960,00+ 141,12) * 3 + 570,00=65973,36 грн.

Експлуатаційні витрати для комп`ютера розрахуємо по формулі (5.2):

$$Взр = Аоб + Вм + Ве, \quad (5.2)$$

де: Аоб – амортизаційні відрахування, грн.

Вм – річні витрати на основні і допоміжні матеріали, грн.

Ве – витрати на електроенергію, грн.

Амортизаційні відрахування розраховується по формулі (5.3)

$$Мсм = Вк/Скв, \quad (5.3)$$

де: Мсм – місячна сума амортизації, грн.

ВК – вартість компьютера.

Скв – срок корисного використання.

Місячна сума амортизації = 20000 / 60= 333,33 грн.

Амортизаційні відрахування на устаткування складають за рік:

Аоб = 333,33*12 = 4000,00 грн.

У масштабах закладу річні витрати на основні і допоміжні матеріали (носії, папір, …) визначаються в розмірі 5% вартості основного устаткування:

Вм = 20000 * 0,05 = 1000,00 грн.

Річний обсяг робіт комп`ютера у годинах визначається по формулі (5.4)

$$Фм = 253,3 * Тз, \quad (5.4)$$

де Тз – це середнє денне завантаження устаткування (близько 7 годин)

253,3 – середня кількість робочих днів у році.

Отже, річний обсяг роботи комп`ютера складе:

$$Фм = 253,3 * 7 = 1773,1 \text{ годин}$$

Витрати на електроенергію Ве при 1773,1 годинах роботи устаткування в рік складуть

$$Ве = 1773,1* 1,68 * 0,5 = 1489,41 \text{ грн.}$$

Експлуатаційні витрати для комп`ютера за рік складуть:

$$Взр = 4000,00 + 1000,00 + 1489,41= 6489,41 \text{ грн.}$$

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

Все = 65973,36 + 6489,41 = 72462,77 грн.

5.3 Економічна ефективність розробки та впровадження програмного забезпечення

Основним показником економічної ефективності функціонування ПЗ [49] є зменшення витрат, підвищення ефективності роботи та удосконалена ступень захисту.

До числа найголовніших факторів, що визначають ефективність роботи в зв'язку з впровадженням ПЗ, відносяться:

- підвищення продуктивності праці;
- підвищення швидкості обробки інформації;
- вивільнення робочого часу;
- умовно-річна економія від підвищення якості захисту інформації.

Розрахуємо пряму економічну ефективність від використання ПЗ [50], ґрунтуючись на тому, що впровадження програмного продукту зменшує кількість працівників (за експертною оцінкою фахівців установи близько двох осіб), яких потрібно було б задіяти для обробки інформації на старому ПЗ.

Зарплата 2 працівників в рік складає:

9000 * 12 * 2 = 216000 грн.

Економічний ефект першого року розраховується за формулою (5.5):

$$A_{экон.эф.} = \Delta C_n - E_n * k, \qquad (5.5)$$

де ΔC_n – вивільнені кошти після впровадження програмного продукту (360000 грн.) мінус експлуатаційні витрати (6489,41 грн.) ;

$$\Delta C_n = 216000 - 6489{,}41 = 206710{,}59 \text{ грн.}$$

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

k – одноразові витрати на впровадження продукту (65973,36 грн).

$$A_{экон.эф.} = 216000 - 0{,}6 * 65973{,}36 = 176415{,}98 \text{ грн.}$$

Строк окупності системи розраховується по формулі (5.6):

$$T = \frac{k}{П} = \frac{65973{,}36}{206710{,}59} \approx 0{,}31 \text{ (року)} \quad (5.6)$$

де П – вивільнені кошти після впровадження програмного продукту (206710,59 грн.);

k – одноразові витрати на впровадження системи (65973,36 грн.)

Отже строк окупності системи складає приблизно 3 місяці.

5.4 Висновки

Економічний ефект проявляється в отриманні економії витрат шляхом зменшення витрат, підвищення ефективності роботи та підвищенні якості перевірки взаємозв'язків між частинами застосунків, реалізованих мовою Java.

З обрахованих вище даних видно, що програмний продукт з точки зору річних експлуатаційних даних може мати практичне застосування.

Реалізація даного програмного продукту є доцільною, як бачимо з отриманих значень періоду окупності та показників ефективності.

6 ОХОРОНА ПРАЦІ

6.1 Визначення, цілі, завдання та актуальність питань, пов'язаних з охороною праці

Охорона праці (ОП) – це система соціально-економічних, правових, організаційно-технічних, санітарно-гігієнічних та лікувально-профілактичних заходів та засобів, які спрямовані на збереження життя, здоров’я та працездатності людини під час трудової діяльності.

Ця тема є актуальною, бо у теперішній час Україна знаходиться в небезпечному стані щодо питань ОП, життя та здоров’я суспільства. Багато законів залишаються лише у документі, а насправді не діють. Чимало працівників не мають офіційного оформлення, а тому не можуть законно захищати свої права.

ОП охоплює безпосередньо три головні складові: правові норми трудового законодавства, гігієну та техніку безпеки, виробничу санітарію, а також протипожежний захист і електробезпеку.

Метою ОП є забезпечення безпечних, незагрозливих та придатних для праці умов.

Загальне керівництво роботою по ОП на підприємстві покладається на директора і головного інженера, а безпосереднє керівництво й організацію цієї роботи здійснює особисто головний інженер, або його заступник по охороні праці. У їхньому підпорядкуванні знаходиться відділ ОП, на який покладаються наступні основні функції:

- контроль за дотриманням керівниками цехів, відділів і інших структурних підрозділів діючого законодавства, стандартів, правил і норм по ОП;

- розробка заходів щодо створення здорових і безпечних умов праці;
- проведення вступного інструктажу для людей, які надходять на підприємство і контроль за своєчасним і якісним проведенням інструктажу на робочих місцях;
- участь у роботі комісій з перевірки знань інженерно-технічних працівників в області техніки безпеки і виробничої санітарії, по розслідуванню причин аварій і нещасних випадків, по розгляду проектів будівництва, капітального ремонту цехів і устаткування;
- проведення паспортизації санітарно-технічного стану цехів;
- облік потерпілих від нещасних випадків на виробництві, проведення аналізу і складання звітів про виробничий травматизм, контроль за освоєнням коштів і впровадженням заходів, спрямованих на поліпшення умов праці;

6.2 Аналіз небезпечних і шкідливих факторів

При організації умов праці необхідно враховувати вплив на працюючих небезпечних і шкідливих виробничих факторів, які можуть привести до травми або іншого раптового різкого погіршення здоров'я та захворювання або зниження працездатності.

Небезпечні і шкідливі виробничі фактори (ДСТ 12.0.003-74) [49] підрозділяються по природі дії на чотири групи: фізичні, хімічні, біологічні і психофізіологічні.

До небезпечних фізичних факторів відносяться: машини і механізми, що рухаються; різні підйомно-транспортні пристрої і переміщувані вантажі; незахищені рухливі елементи виробничого устаткування (приводні і передавальні механізми, різальні інструменти, пристосування, що

обертаються і переміщуються й ін.); відлітаючі частки оброблюваного матеріалу та інструменту, електричний струм, підвищена температура поверхонь устаткування й оброблюваних матеріалів і т.д.

Шкідливими для здоров'я фізичними факторами є: підвищена чи знижена температура повітря робочої зони; висока вологість і швидкість руху повітря; підвищені рівні шуму, вібрації, ультразвуку і різних випромінювань – теплових, іонізуючих, електромагнітних, інфрачервоних і ін..

Хімічні небезпечні і шкідливі виробничі фактори за характером дії на організм людини підрозділяються на наступні підгрупи: загально токсичні, подразнюючі, сенсибілізуючі (ті, що викликають алергійні захворювання), канцерогенні (ті, що викликають розвиток пухлин), мутогені (ті, що діють на статеві клітини організму). У цю групу входять численні пари і гази: пари бензолу і толуолу, окис вуглецю, сірчистий ангідрид, окисли азоту, аерозолі свинцю та ін., токсичні пили, що утворюються, наприклад, при обробці різанням берилію, свинцюватих бронз і латуней і деяких пластмас зі шкідливими наповнювачами. До цієї групи відносяться агресивні рідини (кислоти, луги), що можуть заподіяти хімічні опіки шкіряного покриву при зіткненні з ними.

До біологічних небезпечних і шкідливих виробничих факторів відносяться мікроорганізми (бактерії, віруси й ін.) та макроорганізми (рослини і тварини), вплив яких на працюючих викликає травми або захворювання.

6.3 Заходи щодо запобігання шкідливих факторів при роботі з персональним комп’ютером

Робота з комп'ютером характеризується значною розумовою напругою і нервово-емоційним навантаженням операторів, високою напруженістю

зорової роботи і досить великим навантаженням на м'язи рук при роботі з клавіатурою ЕОМ. Велике значення має раціональна конструкція і розташування елементів робочого місця, що важливо для підтримки оптимальної робочої пози людини-оператора.

У процесі роботи з комп'ютером необхідно дотримуватися правильного режиму праці і відпочинку. В противному випадку у персоналу відзначається значна напруга зорового апарату з появою скарг на незадоволеність роботою, головні болі, дратівливість, порушення сну, утому і хворобливі відчуття в очах, у попереку, в області шиї і руках.

Недостатність освітлення приводить до напруги зору, послабляє увагу, приводить до передчасної стомленості. Надмірно яскраве освітлення викликає осліплення, роздратування і різь в очах. Неправильний напрямок світла на робочому місці може створювати різкі тіні, відблиски, дезорієнтувати працюючого. Усі ці причини можуть привести до нещасливого випадку або профзахворювання, тому настільки важливий правильний розрахунок освітленості.

Існує три види освітлення – природне, штучне і сполучене (природне і штучне разом) .

Природне освітлення – освітлення приміщень денним світлом, яке проникає через світлові прорізи в зовнішніх конструкціях приміщень. Природне освітлення характеризується тим, що міняється в широких межах в залежності від часу дня, пори року, характеру області та інших факторів.

Штучне освітлення застосовується при роботі в темний час доби і вдень, коли не вдається забезпечити нормовані значення коефіцієнта природного освітлення (похмура погода, короткий світловий день). Освітлення, при якому недостатнє по нормах природне освітлення доповнюється штучним, називається сполученим освітленням.

Штучне освітлення підрозділяється на робоче, аварійне, евакуаційне, охоронне. Робоче освітлення, в свою чергу, може бути загальним чи комбінованим. Загальне – освітлення, при якому світильники розміщуються у

верхній зоні приміщення чи рівномірно стосовно до розташування устаткування. Комбіноване – освітлення, при якому до загального додається місцеве освітлення.

Згідно до Сніп II-4-79 у приміщеннях обчислювальних центрів необхідно застосувати систему комбінованого освітлення.

При виконанні робіт категорії високої зорової точності (найменший розмір об'єкта розрізнення 0,3...0,5 мм) величина коефіцієнта природного освітлення (КЕО) повинна бути не нижче 1,5%, а при зоровій роботі середньої точності (найменший розмір об'єкта розрізнення 0,5...1,0 мм) КЕО повинний бути не нижче 1,0%. В якості джерела штучного освітлення зазвичай використовуються люмінесцентні лампи типу ЛБ або ДРЛ, які попарно поєднуються у світильники, та повинні розташовуватися над робочими поверхнями рівномірно.

Вимоги до освітленості в приміщеннях, де встановлені комп'ютери, наступні: при виконанні зорових робіт високої точності загальна освітленість повинна складати 300 лк, а комбінована – 750 лк; аналогічні вимоги при виконанні робіт середньої точності – 200 і 300 лк відповідно.

Обчислювальна техніка є джерелом істотного тепловиділення, яке може привести до підвищення температури і зниження відносної вологості в приміщенні.

У приміщеннях, де встановлені комп'ютери, повинні дотримуватися визначені параметри мікроклімату. У санітарних нормах СН-245-71 встановлені величини параметрів мікроклімату, що створюють комфортні умови. Ці норми встановлюються в залежності від пори року, характеру трудового процесу і характеру виробничого приміщення (таблиця 6.1).

Таблиця 6.1 – Параметри мікроклімату для приміщень, де встановлені комп'ютери

Період	**Параметр мікроклімату**	**Величина**
Холодний	Температура повітря в приміщенні	22…24 °С
	Відносна вологість	40…60 %
	Швидкість руху повітря	до 0,1 м/с
Теплий	Температура повітря в приміщенні	23…25 °С
	Відносна вологість	40…60 %
	Швидкість руху повітря	0,1…0,2 м/с

Шумом називають усякий несприятливо діючий на людину звук. З фізичної точки зору звук являє собою механічні коливання пружного середовища.

Слуховий орган людини сприймає у вигляді чутного звуку коливання пружного середовища, які мають частоту приблизно від 20 до 20000 Гц, але найбільш важливий для слухового сприйняття інтервал від 45 до 10000 Гц.

Сприйняття людиною звуку залежить не тільки від його частоти, але і від інтенсивності і звукового тиску.

Несприятлива дія шуму на людину залежить не тільки від рівня звукового тиску, але і від частотного діапазону шуму, а також від рівномірності впливу протягом робочого часу.

У результаті несприятливого впливу шуму на працюючу людину відбувається зниження продуктивності праці, збільшується брак у роботі, створюються передумови до виникнення нещасливих випадків. Усе це обумовлює велике оздоровче й економічне значення заходів щодо боротьби із шумом.

У таблиці 6.2 зазначені граничні рівні звуку в залежності від категорії ваги і напруженості праці, які є безпечними у відношенні збереження здоров'я і працездатності.

Таблиця 6.2 – Граничні рівні звуку, дБ, на робочих місцях.

Категорія напруженості праці	**Категорія важкості праці**			
	I Легка	**II Середня**	**III Важка**	**IV Дуже важка**
I Мало напружена	80	80	75	75
II Помірковано напружена	70	70	65	65
III Напружена	60	60	–	–
IV Дуже напружена	50	50	–	–

Для підтримки тривалої працездатності людини велике значення має режим праці і відпочинку. Під раціональним фізіологічно обґрунтованим режимом праці і відпочинком мається на увазі таке чергування періодів роботи з періодом відпочинку, при якому досягається висока ефективність суспільно-корисної діяльності людини, гарний стан здоров'я, високий рівень працездатності і продуктивності праці.

У таблиці 6.3 представлені зведення про регламентовані перерви, які необхідно робити при роботі на комп'ютері, в залежності від тривалості робочої зміни, видів і категорій трудової діяльності з ВДТ (відеодисплейний термінал) і ПЕОМ (відповідно до СанПіН 2.2.2 542-96 «Гігієнічні вимоги до відеодисплейних терміналів, персональних електронно-обчислювальних машин і організації робіт»).

Таблиця 6.3 – Час регламентованих перерв при роботі на комп'ютері

Категорія роботи с ВДТ чи ПОЕМ	Рівень навантаження за робочу зміну при видах роботи з			Сумарний час регламентованих перерв, хв	
	Група А, кількість знаків	Група Б, кількість знаків	Група В, годин	При 8-годинній зміні	При 12-годинній зміні
I	до 20 000	до 15 000	до 2,0	30	70
II	до 40 000	до 30 000	до 4,0	50	90
III	до 60 000	до 40 000	до 6,0	70	120

Примітка. Час перерв даний при дотриманні зазначених Санітарних правил і норм. При невідповідності фактичних умов праці вимогам Санітарних правил і норм час регламентованих перерв варто збільшити на 30%. У відповідності із СанПіН 2.2.2 546-96 усі види трудової діяльності, пов'язані з використанням комп'ютера, розділяються на три групи:

група А: робота зі зчитування інформації з екрана ВДТ чи ПОЕМ із попереднім запитом;

група Б: робота з введення інформації;

група В: творча робота в режимі діалогу з ЕОМ.

Ефективність перерв підвищується при сполученні з виробничою гімнастикою або організацією спеціального приміщення для відпочинку персоналу зі зручними м'якими меблями, акваріумом, зеленою зоною і т.п.

Створення сприятливих умов праці і правильне естетичне оформлення робочих місць на виробництві має велике значення як для полегшення праці, так і для підвищення його привабливості, що позитивно впливає на продуктивність праці.

7 ОХОРОНА НАВКОЛИЩНЬОГО СЕРЕДОВИЩА

7.1 Вплив людини на навколишнє середовище

Охорона навколишнього середовища (ОНС) є формою відносин між природою та суспільством. Існують різні засоби, через які вона може здійснюватися. Серед них: економічні, правові, гігієнічні, науково-технічні, біологічні та інші.

Відносини у галузі охорони навколишнього природного середовища в Україні регулюються Законом України № 1268-XII від 26.06.91р. «Про охорону навколишнього природного середовища», 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. [50]

В міру прискорення темпів науково-технічного прогресу дія людей на природу стає все більш могутньою. І в даний час воно вже 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 токсичних для організмів. Частина з них не включається в природний круговорот і нагромаджується в біосфері, що приводить до небажаних екологічних наслідків.

Накопичення промислових відходів сприяє підвищенню захворюваності людей і тварин, прискоренню корозії машин і металевого устаткування, зниженню врожайності сільськогосподарських культур і

продуктивності тваринництва, прискореному і нераціональному використовуванню природних ресурсів і енергії, погіршенню багатьох властивостей екологічних систем, загибелі деяких унікальних природних територіальних комплексів, зникненню окремих видів тварин і рослин.

Основну загрозу для навколишнього середовища в межах галузі розробки ПЗ становлять комплекси персональних комп’ютерів, а саме електромагнітні випромінювання від них.

Дослідження останніх років показали, що електромагнітні випромінювання вищезгаданих електронних 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 проектуванні (конструюванні), виготовленні (будівництві) і експлуатації технічних систем управління 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 шкідливими і небезпечними чинниками, що негативно впливають на організм людини. Наприклад, шкідлива дія на зір надає не при тримання стандартних візуальних ергономічних параметрів екрану, розмір мінімального елементу відображення, мерехтіння зображення, відбивна здатність (відблиски) і ін. Працюючий комп’ютер створює електромагнітне поле, що шкідливо діє на організм людини. Це поле може викликати радіоперешкоди, тобто заважати роботі радіо і телеапаратури, що призводить до зниження надійності технічної системи або системи управління, до збільшення риски виникнення аварійної ситуації у виробничому середовищі. Для забезпечення безпеки роботи з комп’ютером розроблені і повинні повсюдно застосовуватися стандарти на монітори,

вимоги до приміщень для експлуатації комп'ютерів і до організації і устаткування робочих місць.

7.2 Утилізація відходів комп’ютерної техніки

Використання комп'ютерів вимагає вирішення таких важливих питань, як утилізація відходів (мікросхеми з вмістом кольорових металів, плати, диски).

Переробка оргтехніки складається з наступних етапів:

- транспортування обладнання;
- складання паспорта відпрацьованої техніки;
- екологічне дослідження відходів;
- розбирання обладнання на окремі деталі;
- сортування (відділення дорогоцінних металів від забруднюючих домішок);
- утилізація та переробка відходів.

Кожна запчастина комп'ютерної техніки має різну категорію небезпеки, тому для полегшення процесу перевезення, переробки та складування на полігоні деталі упорядковано відповідно до шкідливості. Для зберігання відходів I і II категорії призначені спеціально оснащені приміщення. При цьому деталі упаковуються в цистерни і герметичні контейнери. Відходи III класу менш небезпечні, тому вони зберігаються в текстильних і паперових мішках.

Найбільш популярний і ефективний метод утилізації оргтехніки - термічна переробка відходів. Для цього використовуються циклонні сепаратори, спеціальні мікрохвильові, металоплавильні печі та інше обладнання.

При утилізації старих комп'ютерів відбувається їх розділ на сім фракцій: метали, пластмаси, штекери, дроти, батареї, стекло. Жодна деталь не йде для повторного використання, оскільки не можна гарантувати їх надійність, але у формі вторинної сировини вони йдуть на виготовлення нових комп'ютерів або інших пристроїв.

Детально розглянемо декілька прикладів переробки відходів обчислювальної техніки.

Гадолінієво-галлеві грати (ГГГ) використовуються у виробництві компонентів пристроїв, що запам'ятовують. В ході обробки біля 80% вихідного матеріалу перетворюється на відходи або відбраковується. ГГГ мають високу вартість і їх виділення з відходів представляє інтерес з економічної точки зору.

При здобутті досить чистих продуктів можливі повторне їх використання як вихідний матеріал. При цьому значно підвищується економічність виробництва заготівок з ГГГ. Під терміном відходи маються на увазі кристалічні залишки (залишки середовища для зростання кристалів, частини кристалів, виробництва, що утворюються на різних стадіях), а також дрібний порошок, що виходить при різанні, шліфовці, поліровці кристалів граната або подібних матеріалів.

Переробка цих відходів протягом ряду останніх років викликає труднощі і не вирішена до цих пір. Всі попередні спроби були безуспішними із-за низької розчинності цих складних оксидів.

Вдосконалений процес, розроблений Е. Гуссетом (патент США 4-198231 від 15 квітня 1980 року, фірма «Свісс Алюміній Лтд.», Швейцарія), призначений для виділення галію і гадолінія з відходів, що містять обидва ці елементи у вигляді оксидів або з'єднань, що перекладаються в оксиди. Відходи дрібно подрібнюються і потім розчиняються в сильних мінеральних кислотах. Гадоліній осідає з очищених розчинів у вигляді оксалата, галій виділяється в металевому вигляді електролітично. Електролітичне виділення галію може проводитися до виділення гадолінія у вигляді оксалата з розчину.

Розглянемо приклад проведення такого процесу. Відходи піддаються переробці, є залишки завантаження в пристрої для зростання кристалів, розколені частини кристалів зі всіх стадій переробки, дрібнозернисті порошки і пудри після операції різання, шліфовки і поліровки гранатів GdGaO. Змельчений порошок після обробки кристалів ГГГ висушується при 1200^0C і потім нагрівається при 6000^0C протягом декількох годин для розкладання летких забруднень.

Дрібнозернисті відходи в кількості 1000 г розміром менше 40 мкм., що містять 34% галію і 46% гадолінія кип'ятяться із зворотним холодильником протягом двох годин в 2100 мл 35%-ной соляної кислоти. Це відповідає 99% завантаження. Після кип'ячення частина відходів, що не розчинилася, фільтрується і промивається. Після висушування залишок важить 20 г. Залишки такого типа об'єднуються і піддаються повторній обробці кип'ячення. Фільтрат, що включає промивні води, об'ємом 2300 мл, обробляється 4000 г металевого галію при 500^0C протягом 45 хвилин. Метал має максимально диспергувати.

В ході цієї операції благородніші елементи, присутні в розчині, виділяються і частково розчиняються в галії до його насичення і далі охолоджуються у вигляді інтерметалевих включень або в елементарній формі. Висадження можна проводити з меншою кількістю галію. Метал може періодично замінюватися на нову порцію до досягнення необхідної міри очищення. В результаті процесу очищення виходить розчин з вмістом галію 140 г/л і гадолінія 190 г/л.

Встановлюється величина *ph*=1 шляхом додавання 900 мл 4% розчину перекису водню для окислення домішок заліза. Осадження гадолінія проводиться при 500^0C шляхом додавання 1500 г кристалічної технічної щавлевої кислоти COH*2HO; суспензія акуратно перемішується 12 годин для повторного осадження у вигляді оксалата гадолінія.

Оксалат гадолінія Gd(CO)*10HO відділяється центрифугуванням, промивається 20 мл розбавленої щавлевої кислоти (6 г/л) і висушується при

1300^0C; перетворення на оксид гадолінія досягається прожаренням при 8000^0C. Подальше очищення може проводитися розчиненням в кислоті і осадженням у вигляді окасалата гадолінія. До рідини після центріфугованія і промивань осаду (3300 мл) додають 2300 г КОН при інтенсивному перемішуванні до рн=12,6000 мл розчину з вмістом галію 55 г/л і гадолінія 1 г/л піддається електролізу при 600 Із з використанням катода з нержавіючої сталі і графітового анода при щільності струму близько 0,1 А/кв.см. Після 48 годин осідає 325 г галію і залишається розчин з вмістом 0,4 г/л, що не піддається подальшій переробці. Обложений метал має чистоту 99,99% і може бути безпосередньо перетворений в оксид.

Сучасна технологія виготовлення елементів засобів обчислювальної техніки дозволяє досягти дуже низького рівня відмов елементів під час експлуатації. У зв'язку з цим відпадає необхідність проведення ремонтних робіт на місці експлуатації сучасних засобів обчислювальної техніки і як наслідок не утворюються відходи (несправні мікросхеми), що містять дорогоцінні і рідкоземельні метали. Природно, в сервісних центрах, що спеціалізуються на ремонті і технічному обслуговуванні комп’ютерів, має бути організований збір і облік матеріалів, що містять коштовні метали, з подальшою обробкою цих матеріалів на спеціалізованих заводах з метою їх вилучення. У зв'язку з тим, що вітчизняне виробництво сучасних компонентів інформаційних технологій знаходиться в сьогоднішні дні лише в зачатковому стані, комп’ютери складаються з парку імпортних машин і устаткування. Із-за відсутності інформації про вміст дорогоцінних металів в елементах устаткування, строгий облік не представляється можливим і має бути покладений на фахівців експортних фірм. В умовах ринкової економіки підприємства мають бути самі зацікавлені у вторинній переробці, що містять дорогоцінні метали вузлів і елементів за умови неможливості їх використання.

Технологічний процес витягання дорогоцінних металів з плат здійснюється за наступною схемою. Плати сортуються по переважанню в них

кількості дорогоцінних металів, дробляться і подрібнюються, обпалюються і плавляться. В процесі випалення піролітичному розкладанню піддається пластмасова основа, а основа дорогоцінних металів у вигляді металевих залишків відновлюється до оксидів. Металевий залишок подрібнюється, гранулюється, проходить магнітну сепарацію і відбувається відділення магнітних від немагнітних часток. Отриманий таким чином порошок, розділений по видах дорогоцінних металів, у вигляді гранул розплавляється в індукційних плавильних печах з подальшим розділенням кожного металу окремо.

ВИСНОВКИ

Результатом виконання кваліфікаційної роботи є підвищення достовірності визначення аномалій у взаємозв’язках між частинами застосунків, що розроблені мовою Java, з точки зору об’єктно-орієнтовного проектування. Удосконалення математичної моделі для перевірки взаємозв'язків між частинами застосунків було виконано за рахунок використання трансформованого еліпсу передбачення, побудованого на основі двовимірного нормалізуючого перетворення Джонсона сім’ї S_b.

Всі поставлені завдання до кваліфікаційної роботи виконані в повному обсязі і отримані наступні результати:

- проаналізовано та порівняно існуючі моделі перевірки взаємозв'язків Java-застосунків;
- обґрунтовано необхідність удосконалення математичної моделі для перевірки взаємозв'язків Java-застосунків;
- зібрані необхідні метрики з кожного проекту;
- нормалізовані отримані емпіричні дані на основі двовимірного нормалізуючого перетворення Джонсона сім’ї S_b;
- перевірені емпіричні дані на викиди;
- побудовано рівняння трансформованого еліпсу передбачення для нормалізованих даних;
- удосконалено математичну модель для перевірки взаємозв'язків Java-застосунків;
- розроблено програму для перевірки взаємозв'язків застосунків, реалізованих мовою Java. Програма дозволяє визначити аномалії у взаємозв'язках між частинами застосунків, забезпечує зберігання результатів визначення, а також надає користувачу швидкий доступ до попередніх результатів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ISO/IEC/IEEE 12207:2017 Systems and software engineering – Software life cycle processes. URL: https://www.iso.org/ru/standard/63712.html (дата звернення: 28.10.2021).

2. Quality of Open Source Software: how many eyes are enough? URL: https://blogs.worldbank.org/opendata/quality-open-source-software-how-many-eyes-are-enough (дата звернення: 01.11.2021).

3. Examining open source security and the road ahead in the 2017 Coverity Scan Report. URL: https://www.synopsys.com/blogs/software-security/2017-coverity-scan-report-open-source-security/ (дата звернення: 01.10.2021).

4. TIOBE Index for November 2021. URL: https://www.tiobe.com/tiobe-index/ (дата звернення: 21.10.2021).

5. Juliano R., Travençolo B., Soares M. Detection of Software Anomalies Using Object-oriented Metrics. *In Proceedings of the 16th International Conference on Enterprise Information Systems (ICEIS-2014).* 2014. P. 241-248 . – ISBN: 978-989-758-028-4

6. Bassey Isong, Obeten Ekabua. State-of-the-art in empirical validation of software metrics for fault proneness prediction: systematic review. *International Journal of Computer Science & Engineering Survey (IJCSES).* 2015.Vol.6, No.6 – DOI:10.5121/ijcses.2015.6601

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

8. Prykhodko S.B. Statistical anomaly detection techniques based on normalizing transformations for non-Gaussian data. *Computational Intelligence*

(Results, Problems and Perspectives). Proceedings of the International Conference, Kyiv-Cherkasy, Ukraine, May 12-15, 2015, P. 286-287.

9. Prykhodko S., Prykhodko N., Makarova L., Pugachenko K. Detecting Outliers in Multivariate Non-Gaussian Data on the basis of Normalizing Transformations. *Electrical and Computer Engineering : the 2017 IEEE First Ukraine Conference (UKRCON) «Celebrating 25 Years of IEEE Ukraine Section»,* Kyiv, Ukraine, May 29 – June 2, 2017. P. 846-849.

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

11. Prykhodko S.B., Prykhodko N. V., Kudin O. O., Smykodub T. G. Constructing the transformed prediction ellipses on the basis of normalizing transformations for bivariate non-Gaussian data. *Проблемы информационных технологий.* 2017. № 1 (021). С.134-138. ISSN 1998-7005

12. Prykhodko S.B., Prykhodko N. V., Makarova L. M., Kudin O. O., Smykodub T. G. Detecting bivariate outliers on the basis of normalizing transformations for non-Gaussian data. *Advanced Information Systems and Technologies: proceedings of the V international scientific conference, Sumy, May 17-19 2017 / Edited by S. I. Protsenko, V. V. Shendryk.* – Sumy: Sumy State University, 2017. P.95-97.

13. Stanfield P.M., Wilson J.R., Mirka G.A., Glasscock N.F., Psihogios J.P., Davis J.R. Multivariate input modeling with Johnson distributions. *Proceedings of the 28th Winter simulation conference WSC'96,* December 8-11, 1996. Coronado. CA. USA. P. 1457-1464.

14. Приходько С.Б., Смикодуб Т. Г. Математична модель для перевірки взаємозв'язків між частинами Java-застосунків за метриками RFC та CBO. *Інформаційні технології: моделі, алгоритми, системи (ITMAS – 2021): Матеріали II Всеукраїнської науково-практичної Інтернет конференції (26-*

28 жовтня 2021 р.). – Миколаїв: НУК імені адмірала Макарова, 2021. С.25-26.

15. Emanuel A.W.R., Wardoyo R., Istiyanto J.E., Mustofa K. Modularity Index Metrics for Java-Based Open Source Software Projects. *Journal of Advanced Computer Science and Applications*. 2011. 2(11).

16. Jinghui Cheng, Jin L.C. Guo. Activity-Based Analysis of Open Source Software Contributors: Roles and Dynamics. *IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE). May 2019*. 2019. DOI:10.1109/CHASE.2019.00011

17. Опенсорс приложения – что это, и чем они лучше коммерческого ПО? URL: https://apix-drive.com/ru/blog/useful/open-source (дата звернення: 07.09.2021).

18. Measuring the Cost of Software Quality of a Large Software Project at Bombardier Transportation. URL: https://www.researchgate.net/publication/236398768_Measuring_the_Cost_of_Software_Quality_of_a_Large_Software_Project_at_Bombardier_Transportation – (дата звернення: 17.10.2021).

19. Vishal Midha, Prashant Palvia. Factors affecting the success of Open Source Software. *Journal of Systems and Software*. 2012. Vol.85. (4), P. 895-905

20. 1061-1998-IEEE Standard for a Software Quality Metrics Methodology . URL: https://ieeexplore.ieee.org/document/749159 (дата звернення: 11.10.2021).

21. Makarova L.M., Prykhodko N.V., Kudin O.O. Constructing the non-linear regression model for size estimation of Web-applications implemented in Java. *Вісник ХНТУ*. Херсон. 2019. № 2(69), С.145-154

22. 1061-1998-IEEE Standard for a Software Quality Metrics Methodology. URL: https://ieeexplore.ieee.org/document/749159.(дата звернення: 11.10.2021).

23. Chidamber S., Kemerer C. A Metrics Suite for Object-Oriented Design. *IEEE Transactions on Software Engineering*. June.1994. P. 476-492.

24. Li W., Henry S. Maintenance Metrics for the Object-Oriented Paradigm. *Proceedings of the 1st International Software Metrics Symposium (ISMS'93)*. Baltimore. Maryland. USA. 1993. May 21 -22. P. 52-60.

25. Briand L. C., Daly J., Wuest J. A Unified Framework for Coupling Measurement in Object-Oriented Systems. *IEEE Transactions on Software Engineering*. vol. 25. no. 1. 1999. P. 91 -121.

26. Chidamber S., Kemerer C. A Metrics Suite for Object Oriented Design. *IEEE Transactions on Software Engineering*. 1994. 20(6). 476-493.

27. Abreu F. B., Goulao M., Esteves R. Toward the Design Quality Evaluation of Object-Oriented Software Systems. *Proceedings of the 5th International Conference on Software Quality*. Austin. Texas. USA. 1995. October 23-26. P. 43-57.

28. Li W., Henry S. Object-Oriented Metrics that Predict Maintainability. *J. Systems and Software*. 1993. vol. 23. P. 111-122.

29. Subramanyam R., Krishnan M.S. Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects. *IEEE Trans. Software Eng*. 2003. No.29. P. 297-310.

30. Calculating the Metrics of UML Class Diagrams. URL: https://www.umlzone.com/articles/calculating-the-metrics-of-uml-class-diagrams/ (дата звернення: 18.10.2021).

31. Вайсфельд М. Объектно-ориентированный подход. Библиотека программиста. СПб.: Питер Пресс, 2020. 256 с.

32. Chew V. Confidence, prediction and tolerance regions for the multivariate normal distribution. *Journal of the American Statistical Association*. Vol. 61. 1966. P.605-617.

33. Friendly M. Elliptical Insights: Understanding Statistical Methods Through Elliptical Geometry. *Statistical Science*. Vol. 28. 2013. No. 1. P.1-39.

34. Mardia K. V. Measures of multivariate skewness and kurtosis with applications. Biometrika. 1970. 3(57), 519–530. doi:10.1093/biomet/57.3.519.

35. Johnson N.L. System of Frequency Curves Generated by Methods of Translation. Biometrica. 1949. Vol. 36, № ½. P. 149–176.

36. Приходько С.Б., Макарова Л.Н., Приходько А.С. Аналитическая зависимость для выбора семейства распределений Джонсона. *Проблеми інформаційних технологій*. Херсон: ХНТУ. 2016. №02 (020). С. 105-110.

37. Henze N. Invariant tests for multivariate normality: a critical review. *Statistical Papers*. 2002. 43. P. 467–506.

38. Prykhodko S., Prykhodko N., Makarova L., Pugachenko K. Multivariate outlier detection technique based on normalizing transformations for non-Gaussian data. *Інформаційні технології та комп`ютерне моделювання: матеріали статей Міжнародної науково-практичної конференції*. 15-20 травня 2017 року. Івано-Франківськ: п. Голіней О.М. 2017. С. 170-174.

39. Моделювання UML. URL: http://ni.biz.ua/8/8_8/8_88488_modelirovaniya-UML.html (дата звернення: 18.10.2021).

40. Uml діаграма компонентів опис. Моделювання UML. Діаграми реалізації. Діаграми огляду взаємодії. URL: https://maccase.ru/uk/android/uml-diagramma-komponentov-opisanie-modelirovanie-na-uml-diagrammy.html (дата звернення: 18.10.2021).

41. Литвин В.В., Пасічник В.В., Шаховська Н.Б. Проектування інформаційних систем. – Львів: «Магнолія 2006», 2021. –380 с.

42. UML-діаграма. Види діаграм UML. URL: http://poradu.pp.ua/nauka/25991-uml-dagrama-vidi-dagram-uml.html (дата звернення: 18.10.2021).

43. Створення концептуальної моделі. URL: http://ni.biz.ua/3/3_6/3_65278_tema--sozdanie-kontseptualnoy-modeli.html (дата звернення: 15.10.2021).

44. Csv это. URL: https://seoblog.life/uchebnik/vordpress/nastrojka-vordpress/csv-eto.html (дата звернення: 19.10.2021).

45. Діаграма класів. URL: https://studopedia.org/10-134736.html (дата звернення: 19.10.2021).

46. Діаграми послідовності. URL: http://um.co.ua/1/1-5/1-53877.html (дата звернення: 19.10.2021).

47. Java. URL: https://www.tadviser.ru/index.php/%D0%9F%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%82:Java (дата звернення: 19.10.2021).

48. Розрахунок повної собівартості одиниці нової продукції. URL: https://web.posibnyky.vntu.edu.ua/fmib/18nikiforova_ekonomika_organizaciya_vyrobnictva/p5.html (дата звернення: 19.10.2021).

49. Мищишин О.Я. Опорний конспект лекцій з дисципліни “Ефективність інформаційних систем” з освітньо-кваліфікаційного рівня “Магістр” для спеціальності “Інформаційні технології в бізнесі ”. – Львів:, 2017. – 98 с.

50. Методика оцінки об'єктів інтелектуальної власності. URL: http://magazine.faaf.org.ua/metodika-ocinki-ob-ektiv-intelektualnoi-vlasnosti-praktichniy-aspekt.html (дата звернення: 19.10.2021).

51. ГОСТ 12.0.003-74 Небезпечні та шкідливі виробничі фактори. Класифікація. ГОСТ (Міждержавний стандарт). Дата оновлення: 26.04.2019. URL: http://online.budstandart.com/ua/catalog/doc-page.html?id_doc=48127 (дата звернення: 12.03.2021).

52. Про охорону навколишнього природного середовища: Закон України № 1268-XII від 26.06.91р. URL: https://zakon.rada.gov.ua/laws/show/1264-12#Text (дата звернення: 12.13.2021)

Додаток А
Технічне завдання

Вступ

Назва розроблюваного проекту: «ПЗ для перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java». Галузь застосування – підприємство, яке займається розробкою ПЗ.

1. Підстави для розробки

Підставою для розробки є завдання на кваліфікаційну роботу, видане кафедрою ПЗАС ННІКНУП НУК ім. адмірала Макарова.

2. Призначення розробки

2.1 Експлуатаційне призначення

Дана розробка призначена для визначення аномалій у взаємозв’язках між частинами застосунків, що розроблені мовою Java, з точки зору об’єктно-орієнтовного проектування.

2.2 Функціональне призначення програми

Функціональним призначенням ПЗ є автоматизація процесів:

– вводу статистичних даних;

– нормалізації вихідних вибірок за допомогою двовимірного перетворення Джонсона;

– розрахунку параметрів для вихідних та нормалізованих вибірок;

– побудови рівняння трансформованого еліпсу передбачення;

– визначення аномалій у взаємозв’язках між частинами застосунків.

3 Вимоги до програмного забезпечення

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу виконуваних функцій

Програмне забезпечення повинно виконувати наступні функції:

1) Загальні функції:

– імпорт вхідних даних з файлу;

– розрахунок параметрів для вихідних вибірок;

– розрахунок параметрів двовимірного перетворення Джонсона;

– нормалізація вихідних вибірок за допомогою двовимірного нормалізуючого перетворення Джонсона;

– розрахунок параметрів для нормалізованих вибірок;

– перевірка вихідних та нормалізованих вибірок на нормальність розподілу;

– побудова рівняння еліпсу передбачення для нормалізованих даних;

– побудова рівняння трансформованого еліпсу передбачення;

– перевірка взаємозв’язків між частинами застосунків;

– експорт результатів у файл.

3.1.2 Вимоги до організації вхідних та вихідних даних

Введення даних здійснюється за допомогою пристроїв введення даних (клавіатура або миша). Дані вводяться через завантаження файлу у вікні програми та за допомогою ручного вводу.

Виведення даних здійснюється за допомогою пристрою виведення даних – монітору комп’ютера.

Обмін інформацією між ПЗ відбувається за допомогою файлів з довільним доступом з розширенням .csv.

Вхідні дані: параметри нормалізації Джонсона (γ, η, ϕ, λ) для першого наближення.

Вихідні дані:

1) Розраховані параметри для вихідної вибірки:
   - кількість значень вибірки;
   - вибіркове середнє;
   - дисперсія;
   - середньоквадратичне відхилення;
   - асиметрія;

- квадрат асиметрії;
- ексцес.

2) Розраховані параметри одновимірного перетворення Джонсона: γ, η, ϕ, λ.
3) Розраховані параметри для нормалізованої вибірки:
   - кількість значень вибірки;
   - вибіркове середнє;
   - дисперсія;
   - середньоквадратичне відхилення;
   - асиметрія;
   - квадрат асиметрії;
   - ексцес.
4) Параметри для перевірки вихідних та нормалізованих вибірок на нормальність розподілу:
   - значення багатовимірної асиметрії ;
   - значення багатовимірного ексцесу.
5) Графік кривої еліпса передбачення виводиться на монітор користувача. Значення верхньої та нижньої границі записуються у вихідний файл.
6) Графік кривої трансформованого еліпса передбачення виводиться на монітор користувача. Значення верхньої та нижньої границі записуються у вихідний файл.
7) Результат перевірки взаємозв’язків між частинами застосунків, реалізованих мовою Java.

3.2 Вимоги до надійності

Програмний продукт повинен нормально функціонувати при безперебійній роботі ПК. При виникненні збоїв в роботі, відновлення нормальної роботи повинне проводитися після перезавантаження програми.

Відмови програми можливі внаслідок некоректних дій користувача при взаємодії з програмою. Щоб зменшити кількість відмов програми за

вказаною вище причиною слід забезпечити зрозумілість програми, через надання підказок кінцевому користувачеві у разі невірних дій.

3.3 Вимоги до умов експлуатації

Необхідний рівень підготовки користувачів: мінімальні навички в користуванні комп'ютером. Комп'ютер призначений для роботи в закритому опалювальному приміщені при наступних умовах навколишнього середовища:

- температура повітря від +10°С до +30°С;
- відносна вологість повітря не більше 80
- запиленість повітря не більше 0,75 мг/м^2.

3.4 Вимоги до складу та параметрів технічних засобів

Система користувача повинна задовольняти вказаним мінімальним вимогам для коректного запуску програми:

– CPU: Intel i5-10400 ;
– RAM: 4 GB;
– вільної пам'яті на жорсткому диску не менше 500 Мб.

3.5 Вимоги до інформаційної та програмної сумісності

Для повноцінного функціонування системи необхідно використовувати: ОС Windows 7 – 32-bit/64-bit або вище.

3.6 Вимоги до програмної документації

Програмна документація повинна містити:

– технічне завдання.
– опис програми
– текст програми
– інструкція користувача
– програма і методика випробувань ПЗ

4 Техніко-економічні показники

Не розраховуються в зв'язку з тим, що продукт розробляється в рамках кваліфікаційної роботи.

5 Стадії та етапи розробки

Стадії та етапи розробки представлені нижче в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки проекту

Стадії розробки	Етапи робіт	Термін виконання робіт	
		Початок етапу	Кінець етапу
1	2	3	4
1. Технічне завдання	1.1Обгрунтування необхідності розробки програми	06.09.21	11.09.21
	1.2 Розробка технічного завдання	11.09.21	16.09.21
	1.3 Затвердження технічного завдання	16.09.21	28.09.21
2. Ескізний проект	2.1 Розробка ескізного проекту	28.09.21	08.10.21
	2.2 Затвердження ескізного проекту	08.10.21	15.10.21
3. Технічний проект	3.1 Розробка технічного проекту	15.10.21	21.10.21
	3.2 Затвердження технічного проекту	21.10.21	01.11.21
4. Робочий проект	4.1 Розробка робочого проекту	01.11.21	07.11.21
	4.2 Затвердження робочого проекту	07.11.21	15.11.21
	4.3 Розробка документації	15.11.21	20.11.21

6 Порядок контролю та прийому

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

Прийом проводиться відповідно до програми та методики випробувань і документується за допомогою протоколу проведення випробувань. У разі знаходження помилок підчас прийому програмного виробу складається акт про знайдені помилки, який підписуються представниками замовника і розробника, 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

Текст програми

```
package com.statistic.domain;

import lombok.Getter;
import lombok.Setter;

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

@Getter
@Setter
public class Analysis {
    private final int N;
    private List<AnalysisData> dataList;
    private double[] min;
    private double[] max;
    private double[] average;
    private double[] dispersion;
    private double[] medianDeviation;
    private double[] asymmetry;
    private double[] excess;

    private double chi2;
    private double fisher;
    private double coefficient;
    private double beta2;
    private double Sy;
    private double tStud;

    private double[][] throwingMatrix;
    private double[][] inverseThrowingMatrix;
    private double[][] kovarMatrix;
    private double[][] inverseKovarMatrix;
    private double[] mnkResult;

    public Analysis(int n) {
```

```
        this.N = n;
        min = new double[n+1];
        max = new double[n+1];
        average = new double[n+1];
        dispersion = new double[n+1];
        medianDeviation = new double[n+1];
        asymmetry = new double[n+1];
        excess = new double[n+1];
        dataList =  new ArrayList<>();
    }

    public void addAnalysisData(AnalysisData data) {
        dataList.add(data);
    }

    public void removeAnalysisData(AnalysisData data) {
        dataList.remove(data);
    }

    @Override
    public String toString() {
        return new StringJoiner(", ", Analysis.class.getSimpleName() + "[",
"]")
                .add("N=" + N)
                .add("min=" + Arrays.toString(min))
                .add("max=" + Arrays.toString(max))
                .add("average=" + Arrays.toString(average))
                .add("dispersion=" + Arrays.toString(dispersion))
                .add("medianDeviation=" + Arrays.toString(medianDeviation))
                .add("asymmetry=" + Arrays.toString(asymmetry))
                .add("excess=" + Arrays.toString(excess))
                .add("chi2=" + chi2)
                .add("fisher=" + fisher)
                .add("coefficient=" + coefficient)
                .add("beta2=" + beta2)
                .add("Sy=" + Sy)
                .add("tStud=" + tStud)
                .add("throwingMatrix=" + matrixToString(throwingMatrix))
                .add("inverseThrowingMatrix=" +
matrixToString(inverseThrowingMatrix))
                .add("kovarMatrix=" + matrixToString(kovarMatrix))
                .add("inverseKovarMatrix=" +
matrixToString(inverseKovarMatrix))
```

```
                .add("mnkResult=" + Arrays.toString(mnkResult))
                .toString();
    }

    private String matrixToString(double[][] m) {
        StringBuilder sb = new StringBuilder();
        for (double[] d : m) {
            sb.append(Arrays.toString(d)).append('\n');
        }
        return sb.toString();
    }

    public double gettStud() {
        return tStud;
    }

    public void settStud(double tStud) {
        this.tStud = tStud;
    }
}
```

```
package com.statistic.service;

import com.statistic.domain.AnalysisHeader;
import com.statistic.domain.Field;
import com.statistic.logic.AnalysisManager;

import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.List;
import java.util.Set;

@Service
public class DownloadService {
    @Value("${files.path}")
    private String path;


    public Path getStatisticFile(AnalysisManager manager, AnalysisHeader
header, String type, Set<Field> fields) {
```

```
        Path result = getResultingPath(header, type);
        manager.getCsvWriter().writeExel(header, result, fields);
        return result;
    }

    public Path getStatisticFile(AnalysisManager manager, AnalysisHeader
header, String type, Set<Field> fields,
                                 List<String[]> norm) {
        Path result = getResultingPath(header, type);
        manager.getCsvWriter().writeExelWithNormalization(header, result,
fields, norm);
        return result;
    }

    private Path getResultingPath(AnalysisHeader header, String type) {
        Path dir = Paths.get(path);
        if (!Files.exists(dir) && !Files.isDirectory(dir)) {
            dir.toFile().mkdir();
        }
        String fileName = type + "Statistic" + header.getN() + "Params" +
LocalDateTime.now()
                .format(DateTimeFormatter.ofPattern("MM-dd_HH-mm")) + ".csv";
        return Paths.get(this.path + "\\" + fileName);
    }
}
```

```
      package com.statistic.logic;

import com.statistic.domain.*;
import org.apache.commons.math3.distribution.ChiSquaredDistribution;
import org.apache.commons.math3.distribution.FDistribution;
import org.apache.commons.math3.distribution.TDistribution;
import org.apache.commons.math3.linear.LUDecomposition;
import org.apache.commons.math3.linear.MatrixUtils;
import org.apache.commons.math3.linear.RealMatrix;

import java.util.*;
import java.util.stream.Collectors;

public abstract class DefaultStatisticAnalyzer implements Analyzer {
```

```
    @Override
    public void kovarCalculation(double[][] m, Analysis header) {
        int n = header.getN();
        List<AnalysisData> dataList = header.getDataList();

        for (AnalysisData ad : dataList) {
            double[] diff = ad.getDiff();
            double[] regElements = new double[n];
            double reg = 0;

            for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                    regElements[i] += m[j][i] * diff[j+1] ;
                }
                reg += regElements[i] * diff[i+1];
            }

            ad.setRegressionElements(regElements);
            ad.setR(reg);
        }
    }

    @Override
    public double[][] kovarMatrix(Analysis header) {
        int n = header.getN();
        double[][] m = new double[n][n];
        List<Double> tmp = new ArrayList<>();
        List<double[]> diff2 =
header.getDataList().stream().map(AnalysisData::getDiff2).collect(Collectors.
toList());
        List<double[]> elements =
header.getDataList().stream().map(AnalysisData::getMatrixElements).collect(Co
llectors.toList());

        for (int i = 0; i < n; i++) {
            for (double[] d : diff2) {
                tmp.add(d[i+1]); // we need only x members
            }
            m[i][i] = tmp.stream().mapToDouble(e -> e).sum();
            tmp.clear();
        }
```

```
        int k = n;
        for (int i = 0; i < n; i++) {
            for (int j = i+1; j < n; j++) {
                for (double[] d : elements) {
                    tmp.add(d[k]);
                }
                m[i][j] = m[j][i] = tmp.stream().mapToDouble(e -> e).sum();
                tmp.clear();
                k++;
            }
        }
        return m;
    }

    @Override
    public void additionalStatistic(double kvantil, Analysis header) {
        List<AnalysisData> dataList = header.getDataList();

header.setSy(dataList.stream().mapToDouble(AnalysisData::getyLinearDiff2)
                .sum() / (dataList.size() - header.getN() - 1));
        header.settStud(new TDistribution(dataList.size() - (header.getN() +
1))
                .inverseCumulativeProbability(1 - 0.05 / 2    }




    @Override
    public void calculateDistancesByStatisticTests(Analysis header) {
        List<AnalysisData> dataList = header.getDataList();
        for (AnalysisData ad : dataList) {
            double diMalohob = ad.getDiMalohob();
            ad.setChi2Test( diMalohob > header.getChi2() );
            ad.setDi2(diMalohob * diMalohob);
            ad.setTS(diMalohob * header.getCoefficient());
            ad.setFisherTest(ad.getTS() > header.getFisher());
        }

header.setBeta2(dataList.stream().mapToDouble(AnalysisData::getDi2).sum() /
dataList.size());
    }

    @Override
```

```
    public void statisticByChi2AndFisherTest(double alpha, Analysis header) {
        int n = header.getN();
        double size = header.getDataList().size();
        header.setChi2(new ChiSquaredDistribution(n +
1).inverseCumulativeProbability(1 - alpha));
        header.setFisher(new FDistribution(n + 1, size - (n+1))
                .inverseCumulativeProbability(1 - alpha));
        header.setCoefficient((size - (n + 1)) * size / (size * size - 1) /
(n+1));
    }

    @Override
    public void distanceCalculation(double[][] m, Analysis header) {
        int n = header.getN();
        List<AnalysisData> data = header.getDataList();
        for (AnalysisData ad : data) {
            double[] diff = ad.getDiff();
            double[] disElements = new double[n+1];
            double disMal = 0;

            for (int i = 0; i <= n; i++) {
                for (int j = 0; j <= n; j++) {
                    disElements[i] += m[j][i] * diff[j];
                }
                disMal += disElements[i] * diff[i];
            }
            ad.setDistanceCheckElements(disElements);
            ad.setDiMalohob(disMal);
        }
    }

    @Override
    public double[][] getInverseMatrix(double[][] m) {
        RealMatrix inverse = new
LUDecomposition(MatrixUtils.createRealMatrix(m)).getSolver().getInverse();
        return inverse.getData();
    }

    @Override
    public double[][] throwingMatrix(Analysis header) {
        int n = header.getN();
        int size = header.getDataList().size();
        List<double[]> diff2 =
```

```
header.getDataList().stream().map(AnalysisData::getDiff2).collect(Collectors.
toList());
        List<double[]> mElements =
header.getDataList().stream().map(AnalysisData::getMatrixElements).collect(Co
llectors.toList());
        List<Double> tmp = new ArrayList<>();
        double[][] m = new double[n+1][n+1];

        for (int i = 0; i <= n; i++) {
            for (double[] d : diff2) {
                tmp.add(d[i]);
            }
            m[i][i] = tmp.stream().mapToDouble(e -> e).sum() / size;
            tmp.clear();
        }

        int k = 0;
        for (int i = 0; i <= n; i++) {
            for (int j = i+1; j <= n; j++) {
                for (double[] d : mElements) {
                    tmp.add(d[k]);
                }
                m[i][j] = m[j][i] = tmp.stream().mapToDouble(e -> e).sum() /
size ;
                k++;
                tmp.clear();
            }
        }
        return m;
    }

    @Override
    public void calculateMatrixElements(Analysis header) {
        int n = header.getN();
        int size = ( (n + 1) * (n + 1) - (n + 1) ) / 2;
              for (AnalysisData ad : header.getDataList()) {
            double[] diff = ad.getDiff();
            double[] matrixElements = new double[size];
            int k = 0;
            for (int i = 0; i <= n; i++) {
                for (int j = i+1; j <= n; j++) {
                    matrixElements[k] = diff[i] * diff[j];
                    k++;
```

```
                }
            }
            ad.setMatrixElements(matrixElements);
        }
    }

    @Override
    public void baseHeaderCalculation(Analysis analysis) {
        List<AnalysisData> dataList = analysis.getDataList();
        List<double[]> x =
dataList.stream().map(AnalysisData::getX).collect(Collectors.toList());
        List<Double> tmp = new ArrayList<>();
        int n = analysis.getN();
        double[] min = analysis.getMin();
        double[] max = analysis.getMax();
        double[] average = analysis.getAverage();

        min[0] =
dataList.stream().mapToDouble(AnalysisData::getY).min().getAsDouble();
        for (int i = 1; i <= n; i++) {
            for (double[] d : x) {
                tmp.add(d[i - 1]);
            }
            min[i] = Collections.min(tmp);
            tmp.clear();
        }

        max[0] =
dataList.stream().mapToDouble(AnalysisData::getY).max().getAsDouble();
        for (int i = 1; i <= n; i++) {
            for (double[] d : x) {
                tmp.add(d[i - 1]);
            }
            max[i] = Collections.max(tmp);
            tmp.clear();
        }

        average[0] =
dataList.stream().mapToDouble(AnalysisData::getY).average().getAsDouble();
        for (int i = 1; i <= n; i++) {
            for (double[] d : x) {
                tmp.add(d[i - 1]);
            }
```

```
            average[i] = tmp.stream().mapToDouble(e ->
e).average().getAsDouble();
            tmp.clear();
        }

    }

    @Override
    public void statisticHeadCalculate(Analysis header) {
        int n = header.getN();
        double num = header.getDataList().size();
// less casts in calculation
        double[] dispersion = header.getDispersion();
        double[] medianDeviation = header.getMedianDeviation();
        double[] asymmetry = header.getAsymmetry();
        double[] excess = header.getExcess();

        List<double[]> diff2 =
header.getDataList().stream().map(AnalysisData::getDiff2).collect(Collectors.
toList());
        List<double[]> diff3 =
header.getDataList().stream().map(AnalysisData::getDiff3).collect(Collectors.
toList());
        List<double[]> diff4 =
header.getDataList().stream().map(AnalysisData::getDiff4).collect(Collectors.
toList());
        List<Double> tmp = new ArrayList<>();

        for (int i = 0; i <= n; i++) {
            for (double[] d : diff2) {
                tmp.add(d[i]);
            }
            dispersion[i] = tmp.stream().mapToDouble(e -> e).sum() / (num -
1);
            medianDeviation[i] = Math.sqrt(dispersion[i]);
            tmp.clear();
        }

        double nA = num / (num - 1) / (num - 2);
        for (int i = 0; i <= n; i++) {
            for (double[] d : diff3) {
                tmp.add(d[i]);
            }
```

```
            asymmetry[i] = nA * tmp.stream().mapToDouble(e -> e).sum() /
Math.pow(medianDeviation[i], 3);
            tmp.clear();
        }

        double nE = num * (num + 1) / (num - 1) / (num - 2) / (num - 3);
        for (int i = 0; i <= n; i++) {
            for (double[] d : diff4) {
                tmp.add(d[i]);
            }
            excess[i] = nE * tmp.stream().mapToDouble(e -> e).sum() /
Math.pow(medianDeviation[i], 4);
            tmp.clear();
        }
    }


}
```

Додаток В

Опис програми

1 Загальні відомості

Дана розробка призначена для визначення аномалій у взаємозв’язках між частинами застосунків, що розроблені мовою Java, з точки зору об’єктно-орієнтовного проектування.

2 Функціональне призначення

Програма для перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java, дозволяє визначити аномалії у взаємозв'язках між частинами застосунків, забезпечує зберігання результатів визначення, а також надає користувачу швидкий доступ до попередніх результатів. Функціональним призначенням ПЗ є автоматизація процесів:

Програмне забезпечення повинно виконувати наступні функції:

2) Загальні функції:

– імпорт вхідних даних з файлу;

– розрахунок параметрів для вихідних вибірок;

– розрахунок параметрів двовимірного перетворення Джонсона;

– нормалізація вихідних вибірок за допомогою двовимірного нормалізуючого перетворення Джонсона;

– розрахунок параметрів для нормалізованих вибірок;

– перевірка вихідних та нормалізованих вибірок на нормальність розподілу;

– побудова рівняння еліпсу передбачення для нормалізованих даних;

– побудова рівняння трансформованого еліпсу передбачення;

– перевірка взаємозв’язків між частинами застосунків;

– експорт результатів у файл.

3 Опис логічної структури

Програма розроблена на модульній основі, тобто кожний модуль програми відповідає за конкретну дію: імпорт даних, нормалізацію, побудову лінійного та нелінійного рівняння регресії та інші.

4 Технічні та програмні засоби

Система повинна задовольняти наступній мінімальній комплекції:

– CPU: Intel i5-10400 ;
– RAM: 4 GB;
– вільної пам’яті на жорсткому диску не менше 500 Мб.

Система повинна відповідати вимогам до інформаційної та програмної сумісності. Для повноцінного функціонування системи необхідно використовувати операційну систему Windows 7 – 32-bit/64-bit або вище.

5 Виклик та завантаження

Програма для перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java, встановлюється безпосередньо на комп’ютер користувача. Програма відкривається або за допомогою ярлика, який знаходиться на робочому столі, або за допомогою вибору відповідного пункту меню «Пуск» – «Програми». Програма має один режим використання.

6 Вхідні та вихідні дані

Введення даних здійснюється за допомогою пристроїв введення даних (клавіатури та миші). Дані або вводяться вручну, або обираються із представлених списків. Обмін інформацією між ПЗ відбувається за допомогою файлів з довільним доступом з розширенням .csv.

Додаток Г

Інструкція користувача

Програма для перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java, яка розроблялась у рамках кваліфікаційної роботи, запакована в jar-архів під назвою metricsanalysis.jar.

Після виконання команди java -jar metricsanalysis.jar, запускається додаток MetricsAnalysis. Безпосередньо після цього користувач системи повинний пройти авторизацію – ввести в діалоговому вікні, що з'явилося, користувальницьке ім'я та пароль. Пароль є унікальною комбінацією буквено-цифрових символів, відомих тільки користувачу.

Якщо ім’я та пароль введені хибно, вхід в програму не відбудеться. Треба знову повторити попередню операцію. На рис. Д.1 представлена аутентифіцація користувача.

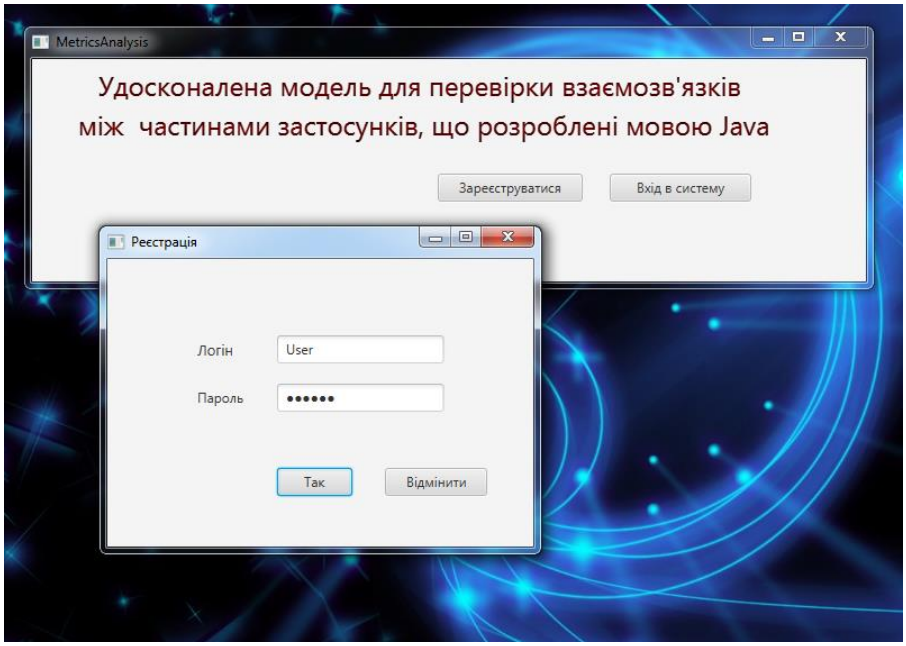

Рисунок Д.1 – Форма авторизації користувача

Після успішної аутентифікації користувача буде відображено головне

вікно програми, де користувач зможе або переглянути попередні результати роботи в системі, або відкрити файл з новими даними. Після успішного відкриття файлу з даними, для користувача доступні функції «Перевірити на нормальність» та «Розрахунок параметрів для вихідних вибірок».

Якщо користувач спочатку вибрав режим «Розрахунок параметрів для вихідних вибірок», а потім натиснув на кнопку «Перевірити на нормальність», тоді відкриється вікно «Перевірка на нормальність», рис. Д.2.

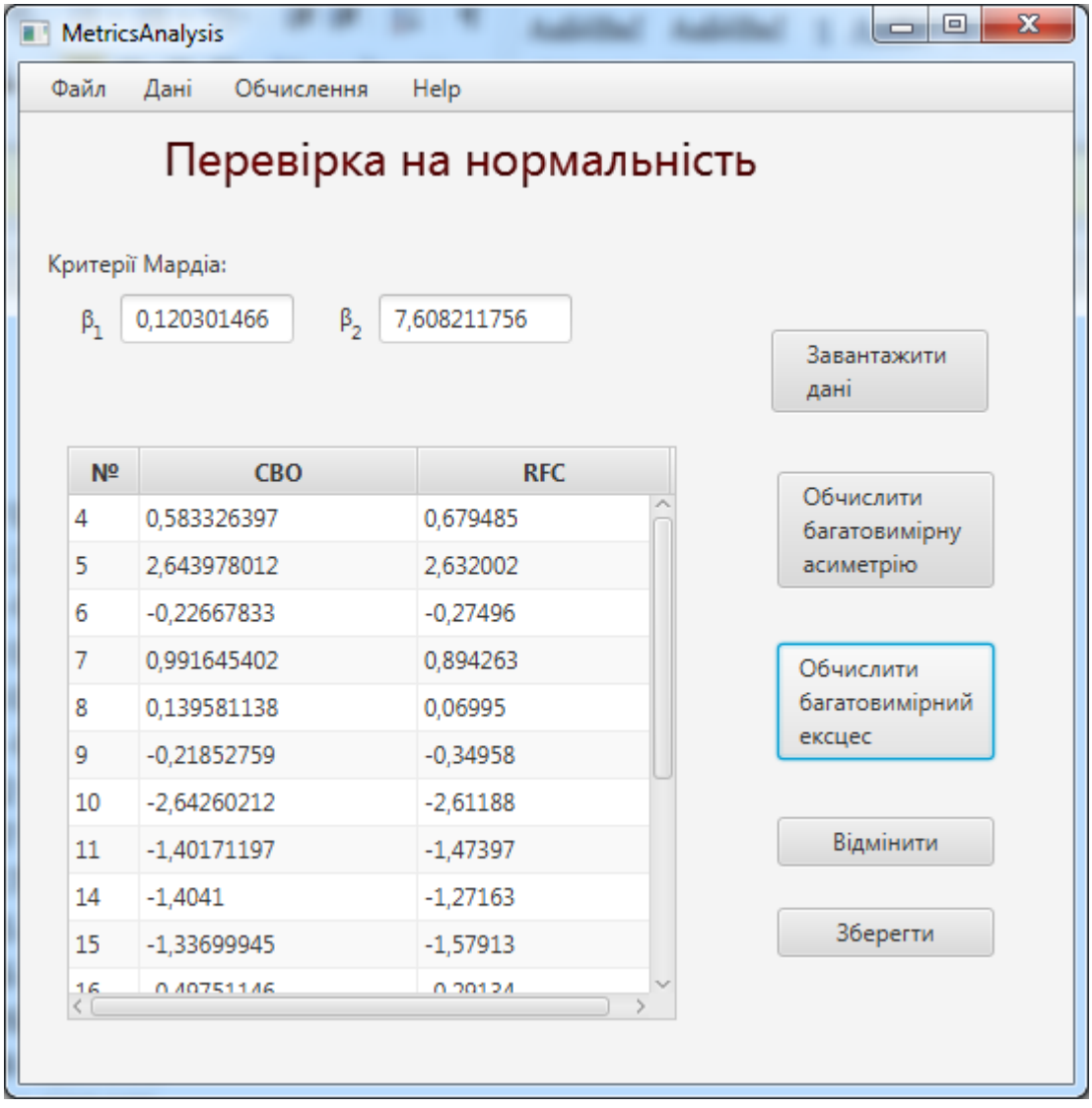

Рисунок Д.2 – Форма «Перевірка на нормальність»

Якщо користувач в режимі «Перевірка на нормальність» натиснув на кнопку «Завантажити дані», відбудеться завантаження даних із файлу, далі

відбувається обчислення параметрів для вихідних вибірок: кількість значень вибірки; вибіркове середнє; дисперсія; середньоквадратичне відхилення; асиметрія; квадрат асиметрії;ексцес; мінімальне значення; максимальне значення. Отже користувачеві вже не потрібно переходити до режиму натискати «Розрахунок параметрів для вихідних вибірок», натискати кнопку «Обчислити». В режимі «Розрахунок параметрів для вихідних вибірок» вже будуть відображатися розрахункові дані.

Для перевірки даних на нормальність користувачеві в режимі «Перевірка на нормальність» потрібно натиснути на кнопку «Обчислити багатовимірну асиметрію», результат буде виведено в тестове поле β_1, потім користувачеві потрібно натиснути на кнопку «Обчислити багатовимірний ексцес», результат буде виведено в тестове поле β_2. На рис. Д.3 представлена форма «Перевірка на нормальність»

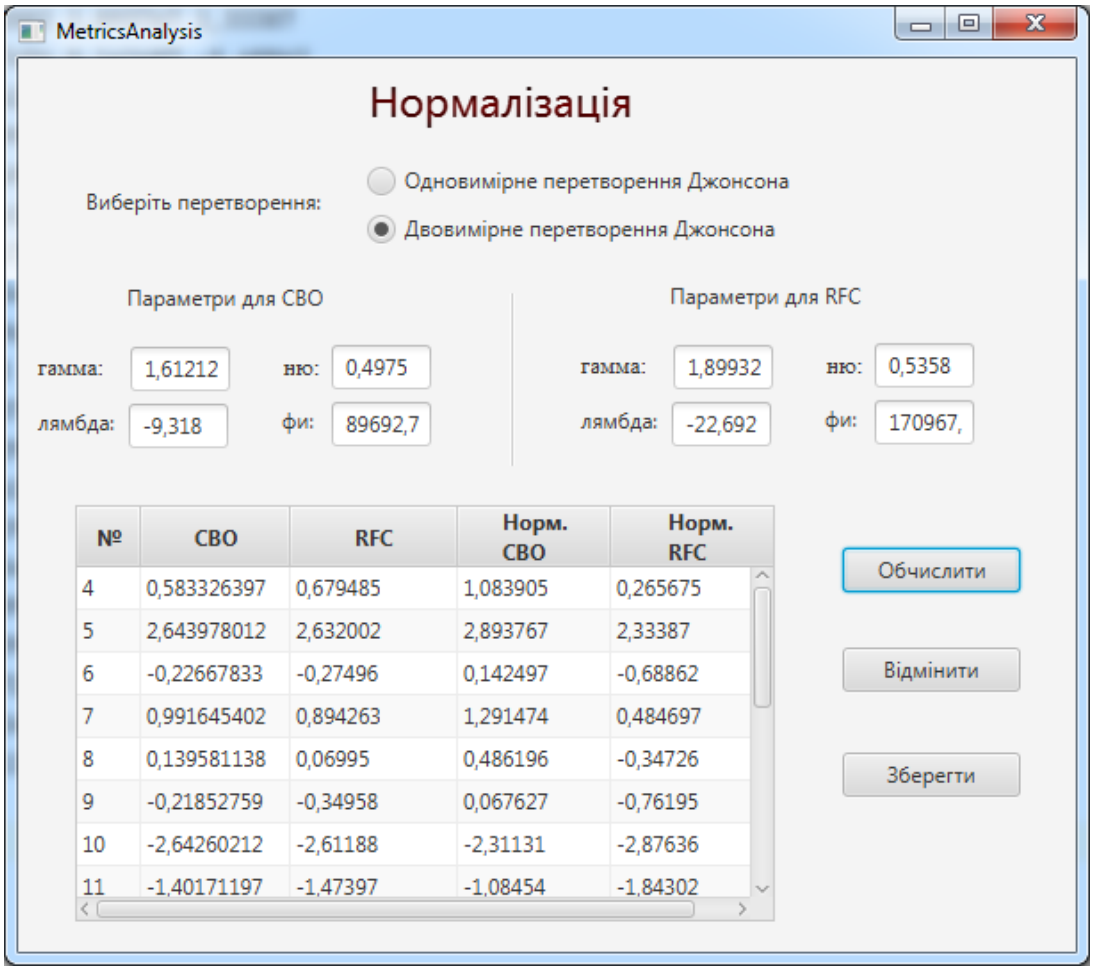

Рисунок Д.3 – Форма «Нормалізація»

Для нормалізації даних користувачеві потрібно вибрати тип перетворення, це відбувається шляхом встановлення відповідного перемикача. Після чого в групах «Параметри для CBO» та «Параметри для RFC» відобразяться відповідні параметри перетворення Джонсона. Після нормалізації, дані треба зберегти, натиснувши на кнопку «Зберегти», рис. Д.3.

В режимі «Аналіз даних на викиди» треба ввести значення значущості (за замовчуванням береться рівний 0,005) та кількість ступенів свободи. Для визначення викидів треба спочатку натиснути кнопку «Обчислити Ts» рис. Д.4.

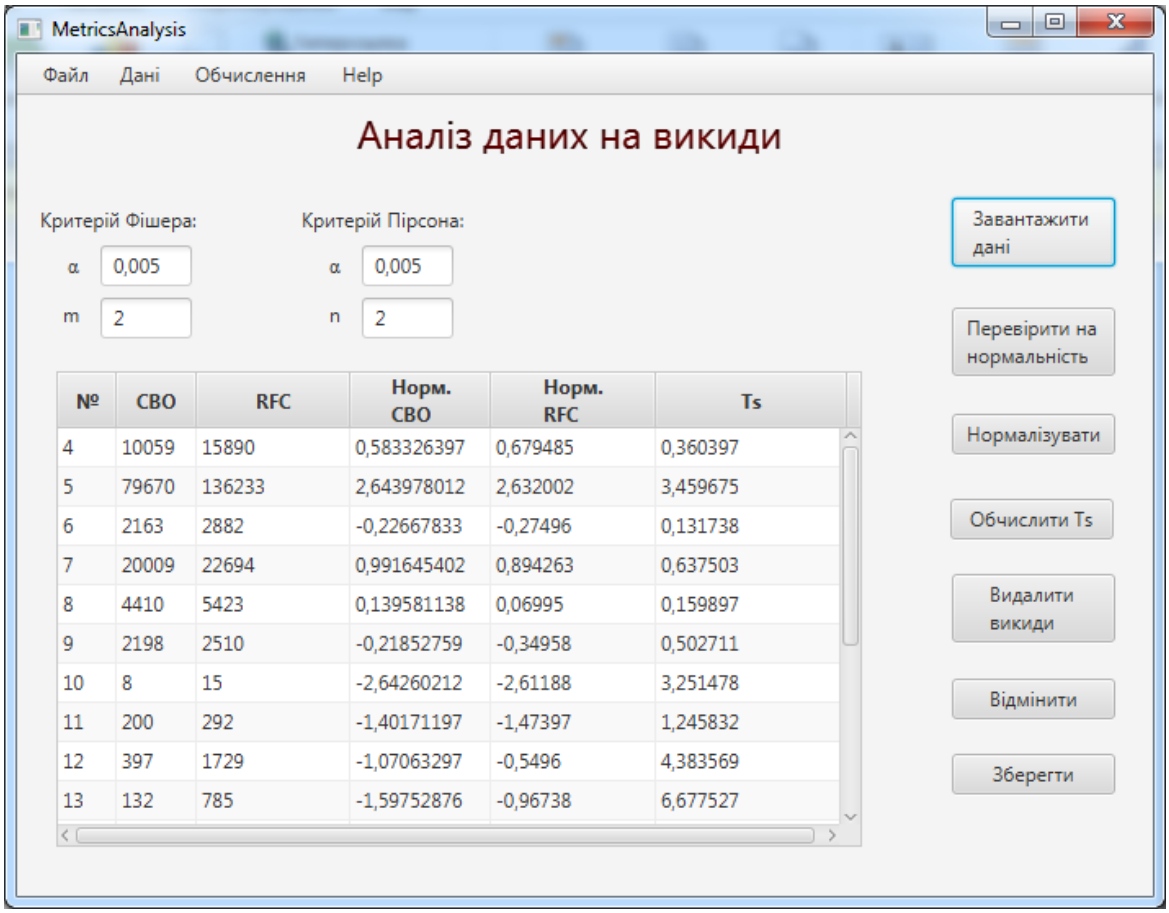

Рисунок Д.4 – Форма «Аналіз даних на викиди»

В режимі «Удосконалена модель для перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java» користувач повинен заповнити поля для вводу, потім потрібно натиснути на кнопку «Перевірка взаємозв’язків».

В робочій частині форми буде побудовано графік трансформованого еліпсу передбачення, де на вісі OX відкладено інтегровані значення RFC, а на вісі OY – CBO. Тестове значення на графіку маркірується червоним кольором. Перевірка взаємозв’язків між частинами застосунків, що розроблені мовою Java, здійснюється емпірично. Якщо тестове значення знаходиться за межами еліпсу, це свідчить про наявність аномалій у відношенням між частинами застосунків з точки зору об’єктно-орієнтовного проектування.

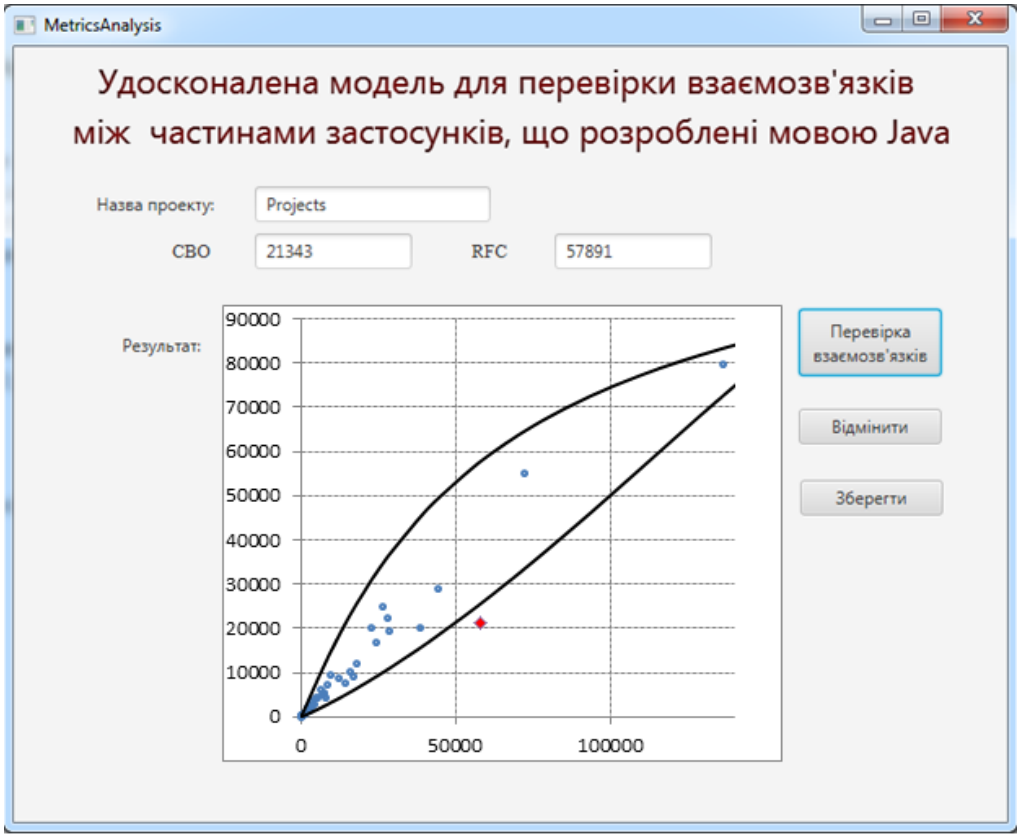

Рисунок Д.5 – Форма «Удосконалена модель для перегляду результатів розрахунку для перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java».

Додаток Д

Програма та методика випробувань ПЗ

1 Об’єкт випробувань

Об’єктом випробовувань є програма для перевірки взаємозв'язків між частинами застосунків, що розроблені мовою Java. Функціонує під управлінням операційної системи MS Windows XP/7/8/10.

2 Мета випробувань

Програма випробувань програмного забезпечення переслідує цілі:

– перевірка придатності розробленого програмного забезпечення до використання;

– перевірка придатності розробленого програмного забезпечення для розв’язання задач у предметній галузі, для якої даний програмний продукт був створений;

– перевірка досягнутих характеристик програмного продукту.

Метою випробувань є відповідність програми вимогам, сформульованим у технічному завданні, наведеному у Додатку А і виявлення помилок проектування. Якщо прийнятий рівень відповідності ПЗ досягнутий, а також відсутні помилки і грубі недоліки, то ПЗ 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.

4 Вимоги до програмної документації

У комплект програмної документації повинні входити наступні документи:

- технічне завдання;
- інструкція користувача;
- текст програми;

– програма й методика випробувань.

5 Порядок та методи випробувань

Випробування будемо проводити за стратегією «чорного ящика». За рахунок дуже великої кількості функцій, які потрібно випробувати, представимо тільки декілька результатів випробувань функцій програми. Всі функції програми будуть проходити випробування.

Складемо програму випробувань:

Випробування можливості введення даних з файлу:

Якщо натиснути кнопку «Вибрати файл», з’явиться вікно, в якому треба обрати шлях до файлу з розширенням .csv. Після натискання кнопки «Вибрати файл» файл повинен додатися.

Випробування роботи функції «Нормалізація даних»:

У вікні «Нормалізація» треба вибрати перемикач «Двовимірне перетворення Джонсона», після цього в текстових полях відобразяться значення параметрів γ, η, ϕ, λ. Після натискання кнопки «Обчислити», значення нормалізованих даних з’являться в таблиці.

Випробування роботи функції «Перевірка взаємозв’язків між частинами застосунків»:

У вікні «Перевірка взаємозв’язків між частинами застосунків» треба ввести значення метрик для проекту з тестової вибірки у поля для вводу. Після натискання кнопки «Перевірка взаємозв’язків», результат перевірки повинен відобразитися в робочій області вікна.