НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ
(повна назва інституту/факультету)

кафедра БІОМЕДИЧНОЇ КІБЕРНЕТИКИ
(повна назва кафедри)

«На правах рукопису»
УДК 004.42

«**До захисту допущено**»
Завідувач кафедри БМК
_____ Є.А. НАстенко
(підпис) (ініціали, прізвище
"___"_____2018р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 122 «Комп'ютерні науки та інформаційні технології»
(код і назва)

на тему: «Оцінка психофізичного стану студентів на заняттях фізичного виховання»

Виконав (-ла): студент (-ка) **VI** курсу, групи БС-61м
(шифр групи)

СЕРБА ЛЕВ ЄВГЕНІЙОВИЧ _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник *зав. каф. БМК, с.н.с., д.б.н. Настенко Є.А.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з розділів МД _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент *доц. каф. БМІ, доц., к.т.н., Зубчук В.І* _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент *доц. каф. ФВ к.п.н Бойко Г.Л.* _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____
(підпис)

Київ – 2018 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут (факультет) БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ
(повна назва)

Кафедра БІОМЕДИЧНОЇ КІБЕРНЕТИКИ
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-науковою програмою
спеціальність 122 «Комп'ютерні науки та інформаційні технології»
(спеціалізація) (Інформаційні технології в біології та медицині)
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри БМК
_____ _Є.А. Настенко_
(підпис) (ініціали, прізвище)
«___»_____2018 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

СЕРБІ ЛЕВУ ЄВГЕНІЙОВИЧУ
(прізвище, ім'я, по батькові)

1. Тема дисертації «Оцінка психофізичного стану студентів на заняттях фізичного виховання»

науковий керівник дисертації
Настенко Євгеній Арнольдович, с.н.с., д.б.н.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «_29_»_березня_ 2018 р. №_1041-с

2. Термін подання студентом дисертації ***11-12 травня 2018 року***
3. Об'єкт дослідження Алгоритм для лінійного регресійного аналізу об'єктів

4. Предмет дослідження Алгоритм лінійної регресії методом найменших квадратів

5. Перелік завдань, які потрібно розробити
Проаналізувати алгоритм лінійної регресії методом найменших квадратів, програмно реалізувати алгоритм, програмно реалізувати роботу з базою даних, проаналізувати результати експериментів

6. Орієнтовний перелік графічного (ілюстративного) матеріалу

Таблиця з результатами розрахунків, таблиця застосованих фільтрів

7. Орієнтовний перелік публікацій 1. Програмна система для регресійного аналізу параметрів артеріального тиску з метою оцінки стану студентів на заняттях фізичного виховання // SIS-journal №17. 2. Задачі кластеризації. Огляд алгоритмів кластерного аналізу даних // Актуальные научные исследования в современном мире // Сб. научных трудов 3. Задачі кластеризації. Порівняння ієрархічних та неієрархічних алгоритмів кластеризації // Перспективні шляхи розвитку наукової думки

8. Консультанти розділів дисертації*

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

9. Дата видачі завдання ***19 березня 2018 р.***

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Отримати завдання на МД	19 березня 2018р..	
2	Огляд літератури	30 березня 2018р.	
3	Побудова структури програмної системи	10 березня 2018р.	
4	Проведення експериментів	10 квітня 2018р.	
5	Написання МД	1травня 2018р.	
6	Предзахист МД та допуск до захисту дисертації	3 травня 2018р..	
7	Подання МД рецензенту. Отримання рецензії.	4-7 травня 2018р.	
8	Подання в електронному вигляді МД та анотації до неї на сайт кафедри.	11-12 травня 2018р.	
9	Подання пакету документів по МД до захисту в ЕК	11-12 травня 2018р.	
10	Захист МД в ЕК	18-19 травня 2018р..	

Студент _____ Л.Є. Серба
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Є.А. Настенко
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено науковго керівника магістерської дисертації.

Нормоконтролер

__*доц. каф. БМК, к.т.н.*__	_____	___О.К. Носовець ___
(посада, науковий ступінь, вчене звання)	(підпис)	(ініціали, прізвище)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 10
ВСТУП 11
РОЗДІЛ 1 ЗАГАЛЬНІ ВІДОМОСТІ З ПРЕДМЕТНОЇ ОБЛАСТІ 14
1.1. Лінійна регресія 15
1.2. Критерій Фішера 16
1.3. Дисперсія 17
1.4. Коефіцієнт кореляції 21
1.5. Регресійна залежність 23
1.6. Лінійна регресія та метод найменших квадратів 24
1.7. Кластерний аналіз даних 28
1.8. Кластерний аналіз методом k-середніх 30
Висновки до розділу 1 31
РОЗДІЛ 2 РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ 31
2.1. Програмні засоби, які були використані для реалізації 31
2.2. Проектування програмного продукту 46
2.2.1. Модель життєвого циклу 46
2.2.2. Побудова ієрархічної структури та розрахунок нев’язки 59
2.2.3. Діаграма сутність-зв’язок (ERD) 61
2.2.4. Контекстна діаграма IDEF0 63
2.2.5. Діаграма декомпозиції IDEF0 65
2.2.6. Backlog 67
2.2.7. Методологія IDEF3 70
2.2.8. Методологія DFD 71
2.2.9. Діаграма класів 75
2.2.10. Діаграми реалізації 76
2.2.11. Діаграма діяльності 77
2.2.12. Діаграма дерева вузлів 78
2.2.13. Діаграма варіантів 79

2.3. Реалізація програмного продукту 80
Висновки до розділу 2 81
РОЗДІЛ 3 РОБОТА З РОЗРОБЛЕНИМ ПРОГРАМНИМ ПРОДУКТОМ .. 82
Висновки до розділу 3 87
РОЗДІЛ 4 РЕЗУЛЬТАТИ СТАТИСТИЧНИХ ЕКСПЕРИМЕНТІВ 88
Висновки до розділу 4 100
РОЗДІЛ 5 СТАРТАП-ПРОЕКТ 101
Висновки до розділу 5 109
ЗАГАЛЬНІ ВИСНОВКИ 110
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 111
Додаток А 115

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

АТ – артеріальний тиск.

ЧСС – частота серцевих скорочень

ПЗ – програмне забезпечення

ПП – програмний продукт

DFD – Data Flow Diagram

ВСТУП

Актуальність теми.

Дослідження стану студентів на заняттях фізичного виховання грає важливу роль у моніторингу стану здоров'я та завчасному запобіганні утворенню проблем серцево-судинної системи в молодому віці.

Програмна система для регресійного аналізу параметрів артеріального тиску, що були отримані з різних ділянок тіла дозволить встановити та проаналізувати взаємозв'язок показників тиску на лівому плечі та інших ділянках верхніх та нижніх кінцівок в стані спокою та одразу після фізичного навантаження.

Актуальність регресійного аналізу полягає в тому, що за допомогою нього можна побудувати математичну модель і визначити її статичну надійність. Даний вид аналізу є одним з найбільш поширених методів обробки спостережень при вивченні залежностей у фізиці, біології, економіці, техніці та інших областях.

Мета і завдання дослідження.

Розробити програмну систему для регресійного аналізу параметрів артеріального тиску з метою оцінки стану студентів на заняттях фізичного виховання.

У відповідності до мети сформовано наступні задачі:

- провести аналіз алгоритмів лінійної регресії;
- створити програмну систему для регресійного аналізу параметрів артеріального тиску;
- програмно реалізувати алгоритм лінійної регресії методом найменших квадратів;
- програмно реалізувати роботу з базою даних у форматі .xls;
- програмно реалізувати механізм фільтрації з метою створення вибірки вхідних даних для аналізу;

- проаналізувати результати роботи програми.

Об’єкт дослідження – алгоритми для лінійного регресійного аналізу об’єктів.

Предмет дослідження – алгоритм лінійної регресії методом найменших квадратів.

Для досягнення мети дослідження та реалізації програмної системи використано IntelliJ IDEA 2017 community edition з використанням мови програмування Java.

Наукова новизна одержаних результатів.

Дістало подальший розвиток дослідження взаємозв’язку параметрів артеріального тиску в точці на лівому печі та інших точках нижніх та верхніх кінцівок. Вперше досліджена зміна кореляції параметрів тиску в залежності від фізичного навантаження.

Практичне значення одержаних результатів.

Дослідження та розроблена програмна система може використовуватися для дослідження та моніторингу стану серцево-судинної системи людини при фізичних навантаженнях.

Реалізація результатів роботи.

Дисертація виконана на замовлення кафедри фізичного виховання факультету біомедичної інженерії НТУУ «КПІ» імені Ігоря Сікорського. Одержані результати дослідження впроваджені в навчальний процес кафедри фізичного виховання факультету біомедичної інженерії НТУУ «КПІ» імені Ігоря Сікорського.

Апробація результатів роботи.

Основні положення та результати магістерської дисертації були викладені в наступних роботах:

1. Серба Л.Є. Програмна система для регресійного аналізу параметрів артеріального тиску з метою оцінки стану студентів на заняттях фізичного виховання / Серба Л.Є. // SIS-journal № 17.

2. Серба Л.Є. Задачі кластеризації. Огляд алгоритмів кластерного аналізу даних / Серба Л.Є. // Актуальные научные исследования в современном мире. - Сб. научных трудов - Переяслав- мельницкий. - 2017. - Вып. 12(32). - ч. 7. - С.23-28

3. Задачі кластеризації. Порівняння ієрархічних та неієрархічних алгоритмів кластеризації // Теорія і практика наукових знань (частина IV): матеріали II Міжнародної науково-практичної конференції м. Київ, 28-29 грудня 2017 року. – Київ.: МЦНД, 2017. С. 20-25.

Структура дисертації

Дисертація побудована за класичним типом та викладена на 119 сторінках машинописного тексту. Складається з вступу, 5 розділів, висновків, списку використаних літературних джерел, який містить 40 найменувань, 25 – на кирилиці, 15 – на латиниці. У роботі представлено 56 рисунка і 6 таблиць.

РОЗДІЛ 1
ЗАГАЛЬНІ ВІДОМОСТІ З ПРЕДМЕТНОЇ ОБЛАСТІ

У кровоносній системі живого організму циркуляція крові здійснюється узгодженими діями серця і судин, що забезпечує адаптацію до умов життєдіяльності в межах гомеостатичних кордонів. Серед інтегральних характеристик серцево-судинної системи найбільш доступною і важливою характеристикою є величина артеріалиюю тиску (АТ), котра залежить від об'єму циркулюючої крові, роботи серця і судин (периферичного судинного опору, еластичності судин), а також реологічних властивостей крові. Величина тиску виражається декількома показниками: систолічним тиском S (максимальний тиск при скороченні серцевого м'яза), тиском діастоли D (мінімальний тиск між ударами серця в момент розслаблення серцевого м'яза), а також пульсовим тиском W (за визначенням: W = S - D). Незважаючи на великі коливання внутрішньосудинного тиску під час систоли і діастоли, кровотік здійснюється при стійкому режимі тиску, який називається середнім гемодинамічним тиском. Таким чином, величина тиску містить дві компоненти - постійний, що характеризується величиною середнього гемодинамічного тиску M, і пульсуючий, який характеризується величиною W. Вважається, що M відображає енергію безперервного руху крові, залежить від скоротливої функції серця і загального периферичного судинного опору. Тому ця величина може використовуватися для оцінки компенсаторних можливостей кровообігу [1]. Середній тиск M обчислюється зазвичай за найбільш популярною формулою Хікема: M = D + (S - D) / 3. Пульсовий тиск W містить в собі результат взаємодії скорочувальної функції серця, розтяжності артерій і величини хвилі відображення [2].

Фізичне навантаження має досить суттєвий вплив на кровообіг. Через збільшення кровотоку через м'язи під навантаженням, кількість

крові, накачаної серцем, може збільшитися в чотири рази, а у елітних спортсменів в шість разів. Збільшення об'єму крові, викинутої при кожному серцевому ритмі, може призвести до збільшення систолічного артеріального тиску до 180 мм рт. Однак через те, що кров тече дуже швидко з артерій, особливо для робочого м'язу, де судини опору розширені, діастолічний тиск залишається відносно незмінним, або навіть може зменшуватися. Ізометричні вправи мають зовсім інший ефект через набагато менший вплив на загальну кількість крові, що накачується серцем, але рефлекси, особливо ті що виникають через скорочення м'язів, змушують кровоносні судини в іншому місці стискатися, а отже, і систолічний, і діастолічний артеріальний тиск різко зростають [3]. Також варто зазначити, що показники тиску можуть відрізнятися в залежності від місця, на якому проводилося вимірювання. За референтне значення тиску прийнято вважати значення, отримане на лівому плечі. З метою встановлення взаємозв`язку між показниками тиску у референтній точці та показниками тиску в будь якій іншій точці тіла доцільно провести лінійний регресійний аналіз.

1.1. Лінійна регресія

Якщо дано сукупність показників у, що залежать від факторів х, то постає завдання знайти таку економетричну модель, яка б найкраще описувала існуючу залежність. Одним з методів є лінійна регресія. Лін ійна регресія передбачає побудову такої прямої лінії, при якій значення показників, що лежать на ній будуть максимально наближені до фактичних, і продовжуючи цю пряму одержуємо значення прогнозу. Процес продовження прямої називається екстраполяцією. Відповідно до цього постає задача визначити цю пряму, тобто рівняння цієї прямої. В загальному вигляді рівняння прямої виглядає:

$$y` = a + bx. \quad (1.1)$$

де y` - вирівняне значення y для відповідного значення x. Константи a і b - константи, які передбачають зменшення суми квадратів відхилень між фактичним значенням y і вирівняним значенням y`.

$$\Sigma(y - y`)^2 \rightarrow \min \quad (1.2)$$

Коефіцієнт a характеризує точку перетину прямої регресії з лінією координат. Коефіцієнт b характеризує кут нахилу цієї прямої до осі абсцис, а також на яку величину зміниться y` при зміні x на одиницю. Коефіцієнти a і b знаходять із системи рівнянь, що випливає з формули (1.2).

Знайшовши значення параметрів розраховують ряд вирівняних значень для відповідних факторів і проводять дослідження знайденої економетричної моделі [6]. Щоб зробити висновок про доцільність використання знайденої моделі проводять аналіз за наступними напрямками:

1. Розраховують критерій Фішера та перевіряють знайдену модель на адекватність вихідним даним.
2. Розраховують і аналізують дисперсію показників.
3. Розраховують і аналізують коефіцієнт кореляції.
4. Розраховують та аналізують коефіцієнт еластичності.
5. Розраховують довірчий інтервал для прогнозованих показників.

1.2. Критерій Фішера

F-тестом або критерієм Фішера (F-критерієм, φ*-критерієм) — називають будь-який статистичний критерій, тестова статистика якого при виконанні нульової гіпотези має розподіл Фішера (F-розподіл) [8].

Статистика тесту так чи інакше зводиться до відношення вибіркових дисперсій (сум квадратів, ділених на «ступеня свободи»). Щоб статистика мала розподіл Фішера, необхідно, щоб чисельник і знаменник були незалежними випадковими величинами і відповідні суми квадратів мали розподіл Хі квадрат. Для цього потрібно, щоб дані мали нормальний розподіл. Крім того, передбачається, що дисперсія випадкових величин, квадрати яких підсумовуються, однакова. Тест проводиться шляхом порівняння значення статистики з критичним значенням відповідного розподілу Фішера при заданому рівні значимості. Для оцінки знайденої моделі на адекватність порівнюють розрахункове значення критерію Фішера із табличним.

1.3. Дисперсія

У статистиці дисперсія — міра розсіяння (розкиду) числових даних у вибірці. Типові приклади мір статистичної дисперсії — це розмах, дисперсія, стандартне відхилення. Поряд з мірами центру розподілу статистична дисперсія є однією з найпоширеніших і найвикористовуваніших характеристик розподілу даних. Міра статистичної дисперсії — невід'ємне дійсне число, яке дорівнює нулю, якщо всі дані однакові, та зростає, коли розкид даних зростає. У більшості мір дисперсії мають однакові одиниці вимірювання з одиницями вимірювання величин. Іншими словами, якщо вимірюють в метрах або секундах, то це і є одиниця вимірювання дисперсії. Такі міри дисперсії включають в себе:

1. Стандартне відхилення.
2. Розмах.
3. Дисперсія.
4. Середнє абсолютне відхилення.
5. Медіана абсолютного відхилення.

6. Абсолютне відхилення.

7. Відстань стандартного відхилення.

Це часто використовується (разом з коефіцієнтами пропорційності) в ролі оцінки масштабу. Всі вище наведені заходи статистичної дисперсії мають корисну властивість — їхнє розміщення інваріантне. Таким чином якщо випадкова величина X має дисперсію SX то лінійне перетворення $Y = aX + b$ для a , b дійсних коефіцієнтів дисперсія буде $SY = |a|SX$.

Інші міри дисперсії безрозмірнісні (величини з розмірністю одиниця). До них відносяться:

1. Коефіцієнт варіації.

2. Квантиль коефіцієнта дисперсії.

3. Відносна середня різниця, рівна подвоєному коефіцієнту.

Є інші міри дисперсії:

1. Змінна (квадрат стандартного відхилення) — інваріант місцезнаходження, але не в лінійному масштабі.

2. Відношення середнього відхилення — в основному використовується для підрахунку даних.

Деякі міри дисперсії мають спеціальні цілі, в тому числі дисперсія Алана та Адамара [7-9].

У фізиці така мінливість може бути результатом випадкових похибок вимірювання: інструмент вимірювання не абсолютно точний, тому є додатковий оцінювач мінливості в інтерпретації та представлення результатів вимірювань. Можна припустити, що вимірювана величина є стабільною, і що відмінності між результатами стаються черезпохибки спостережень. Система великої кількості частинок характеризується середнім значенням відносно невеликих чисел макроскопічних величин, таких як температура, енергія та щільність.

Стандартне відхилення є важливим заходом в теорії коливності, яка пояснює багато фізичних явищ, в тому числі, чому небо синє. У галузі біологічних наук, вимірювані величини рідко незмінні та стабільні, і

варіація може бути характерним явищем: це може бути пов'язано з мінливість, тобто різниці елементів множини. В економіці, фінансах та інших дисциплін, регресійний аналіз намагається пояснити дисперсію залежної змінної, як правило, оцінюється її дисперсія, використовуючи одну або кілька незалежних змінних, кожна з яких має позитивну дисперсію. Частину пояснення дисперсії називають коефіцієнтом визначення.

Середнє розповсюдження є перехід від одного розподілу ймовірностей до іншого, шляхом поширення однієї або більше частин щільності ймовірності, залишаючи середнє очікуване значення без змін. Поняття середнього розповсюдження забезпечує часткове упорядкування імовірнісних розподілів згідно з їх дисперсією: з двох розподілів ймовірностей можна оцінити їх за величиною дисперсії. Дисперсія в лінійній регресії дає можливість визначити значимість характеристик, вирахуваних в регресійному аналізі (характеристики a і b). Для визначення цих характеристик використовують:

1. Загальна дисперсія - характеризує рівень відхилень між фактичними значеннями ряду і їх середнім значенням.

2. Дисперсія, що пояснюється регресією. Чим більша доля дисперсії, що пояснюється регресією в загальній дисперсії, тим тісніший зв`язок між y і x. Чим ця доля менша, тим відповідно слабший зв`язок. Ця дисперсія визначається, як сума квадратів відхилень між вирівняним значенням ряду і середнім значенням ряду.

3. Залишкова дисперсія - це та частина ЗД, яка не пояснюється регресією.

Невизначенність вимірювання (англ. measurement uncertainty) — параметр, що пов'язаний з результатом вимірювання та характеризує розсіяння значень, які обґрунтовано могли бути приписані вимірюваній величині. Оскільки на практиці вимірюваній величині приписуються значення, отримані в результаті вимірювання (результати вимірювання), то

вказаний параметр характеризує розсіяння результатів вимірювання. Сам термін «невизначеність» означає сумнів в чомусь.

Відносно результату вимірювання термін «невизначеність» означає сумнів у достовірності результату вимірювання. Цей сумнів виникає у зв'язку з тим, що, як правило, завжди невідомо, наскільки результат вимірювання близький до значення фізичної величини. Результат вимірювання величини завжди лише наближено дорівнює її значенню. Таким чином, значення величини точно невизначене. Звідси й термін — «невизначеність».

Отже, говорячи про невизначеність вимірювання, ми підкреслюємо те, що результат вимірювання і значення фізичної величини — це різні речі, а також те, що нам невідомо, наскільки результат вимірювання фізичної величини близький до її значення. Фактично в основу концепції «невизначеності вимірювання» неявно покладенийпостулат, що результат вимірювання — випадкова величина, адже розсіюватися, тобто набувати різних значень, можуть лише випадкові величини. Це підтверджується всією практикою вимірювань, яка свідчить, що вимірюючи одну і ту ж величину, можна отримати різні значення.

З урахуванням зазначеного цілком логічним є аналіз результатів вимірювання в рамках теорії ймовірностей з використанням параметрів, що характеризують розсіяння, прийнятих в цій математичній теорії. Однак в концепції «невизначеності вимірювання» ці параметри отримали інші назви. Такими параметрами можуть бути стандартна невизначеність (сумарна стандартна невизначеність) аборозширена невизначеність. В 1978 р. найвищий світовий авторитет в сфері метрології — Міжнародний комітет мір і ваг (МКМВ) — звернув увагу на відсутність на міжнародному рівні єдності з питання оцінювання точності результатів вимірювань. На його прохання Міжнародне бюро мір і ваг (МБМВ) розіслало анкети з детальними питаннями в 32 національні метрологічні лабораторії та (з інформаційною метою) до п'яти міжнародних організацій.

На початок 1979 р. були отримані відповіді із 21 лабораторії. Майже всі вважали, що важливо прийняти на міжнародному рівні методику оцінювання точності результатів вимірювань. Однак згоди щодо методу оцінювання не існувало. Тоді МБМВ організувало зустріч з метою прийняти єдину загальновизнану методику для оцінки точності, в роботі якої взяли участь експерти 11 національних метрологічних лабораторій. Робоча група розробила Рекомендацію INC −1 (1980) «Подача експериментальних невизначеностей». Рекомендація була прийнята МКМВ в 1981 році та знову затверджена в 1986 р. Задачу розроблення детального документу, який би спирався на принципи Рекомендації (яка була швидше декларацією, а не деталізованим описом), МКМВ передав Міжнародній організації зі стандартизації (ISO).

Розв'язання цієї задачі було покладено на Технічну консультативну групу ISO з метрології (ТКГ 4). ТКГ 4, в свою чергу, створила Робочу групу 3, до складу якої ввійшли експерти, запропоновані МБМВ, Міжнародною електротехнічною комісією (МЕК), ISO, Міжнародною організацією законодавчої метрології (МОЗМ) та затверджені головою ТКГ 4. Перед Робочою групою було поставлено наступне завдання — розробити настановчий документ, який би спирався на принципи INC −1 та давав правила вираження невизначеності вимірювання, котрі використовувались би службами стандартизації, калібрування, акредитації лабораторій та метрології.

1.4. Коефіцієнт кореляції

У статистиці залежність або пов'язаність є будь-яким статистичним відношенням, чи каузальним, чи ні, між двома випадковими величинами або біваріантними даними.Кореляція будь-яким з широкого класу статистичним відношенням, де є залежність, хоча зазвичай про кореляцію говорять тоді, коли дві величини перебувають між собою у

лінійному відношенні. При цьому, зміна однієї або кількох цих величин призводить до систематичної зміни іншої або інших величин. Знайомими прикладами залежних феноменів є кореляція між фізичними параметрами батьків та їхніх дітей і кореляція міжпопитом на товар і його ціною [6].

Користь кореляцій у тому, що вони можуть вказувати на відношення, яке може носити передбачальний характер і тому мати практичне застосування. Наприклад, електрогенеруюча компанія може виробляти менше електрики у періоди з хорошою погодою, базуючись на кореляції між попитом на електрику та погодою. У цьому випадку існує причинно-наслідковий зв'язок, тому що в екстремальну погоду люди використовують більше електрики для опалювання або охолодження. Однак зазвичай самої лише наявності кореляції недостатньо для того, щоб зробити висновок про наявність причинно-наслідкового зв'язку (що часто формулюють фразою "Кореляція не означає причинності".

Кореляція може бути позитивною та негативною (можлива також ситуація відсутності статистичного зв'язку — наприклад, для незалежних випадкових величин). Від'ємна кореляція — кореляція, при якій збільшення однієї змінної пов'язане зі зменшенням іншої, при цьому коефіцієнт кореляції від'ємний. Додатна кореляція — кореляція, при якій збільшення однієї змінної пов'язане зі збільшенням іншої, при цьому коефіцієнт кореляції додатній.

Коефіцієнт кореляції r – міра тісноти зв`язку. Він на відміну від дисперсії характеризує міру тісноти зв`язку (дає її числове значення). Змінюється в межах від -1 до +1. Якщо r=0, то лінія регресії паралельна осі абсцис, тобто залежності між y і t немає (регресія відсутня). Якщо $r \to +1$ (додатна регресія). Із збільшенням t – y_t теж буде зростати. Якщо $r \to -1$ (від`ємна регресія). Із збільшенням t – y_t буде зменшуватись. Коефіцієнт кореляції визначається як корінь квадратний з коефіцієнта детермінації r^2.

1.5. Регресійна залежність

Метою будь-якого дослідження, що здійснюється в даний час, є використання його результатів в майбутньому, або, інакше кажучи, прогнозування стану явища, що вивчається. Приклади такого прогнозування наведені в підручниках всіх природничо-наукових і економічних дисциплін. При цьому, бажаючи вивчати явище у взаємозв'язку з іншими явищами або величинами, доводиться виділяти деякі з них, що впливають на те, що вивчається, оцінювати міру і «якість» впливу, тобто характер зв'язку між тим, що вивчається (основним в даному дослідженні) і величинами якісного або кількісного характеру, що впливають на нього [9].

Надалі «основну», таку, що вивчається, величину називатимемо залежною змінною і позначатимемо літерою Y, інші, впливаючі на Y, величини називатимемо незалежними змінними і позначатимемо літерами x1, x2,…, xk. Як Y, так іх1, x2, …, xk, вважатимемо числовими.

Розрізняють два види зв'язків. Якщо значення залежної змінної стає відомим, як тільки відомі значення незалежних змінних, то зв'язок є динамічним або функціональним, оскільки в цьому випадку існує закон, за яким обчислюється Y залежно від x1, x2, …, xk, Y = f(x1, x2, …, xk). Приклади таких зв'язків: закон вільного падіння тіла; закон Ома; закон Бойля - Маріотта; зв'язок між вартістю одиниці товару і ціною, сплаченою за його партію; залежність продуктивності праці і витрат робочого часу.

Зовсім інакше, коли за значеннями незалежних величин можна встановити лише деяку «середню» тенденцію в значеннях залежної змінної.

Так, наприклад, зрозуміло, що між зростанням людини і її вагою існує залежність, створені таблиці такої залежності, що враховують ще зріст, і вік, проте користуватися ними можна лише, знову ж таки, «в

середньому». Подібного роду зв'язки називають кореляційними (від лат. слова correlatio – співвідношення), а задачею встановлення математичної форми кореляційного зв'язку займається регресійний аналіз. Залежна змінна в при цьому розглядається як випадкова величина, а незалежні змінні можна прямо або опосередковано контролювати. Кореляційний аналіз вивчає спільний розподіл всіх змінних, що вимірюються з аналізом точності оцінювання одних величин через інші.

На відміну від функціонального зв'язку в регресійному аналізі йдеться про встановлення функції регресії. Оскільки незалежні змінні x1, x2, …, xk є контрольованими та керованими, а Y – випадковою величиною, то за даними експерименту, в якому x1, x2, …, xk набули конкретних значень, можна судити лише про оцінку параметра, пов'язаного з розподілом Y. А оцінок же, як правило, можна побудувати багато.

З точки зору подальших використань бажано мати оцінку якомога простішого вигляду і яка задовольняла б деякий критерій оптимальності (подібний до незміщеності, наприклад, для оцінок параметрів). Із всіх елементарних функцій (виключаючи константу) найбільш простою є лінійна, цей випадок розглянемо надалі детально, як найбільш прозорий з ідейної точки зору і такий, що в той же час дає можливість для подальших узагальнень.

1.6. Лінійна регресія та метод найменших квадратів

Моделі лінійної регресії часто встановлюються з використанням методу найменших квадратів. Метод найменших квадратів є стандартним підходом в регресійному аналізі для знаходження наближеного розв'язку надлишково-визначеної системи, тобто безлічі рівнянь, в яких є більше рівнянь, ніж невідомих. «Найменші квадрати» означають, що загальне

рішення мінімізує суму квадратів залишків, отриманих в результаті кожного окремого рівняння [6].

Найбільш важливим застосуванням є збір даних. Найкраща підгонка в методі найменших квадратів зводить до мінімуму суму квадратів залишків (залишкова складова: різниця між спостережуваним значенням і встановленим значенням, наданим моделлю). Коли проблема має істотну невизначеність в незалежній змінній (змінна x), то прості методи регресії і найменших квадратів мають проблеми; в таких випадках замість схеми найменших квадратів може бути розглянута методологія, необхідна для установки моделей помилок в змінних.

Проблеми з найменшими квадратами діляться на дві категорії: лінійні або звичайні найменші квадрати і нелінійні найменші квадрати, в залежності від того, чи є залишки лінійними по всім невідомим. Лінійна проблема найменших квадратів виникає при статистичному регресійному аналізі; він має замкнуте рішення. Нелінійна задача зазвичай вирішується шляхом ітеративного уточнення; на кожній ітерації система апроксимується лінійною [7].

Поліноміальні найменші квадрати описують дисперсію передбачення залежної змінної як функцію незалежної змінної і відхилення від встановленої кривої.

Стандартні моделі лінійної регресії зі стандартними методами оцінки роблять ряд припущень щодо змінних-предикторів, змінних відповіді і їх взаємозв'язку. Існують численні способи, що дозволяють пом'якшити кожне з цих припущень (тобто звести до більш слабкої форми), а в деяких випадках повністю виключити. Як правило, ці способи роблять процедуру оцінки більш складною і трудомісткою, а також можуть вимагати більше даних для створення однаково точної моделі [8].

Нижче наводяться основні допущення, зроблені стандартними лінійними регресійними моделями зі стандартними методами оцінки (наприклад, звичайними найменшими квадратами):

Слабка екзогенність. Це по суті означає, що предикторні змінні x можна розглядати як фіксовані значення, а не випадкові. Це означає, наприклад, що предикторні змінні вважаються безпомилковими, тобто не забруднені помилками вимірювання.

Лінійність. Це означає, що середнє значення змінної відповіді являє собою лінійну комбінацію параметрів (коефіцієнтів регресії) і змінних-предикторів. Оскільки змінні предиктора розглядаються як фіксовані значення, лінійність насправді є лише обмеженням параметрів. Самі змінні предиктора можуть бути довільно перетворені, і насправді можна додати кілька копій однієї і тієї ж основної передбачуваної змінної, кожна з яких перетворюється по-різному. Цей метод використовується, наприклад, в поліноміальній регресії, яка використовує лінійну регресію для відповідності змінної відгуку як довільної поліноміальної функції (аж до заданого рангу) предикторної змінної. Фактично, такі моделі, як поліноміальна регресія, часто «занадто сильні», оскільки вони схильні перевантажувати дані. В результаті, як правило, якась регуляризація зазвичай використовується для запобігання необгрунтованих рішень, що виходять з процесу оцінки.

Постійна дисперсія (гомоскедастичність). Це означає, що різні значення змінної відповіді мають однакову дисперсію в своїх помилках, незалежно від значень предикторних змінних. На практиці це припущення є недійсним (тобто помилки є гетероскедастичнимими), якщо змінна відповіді може варіюватися в широких масштабах. Щоб перевірити гетерогенну дисперсію помилок розумно шукати «ефект роздування» між залишковою помилкою і прогнозованими значеннями [9].

Прості методи оцінки лінійної регресії дають менш точні оцінки параметрів. Однак різні методи оцінки можуть обробляти гетероскедастічність в досить загальному вигляді. Байєсовські методи лінійної регресії можуть також використовуватися, коли передбачається, що дисперсія є функцією середнього. У деяких випадках також можливо

усунути проблему, застосувавши перетворення до змінної відповіді (наприклад, встановити логарифм змінної відповіді з використанням моделі лінійної регресії. Це означає, що змінна відповіді має логарифмічно нормальний розподіл, а не нормальний розподіл).

Незалежність помилок. Це передбачає, що помилки змінних відповіді некорельовані один з одним. Деякі методи (наприклад, узагальнені найменші квадрати) здатні обробляти корельовані помилки, хоча зазвичай вони вимагають значно більшої кількості даних, якщо будь-яка регуляризація не використовується для зсуву моделі в сторону допущення некорельованих помилок. Байєсова лінійна регресія - це загальний спосіб вирішення цієї проблеми.

Відсутність досконалої мультиколінеарності в провісників. Для стандартних методів оцінки найменших квадратів матриця проектування X повинна мати повний ранг стовпчика p; в іншому випадку може спостерігатися досконала мультиколінеарність в предикторних змінних. Це може бути викликано наявністю двох або більше ідеально корельованих предикторних змінних (наприклад, якщо одна і та ж предикторна змінна помилково задається двічі). Це також може статися, якщо є занадто мало даних в порівнянні з кількістю оцінюваних параметрів (наприклад, менша кількість точок даних, ніж коефіцієнти регресії). У разі досконалої багатоколінеарності вектор параметрів β буде неідентифікованим - він не має однозначної відповіді. У кращому випадку ми зможемо визначити деякі параметри, тобто звузити його значення до деякого лінійного підпростору. Варто звернути увагу, що більш складні обчислювальні алгоритми для оцінки параметрів, наприклад ті, що використовуються в узагальнених лінійних моделях, не страждають від цієї проблеми [10].

1.7 Кластерний аналіз даних

Кластеризація - це розбиття множини об'єктів на деякі однорідні підмножини (кластери), параметри яких спочатку невідомі. Для конкретного завдання кількість кластерів може бути довільною або фіксованою. Для кластера характерні внутрішня однорідність (об'єкти одного класу схожі між собою за певними ознаками) і зовнішня ізольованість (об'єкти різних класів суттєво відрізняються) [8].

Нехай є набір даних Xn = {x1,..., xn} X (n> 0) і функція, яка визначає ступінь подібності об'єктів, в більшості випадків це функція відстані між об'єктами ρ (xi, xj).

Потрібно розбити послідовність Xn на непересічні підмножини (кластери) так, щоб кожен кластер складався з об'єктів, близьких за метрикою ρ, а об'єкти різних кластерів істотно відрізнялися. Алгоритм кластеризації - це функція A: X → Y яка будь-якого об'єкта x X ставить у відповідність мітку кластера y_i Y. Найчастіше множина Y заздалегідь не відома і додатковим завданням є визначення оптимального числа кластерів з точки зору того чи іншого показника якості кластеризації [10].

Кінцевою метою процесу кластеризації є отримання змістовних відомостей про структуру досліджуваних даних, що, як правило, є початковим етапом їх більш детального аналізу.

В результаті застосування різних методів кластеризації можуть бути отримані неоднакові результати, це є наслідком особливості роботи того чи іншого алгоритму, які слід враховувати при виборі методу кластеризації для конкретного завдання [11].

Слід враховувати, що кластерний аналіз пов'язаний з рядом складнощів:

1. Вибір методу кластеризації досить ефективного для вирішення певної задачі, вимагає достатнього знання алгоритмів і умов їх застосування.

2. Вибір характеристик, на підставі яких проводиться кластеризація: метрики, початкових значень центрів, умов зупинки алгоритму.

3. Вибір початкового числа кластерів. Якщо немає ніяких відомостей щодо можливого числа кластерів, необхідно здійснити ряд експериментів і проаналізувати отримані результати.

Основні використовувані алгоритми кластеризації зазвичай діляться на два види: ієрархічні і неієрархічні.

Принцип роботи ієрархічних алгоритмів полягає в послідовному об'єднанні маленьких кластерів в великі або навпаки поділі великих кластерів на маленькі.

Відповідно, або на початку роботи алгоритму всі об'єкти є окремими кластерами і на наступних кроках найбільш схожі об'єкти об'єднуються в кластери до тих пір, поки всі об'єкти не об'єднаються в один. Або спочатку всі об'єкти належать одному кластеру, який на наступних кроках розділяється на менші кластери, в результаті чого утворюється послідовність разделяющихся підмножин [11].

Перевага цієї групи методів - їх наочність і можливість отримати детальне уявлення про структуру даних. Недоліки: негнучкість отриманих класифікацій, обмеження обсягу аналізованих даних. Велика складність даних алгоритмів робить їх непридатними при значній кількості досліджуваних.

Неієрархічні методи засновані на поділі набору даних на певну кількість кластерів і виконанні ітеративного процесу оптимізації деякої цільової функції, яка визначає оптимальність (обумовлену особливостями алгоритму) даного розбиття множини об'єктів на кластери. На ітеративний процес накладається умова зупинки, в більшості випадків є параметром алгоритму [13].

Переваги цього типу методів в більш високій стійкості по відношенню до шумів, вибору метрики, додаванню груп незначущих об'єктів у вихідні дані, які беруть участь в кластеризації. Ціною, яку

доводиться платити за переваги подібних методів, є відомості, якими спочатку повинен володіти дослідник. Необхідно заздалегідь визначити параметри кластеризації, в тому числі кількість кластерів, а також кількість ітерацій або правило зупинки і деякі інші [14].

1.8 Кластерний аналіз методом k-середніх

Алгоритм роздільної кластеризації, заснований на розбитті множини елементів векторного простору на визначене заздалегідь число кластерів k. Алгоритм представляє собою ітераційну процедуру, в якій виконуються наступні кроки.

1. Обирається число кластеров k.
2. Із вихідної множини даних випадковим чином вибираються записи, які будуть служити початковими центрами кластерів.
3. Для кожного запису вихідної вибірки визначається найближчий до нього центр кластера. При цьому записи, "притягнуті" за певним центром, утворюють початкові кластери.
4. Обчислюються центроїди - центри тяжіння кластерів. Кожний центроїд - це вектор, елементи якого представляють собою середні значення ознак, обчислені за всіма записами кластера. Після цього центр кластера зміщується в його центроїд.
5. Третій та четвертий кроки ітеративно повторюються. Очевидно, що на кожній ітерації відбувається зміна границь кластерів і зміщення їх центрів. В результаті мінімізується відстань між елементами всередині кластерів.

Зупинка алгоритму проводиться тоді, коли межі кластерів та розташування центроїдів не перестануть змінюватися від ітерації до ітерації, тобто на кожній ітерації в кожному кластері буде залишатися один і той же набір записів. На практиці алгоритм звичайно знаходить набір стабільних кластерів за кілька десятків ітерацій.

Перевагою алгоритму є швидкість та простота реалізації. До його недоліків можна віднести невизначенність вибору початкових центрів кластерів, а також те, що кількість кластерів повинна бути задана з самого початку, що може вимагати деяку апріорну інформацію про вихідні дані.

Висновки до розділу 1

В даному розділі надається теоретичний матеріал стосовно алгоритму лінійної регресії. Детально проаналізований метод найменших квадратів, критерій Фішера, коефіцієнти кореляції та еластичності, кластерний аналіз.

РОЗДІЛ 2
РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

2.1. Програмні засоби, які були використані для реалізації

Для реалізації поставлених задач було використано наступні програмні засоби:

1. IntelliJ IDEA 2017 community edition (мова програмування – java) система для розробки програмного засобу, що реалізує математичні моделі системи аналізу [21].

IntelliJ IDEA 2017 community edition надає розвинений редактор коду, вбудований відладчик та інші засоби, що спрощують розробку додатків на мові java. Також має наступні переваги:

1. CodeLens - являє із себе підказки, які з'являються над вашим кодом, що надають інформацію про те, які залежно є у цього коду, результати тестів цього методу, хто міняв цей код, пов'язані робочі елементи.

2. IntelliTrace - Можливості IntelliTrace значно підвищують продуктивність налагодження. IntelliTrace автоматично веде журнали виконання коду, запам'ятовує і відзначає події в таймлайнах, які далі можна переглядати, переміщатися і перевіряти стан.

3. CodeMap - Ця можливість стане в нагоді при роботі з великими кодовими базами. Підтримується також переміщення по створеній карті з паралельно відкритим кодом. Це допомагає відслідковувати ваше місцезнаходження в коді під час роботи.

4. Доступні інструменти для побудови процесів управління проектами та командною роботою: Team Foundation Server.

5. Можливість підключення бібліотек з відкритим вихідним кодом [22].

Java:

1. Стандартизована. Перша версія Java стандартизована в ECMA (Standard ECMA-334 Java Language Specification, 3rd edition (June 2005)) та ISO (ISO / IEC 23270: 2003, Information technology – Java Language Specification).

2. JIT-компіляція.

3. Можливість використання стандартних бібліотек програмування включених в Java, що включає широкий спектр API-інтерфейсів для роботи як з самим обладнанням так і з операційною системою.

4. Компіляція в CIL (common intermediate language) код, який може виконуватися на будь-якій машині, де підтримується CLR (common language runtime).

5. CodeLens доступний тільки для Java.

6. Garbage collector.

Приблизно в 1990 Джеймс Гослінг, Білл Джой, Патрік Ногтон і інші в Sun Microsystems почали розробляти мову на ім'я Oak. Перш за все вони бачили застосування Java для вбудованих мікрокомп'ютерних модулів побутової техніки, в відеомагнітофонах, тостери, а також для PDA (personal data assistants).

Щоб вирішувати ці завдання, Oak повинен був бути, незалежним від платформи (з тих пір як продукцію стали залучати багато виробників), надзвичайно надійним, компактним.

Однак, в 1993 ринки інтерактивного телебачення і PDA пішли на спад. Тоді бурхливо розвивався internet і мережі. Так що Sun зрушила цільовий ринок в сторону internet-додатків і замінила назву проекту на Java [22].

В основі Java лежать мови C і C ++. Його синтаксис виходить з C, а орієнтовані на об'єкт особливості впливає C ++.

У 1994 Sun's випустила браузер HotJava. Він був написаний на Java за кілька місяців, що показало потужність аплетів, програм які працюють в межах браузера, а також для того, щоб прискорити процес розробки програм [23].

Розвивається поряд з величезним інтересом до internet, Java швидко набув широкого поширення, і чекав зростання для того, щоб стати домінуючою мовою програмування для створення програмного забезпечення для споживачів і браузера [25].

Однак, ранні версії Java не мали широтою і глибиною можливостей, необхідних для додатків клієнта (тобто споживача). Наприклад, графіка в Java 1.0 здавалася грубою і незграбною в порівнянні програмним забезпеченням, розробленим на C та іншими мовами.

Поряд з тим, що Java відставав у розвитку клієнтських додатків, він став дуже популярною мовою для розвитку підприємств, або мікропрограмних засобів, додатків типу інтерактивної пам'яті, діалогових обробок запитів, інтерфейсів бази даних, і т.д. Java також став досить звичайним на невеликих платформах типу стільникових телефонів і PDAs.

Java - повністю об'єктно-орієнтована мова програмування. В Java відсутнє поняття процедур. За допомогою Java ми можемо вирішити різні завдання і той же самий коло проблем, що і на інших мовах програмування. Java може використовуватися для створення двох типів програм: Додатків і аплетів. Додаток - програма, яка виконується на нашому комп'ютері, під його операційною системою. Програми Java можуть бути безпосередньо виконані, використовуючи інтерпретатор Java. Аплет - невелика програма працює з вікнами, які впроваджені в сторінку HTML. Щоб виконати Java аплети, потрібна підтримка Java Web-браузером, тобто Internet Explorer, Netscape Navigator, Hot Java і т.д. або засіб перегляду аплету. Також Java допускав інші засоби, за допомогою яких браузер міг виконати програму Java на нашій системі [27].

Java - це інтерпретується і компілює мову програмування. Оригінальний текст (файли з розширенням a Java) откомпилирован з довідкою компілятора Java (javac), який перетворює вихідний текст в байт-код (файли з розширенням a.class). Мета проектувальників Java полягала в тому, щоб розробити мову, за допомогою якого програміст міг записати код, який міг би виконуватися завжди, в будь-який час [28].

Проектувальники Java намагалися розробити мову, який могли б швидко вивчити програмісти. Також вони хотіли, щоб мова була знаком більшості програмістів, для простоти переходу. Звідси, в Java проектувальниками було видалено безліч складних особливостей, які існували в C і C ++. Особливості, такі як маніпуляції покажчика, перевантаження оператора і т.д. в Java не існують [30].

Java не використовує goto інструкцію, а також не використовує файли заголовка. Конструкції подібно struct і union були видалені з Java.

В Java все може бути об'єктом. Так основна увага приділяється властивостям і методам, які оперують даними в нашому додатку і немає концентрації тільки на процедурах. Властивості і методи разом описують стан і поведінку об'єкта. В Java ми будемо наштовхуватися на термін метод дуже часто, з ним ми будемо повинні познайомитися. Термін метод використовується для функцій [22].

Java може використовуватися для розробки додатків, які працюють на різних платформах, операційних системах і графічних інтерфейсів користувача. Java призначений також для підтримки мережевих додатків. Таким чином Java широко використовується як інструмент розробки в середовищі подібної Internet, де існують різні платформи.

Java - мова зі строгим контролем типів, так що потрібно явне оголошення методу. Java перевіряє код під час трансляції та під час інтерпретації. Таким чином усуваються деякі типи помилок при програмуванні. Java не має покажчиків і відповідно арифметичних операцій над ними. Всі дані масивів і рядків перевіряються під час

виконання, що виключає можливість виходу за межі дозволеного. Перетворення об'єктів з одного типу на інший також перевіряється під час виконання [27].

Автоматична обробка- У традиційних середовищах програмування, програміст повинен був вручну розподіляти пам'ять, і в кінці програми мав явну кількість вільної пам'яті. Виникали проблеми, коли програміст забував звільняти пам'ять. В Java, програміст не повинен турбуватися про проблему, пов'язану зі звільненням пам'яті. Це робиться автоматично, оскільки Java забезпечує обробка винятків для об'єктів, які не використовуються [28].

Обробка винятків спрощує завдання обробки помилок і відновлення.

Віруси - велика причина занепокоєння в світі комп'ютерів. До Java, програмісти повинні були спочатку переглянути файл перед завантаженням і виконанням. Навіть після цього вони не були впевнені в надійності файлу. Також, існує багато спеціальних програм, про які ми повинні знати. Ці програми можуть знаходити уразливі дані нашої системи.

Java забезпечує керовану середу, в якій виконана програма. Java ніколи не припускає, що код може бути безпечно виконаний. І так як Java - більше ніж мова програмування, він забезпечує кілька рівнів контролю захисту. З довідкою цих рівнів, він гарантує безпечне середовище виконання [24].

Перший рівень - це безпека, забезпечена мовою Java. Властивості і методи описуються в класі, і до них можна звернутися тільки через інтерфейс, забезпечений класом. Java не дозволяє ніяких операцій з покажчиками, таким чином забороняє прямий доступ до пам'яті. Уникає переповнення масивів. Проблеми, пов'язані з безпекою і мобільністю, приховані [25].

На наступному рівні компілятор, перш ніж приступити до компіляції коду, перевіряє безпеку коду і потім слід відповідно до протоколів, встановленими Java [26].

Третій рівень - це безпека, забезпечена інтерпретатором. Перш, ніж байт-код буде фактично виконаний, він є повністю укритим верифікатором.

Четвертий рівень піклується про завантаження класів. Завантажувач класу гарантує, що клас чи не порушує обмеження доступу перш, ніж він завантажений в систему [27].

Ми здатні виконати код Java на множинних платформах. Нейтралітет досягається при змішуванні трансляції та інтерпретації.

1. Програми Java оттранслировать в байт-код компілятором.

2. Байт-код виконується інтерпретатором (Віртуальна Машина Java).

3. Інтерпретатор повинен виконувати байт-код для кожної апаратної платформи.

4. Байт-код виконується на будь-якої версії Віртуальної Машини Java.

Незалежність від платформи означає легкість перенесення програми з одного комп'ютера на інший комп'ютер без будь-яких труднощів. Також Java - платформа, незалежна на обох рівнях, тобто на первинному (вихідному) і на вторинному рівні [28].

Java - це мова зі строгим контролем типів, що означає, що ми повинні оголошувати тип для кожної змінної. І ці типи даних в Java однакові для всіх платформ. Java має свої власні бібліотеки фундаментальних класів, які полегшують запис коду для програміста, який може бути переміщений з однієї машини на іншу, без потреби перезапису коду. Коротше кажучи, незалежність від платформи на початковому рівні означає, що ми можемо перемістити наш вихідний текст з однієї системи в іншу, компілюючи код, і працюючи в системі [29].

Платформа, незалежна на вторинному рівні означає, що відкомпільований двійковий файл може бути виконаний на різних платформах, які не перетрансліруя код, якщо вони мають Віртуальну Машину Java, яка функціонує як інтерпретатор [25].

Для програм, які ми записуємо в C і C ++ або на будь-якому іншому мовою, компілятор перетворює набір команди в машинний код або команди процесора. Ці команди є спеціальними для нашого процесора. В результаті, якщо ми хочемо використовувати цей код в деякій іншій системі, ми повинні знайти компілятор для цієї системи, і ми повинні компілювати код ще раз так, щоб ми мали машинний код, визначений для цієї машини. Коли ми подивимося на середу розвитку Java, ми побачимо поділ на дві частини: Компілятор Java і Інтерпретатор Java. На відміну від C і C ++, компілятор Java перетворює вихідний текст в байт-коди, які є машинно-незалежними. Байт-коди - це тільки частини команд Java, розрізані на байти, які можуть бути декодовані будь-яким процесором [21].

Інтерпретатор Java, також іменований як JVM (Віртуальна Машина Java) або Java Runtime Interpreter виконує байт-коди Java. Інтерпретатор Java є частиною середовища розробки. Скоро буде час, коли кожна операційна система буде мати в своєму складі JVM. Java - це мова, що інтерпретується. Це означає, що кожна команда оттранслировать в машинний код під час виконання, а не протягом трансляції.

Ця якість дозволяє реалізовувати нейтралітет платформи "WORA", або Write Once Run Anywhere.

А також дозволяє перезаписувати і змінювати програму, під час її виконання.

Трансляція java і процедура виконання включають таке різні вихідні файли обробляються компілятором javac, для отримання безліч файлів класу. Ці файли містять байт-код, який не залежить від архітектури і платформи виконує його [24].

Файли класу Java можуть бути виконані з довідкою завантажувача (інтерпретатора), утиліти по імені java, яка функціонує, щоб транслювати універсальні байт-коди Java в машинні виконуються коди. Ніякої компоновщик при цьому не потрібно [25].

Файли класу Java можуть бути виконані на будь-якій платформі за умови, що дана платформа має належну утиліту завантажувача java.

Файли класу Java роблять більш ефективне використання пам'яті, ніж окремі (часто великі) виконуються програми, тому що файли класу можуть бути пов'язані загрузчиком на підставі керованого запиту.

Java був розроблений, щоб добре працювати на центральних процесорах з дуже низьким енергоспоживанням. Байт-код Java був ретельно продуманий так, щоб його можна було відразу безпосередньо транслювати в машинний код з високою ефективністю, використовуючи компілятор [26].

Вбудована підтримка багатопоточності постачає програмістів Java потужним інструментом для поліпшення інтерактивної роботи графічних додатків. Якщо наш додаток має виконувати мультиплікацію і музику гри при прокручуванні сторінки і завантаження текстового файлу з сервера, то многопоточність - це спосіб швидко реалізувати поставлену задачу.

Потоки іноді також називають легкими процесами або контекстами виконання.

Потоки - це основний наріжний камінь Java. Бібліотека Java забезпечує клас потоку, який містить велику колекцію методів запуску, виконання, і зупинки потоку, а також перевірки його стану.

Java – динамічна адаптована мова програмування. Програми Java несуть багато інформації під час виконання, для перевірки правильності звернення до об'єктів під час виконання. Це властивість дозволяє динамічно безпечно зв'язати код. JVM (Віртуальна Машина Java) - основа мови програмування Java. Середовище Java складається з п'яти елементів:

1. Мова Java.

2. Визначення байт-коду.

3. Бібліотеки класу Java / Sun.

4. Віртуальна машина Java.

5 Структура файлу .class.

З усіх цих п'яти елементів, елементи, які привели до успіху Java:

1. Визначення байт-коду.

2. Структура файлу .class.

3. Віртуальна машина Java.

Таким чином "write once and run anywhere", було фактично здійснено завдяки мобільності файлу .class, який допомагає виконанню на будь-якому комп'ютері або наборі мікросхем з використанням Віртуальної Машини Java [27].

Віртуальна машина - це програмне забезпечення, засноване на поняттях і ідеї щодо уявного комп'ютера, який має логічний набір команд, і команд, що визначають операції цього комп'ютера. Це .можна сказати, невелика операційна система. Вона формує необхідний рівень абстракції, де досягається незалежність від платформи, використовуваного обладнання [29].

Компілятор конвертує вихідний текст в код, який заснований на уявній системі команд комп'ютерів і не залежить від специфічності процесора. Інтерпретатор-додаток, яке розуміє ці потоки команд і перетворює ці команди для використовуваного обладнання, до якого належить інтерпретатор. JVM створює систему підтримки виконання внутрішньо, що допомагає виконанню коду при:

1. Завантаженні файлів .class.

2. Управлінні пам'яттю.

3. Виконанні обробки винятків.

Через неузгодженість апаратних платформ віртуальна машина використовує поняття стека, який містить наступну інформацію:

1. Описувачі стану методу.

2. Операнди байт-кодами.

3. Параметри методів.

4. Локальні змінні.

Коли код виконується за допомогою JVM, то існує один спеціальний регістр, який використовується як лічильник, вказуючи виконуються в даний час команди. Якщо необхідно, команди змінюють програму, змінюють потік виконання, інакше потік послідовний і переходить від однієї команди до іншої [25].

Інше поняття, яке стає популярним - це використання Just In Time (JIT) компілятор. Браузери подібно Netscape Navigator 4.0 і Internet Explorer 4.0 включають JIT компілятори, які збільшують швидкість виконання кодів Java. Основна мета JIT полягає в тому, щоб перетворити систему команд байт-коду до машинних командам коду, цілеспрямованим для специфічного мікропроцесора. Ці команди зберігаються і використовуються всякий раз, коли запит робиться до цього специфічного методу.

JRE (Java Runtime Environment, Виконавча Java) JVM взаємодіє з апаратними засобами на одній стороні і програмою на іншому. JRE виконує код, відкомпільований для JVM:

1. Завантаження .class файлів.
2. Виконується за допомогою 'завантажувач класів'.
3. Завантажувач класу робить перевірку захисту, якщо файли використовуються в мережі.

Перевірка байт-коду виконується 'верифікатором байт-коду'

Верифікатор байт-коду перевіряє формат коду, перетворення типів об'єктів і перевіряє порушення прав доступу [23].

Виконання коду виконується 'інтерпретатором під час виконання'

Інтерпретатор виконує байт-коди і робить запити на використовуване обладнання.

В С, С ++ або Паскалі, програмісти використовували примітивні методи розподілу та звільнення блоків пам'яті - динамічну пам'ять. Динамічна пам'ять -великий шматок пам'яті, який позначений в обсязі всієї пам'яті.

Вільний список перевіряє блок пам'яті щоразу, коли робиться запит. Використовується механізм розподілу - "метод першого підходящого блоку", за допомогою чого перший найменший блок пам'яті розподіляється в залежності від запиту. Ця процедура розподіляє і звільняє невеликі обсяги пам'яті різних розмірів від динамічної пам'яті, при цьому фрагментація динамічної пам'яті зводиться до мінімуму.

Існує стадія, за допомогою якої виконується запит до пам'яті - для отримання більшого блоку пам'яті, ніж доступно. У таких випадках програма керування динамічною областю повинна створити більше пам'яті. Цю методику називають ущільненням. Це процес, за допомогою якого всі вільні доступні блоки пам'яті об'єднуються разом, переміщаючи вільну пам'ять одного з кінців динамічної пам'яті, таким чином створюючи один великий блок пам'яті.

Віртуальна Машина Java використовує дві окремі динамічних пам'яті для статичного і динамічного розподілу пам'яті.

Динамічна пам'ять - робить обробку винятків динамічної пам'яті, яка зберігає всі властивості класу, постійний пул і таблиці методів.

Друга динамічна пам'ять знову розділена на два розділи, які можуть бути розширені в протилежних напрямках коли буде потрібно. Один розділ використовується, щоб зберігати зразки об'єктів, а інший розділ використовується, щоб зберігати дескриптори в ці зразки. Дескриптор - структура, яка складається з двох покажчиків. Вказуєте на таблицю методів об'єкта та інших пунктів до зразком того об'єкта. Це розміщення в основному усуває потребу збереження шляхів, вказують на об'єкт при модифікуванні покажчиків після ущільнення. Все, що ми повинні зробити - це оновити значення покажчика дескриптора.

Алгоритм обробки виключень застосовується до об'єктів, поміщеним в динамічну пам'ять. Оскільки запит про блок пам'яті отриманий, програма керування динамічною областю перші перевірки вільний список і якщо програма керування динамічною областю не може знайти вільні блоки пам'яті, викликається обробка винятків, як тільки система має простий протягом достатнього періоду часу. У випадках, коли додатки дуже інтерактивні і час простою системи зведено до мінімуму, обробку винятків потрібно викликати явно додатком.

Колектор винятків викликає завершується метод перш, ніж за допомогою обробки винятків збирається зразок об'єкта. Завершується метод використовується щоб очистити зовнішні ресурси подібно файлам і потокам, які є відкритими і про які не подбали в стандартній обробці виключень. Навіть якщо ми явно викликаємо обробку винятків методом (System.gc ()), це не буде працювати швидко. Це просто намічено для того, щоб працювати. Також це означає, що обробка винятків не може бути викликана. Це пояснюється тим, що потоки обробки виключень виконуються в дуже низькому пріоритеті і можуть часто перериватися. Це може статися, коли наш об'єкт ніколи не розташовувався раніше в пам'яті. 1.4 середу редагування Java [26].

До JDK1.4 додалося безліч можливостей, які в свою чергу допомагають розширювати програми та аплети, які ми розробляємо в Java. Перш, ніж ми почнемо вивчати нові особливості JDK1.4, дозвольте нам спочатку розповісти про JDK. Це засіб розробки Java, який допомагає в розробці наших програм. Він складається з:

1. Класів.
2. Компілятора.
3. Відладчика.
4. JRE (Середовища виконання Java).

JDK забезпечує нас декількома інструментальними засобами, які постійно перебувають в каталозі JDK:

Javac - це команда, яка використовується для компіляції вихідного тексту. Вона конвертує вихідний файл (файл з розширенням Java) в файл класу (файл з розширенням .class)

Ця команда використовується для виконання файлу класу, який виконує класи в Віртуальної Машині Java.

Команда appletviewer дозволяє нам виконувати аплети без використання Web-браузера.

Документація API Java - корисний документ, що описує різні особливості java, що містить довідку для класів, пакетів, інтерфейси і т.д.

Є два способи, за допомогою яких ми можемо звернутися до документації [24].

Перший метод завантажує документацію API безпосередньо з сайту Sun. Для завантаження документації API можуть бути використані наступні кроки:

1. Відкрийте internet Explorer.

2. Введіть наступний адресс - http://iava.sun.com/docs/. Ви можете побачити наступне вікно.

3. Натисніть на посилання Download J2SE 1.4.1.

Другий метод звернення до документації API це безпосередньо з сайту Sun, якщо ми пов'язані з Internet. Ми можемо використовувати кілька кроків, щоб завантажити документацію API:

1. Відкрийте Internet Explorer.

2. Для отримання доступу до документації необхідно ввести наступну адресу: http://java.sun.com/i2se/1.4.1/docs/api/index.html.

3. Ви можете побачити web-сторінку, як показано нижче. Потім ви можете перейти на потрібний розділ.

Java - об'єктно-орієнтована мова програмування, що використовується головним чином для створення internet додатків.

Особливості Java:

1. Проста.

2. Об'єктно – орієнтованиа.

3. Розподілена.

4. Стійка.

5. Безпечна.

6. Незалежність від структури системи.

7. Мобільна.

8. Інтерпретує виконання.

9. Висока ефективність.

10. Динамічна.

JVM (Віртуальна машина Java) - основа мови програмування.

Java Runtime Environment (JRE) складається з JVM, яка взаємодіє з апаратними засобами з одного боку і з програмою з іншого [25].

JRE - це засіб розробки Java, яке допомагає в розробці програмного забезпечення.

JDK забезпечує нас декількома інструментальними засобами, які завжди знаходяться в каталозі JDK:

1. Javac.

2. Java.

3. Appletviewer.

Java API документація - корисні документи, що описують багато особливостей java, що містить довідку для класів, пакетів, інтерфейсів і т.д.

Java 8 являє собою новітню версію Java, що містить нові функції, удосконалення та виправлення помилок, які дозволяють підвищити продуктивність при розробці і запуску програм Java. Перед тим як опублікувати нову версію Java в розділі завантажень для кінцевих користувачів на веб-сайті java.com, розробники перевіряють її на наявність основних проблем. Тестування і сертифікація версії може зайняти деякий час.

Починаючи з випуску критичного оновлення за січень 2015 року, користувачам з активним з'єднанням автообновления буде пропонуватися оновити Java 7 до Java 8. Крім того, зверніть увагу, що випуск поновлення критичних виправлень в квітні 2015 р останнім загальнодоступним оновленням Java 7. Додаткову інформацію , а також відомості про те, як отримати підтримку для Java 7 на більш тривалий термін, див. в документі Oracle Java SE Support Roadmap. Деяким постачальникам додатків потрібна певна версія Java, і їх застосування не сертифіковані для Java 8. Якщо у вас виникли проблеми при запуску додатків з Java 8, зверніться до свого постачальника програми та переконайтеся, що додаток сертифіковане для даної версії. Далі наведено короткий опис удосконалень, реалізованих у версії Java 8. Методи лямбда-виразів і віртуального розширення. Найбільш примітна функція Java SE 8 - реалізація лямбда-виразів і допоміжних компонентів для мови програмування і платформи Java. API-інтерфейс дати і часу. Цей новий API-інтерфейс дозволить розробникам використовувати більш природні і зрозумілі методи обробки дати і часу. Механізм Nashhorn JavaScript Engine. Нова полегшена високопродуктивна реалізація механізму JavaScript Engine інтегрована в JDK і доступна для додатків Java через існуючі API-інтерфейси. Покращена безпека. Підтримуваний вручну список методів, чутливих до визиваючих програмам, замінений новим механізмом, який з високою точністю визначає ці методи і дозволяє стабільно виявляти їх викликають програми. Багато змін також з’явилося в EE версії. З ключових функцій можна відзначити численні CDI поліпшення, включаючи підтримку асинхронних подій. Також Java EE 8 підтримує HTTP / 2 в Servlet 4.0 і має новий API безпеки для хмарних і PaaS додатків. У JAX-RS додана підтримка посилаються сервером подій (Server-Sent Events). Реалізовано новий клієнтський API (REST Reactive Client API).

Проект Java EE буде розвиватися в рамках open source, так як Oracle передала права на нього компанії Eclipse, а вихідний код вже доступний в репозиторії GitHub.

Дев’яту версію Java Oracle планували випустити ще в середині літа 2016, однак реліз був перенесений спочатку на півроку, а потім і зовсім на другу половину 2017. І ось, 21 вересня 2017, вихід Java SE 9 відбувся.

У Java SE 9 з'явився новий API для розробки HTTP-клієнтів, що підтримує HTTP / 2.0 і WebSockets. Розширено набір діагностичних команд: print_class_summary, print_codegenlist, datadump_request, print_codeblocks, set_vmflag. Вирішено проблему з монолітними JAR-файлами і поширенням наборів класів, так як з'явилася нова модульна система.

Java SE 9 підтримує криптографічні хеш-функції SHA-3 і забороняє використання сертифікатів на базі SHA-1.

В новинку доданий інтерактивний інструмент Read-Eval-Print-Loop jshell, що дозволяє розробникам експериментувати з новими можливостями і API. А також засоби генерації документації Javadoc в форматі HTML5.

Java SE 9 отримав понад 150 нових функцій. Ця специфікація Java SE 9 Platform, недавно схвалена спільно з Java EE 8 в Java Community Process (JCP), повністю сумісна з попередніми версіями.

2.2. Проектування програмного продукту

2.2.1. Модель життєвого циклу

Модель життєвого циклу - структура, що складається із процесів, робіт та задач, які включають в себе розробку, експлуатацію і супровід програмного продукту; охоплює життя системи від визначення вимог до неї до припинення її використання [27].

Каскадна модель життєвого циклу (модель водоспаду, англ. waterfall model) у 1970 р. була запропонована У. Ройсом. Принциповою особливістю каскадної моделі є т, що перехід на кожну наступну стадію здійснюється тільки після повного завершення роботи поточної стадії, повернення на попередні стадії не передбачено. Кожна з стадій закінчується одержанням результатів, що є вхідними даними для наступної стадії, та випуском повного комплекту документації. Вимоги до ПЗ, визначені на стадії формування вимог, документуються у вигляді технічного завдання і фіксуються на весь час розроблення [27]. Критерієм якості розробки за такої моделі є точність виконання специфікацій технічного завдання.

Цінність цієї моделі полягає в тому, що вона фіксує послідовність етапів розроблень та можливість повернення до попередніх етапів роботи.

Основна увага розробників має зосереджуватися на досягненні найкращих значень технічних характеристик ПЗ, а саме: продуктивності, обсягу пам'яті тощо.

Каскадна стратегія (одноразовий прохід, водоспадна або класична модель) має на увазі лінійну послідовність виконання стадій створення інформаційної системи (рис. 2.4). Іншими словами, перехід з однієї стадії на наступну відбувається тільки після того, як буде повністю завершена робота на поточному [27].

Дана модель застосовується при розробці інформаційних систем, для яких на самому початку розробки можна досить точно і повно сформулювати всі вимоги [27].

Переваги моделі:

- на кожній стадії формується закінчений набір документації, програмного і апаратного забезпечення, який відповідає критеріям повноти і узгодженості;

- виконуються в чіткій послідовності стадії дозволяють впевнено планувати терміни виконання робіт і відповідні ресурси (грошові, матеріальні і людські).

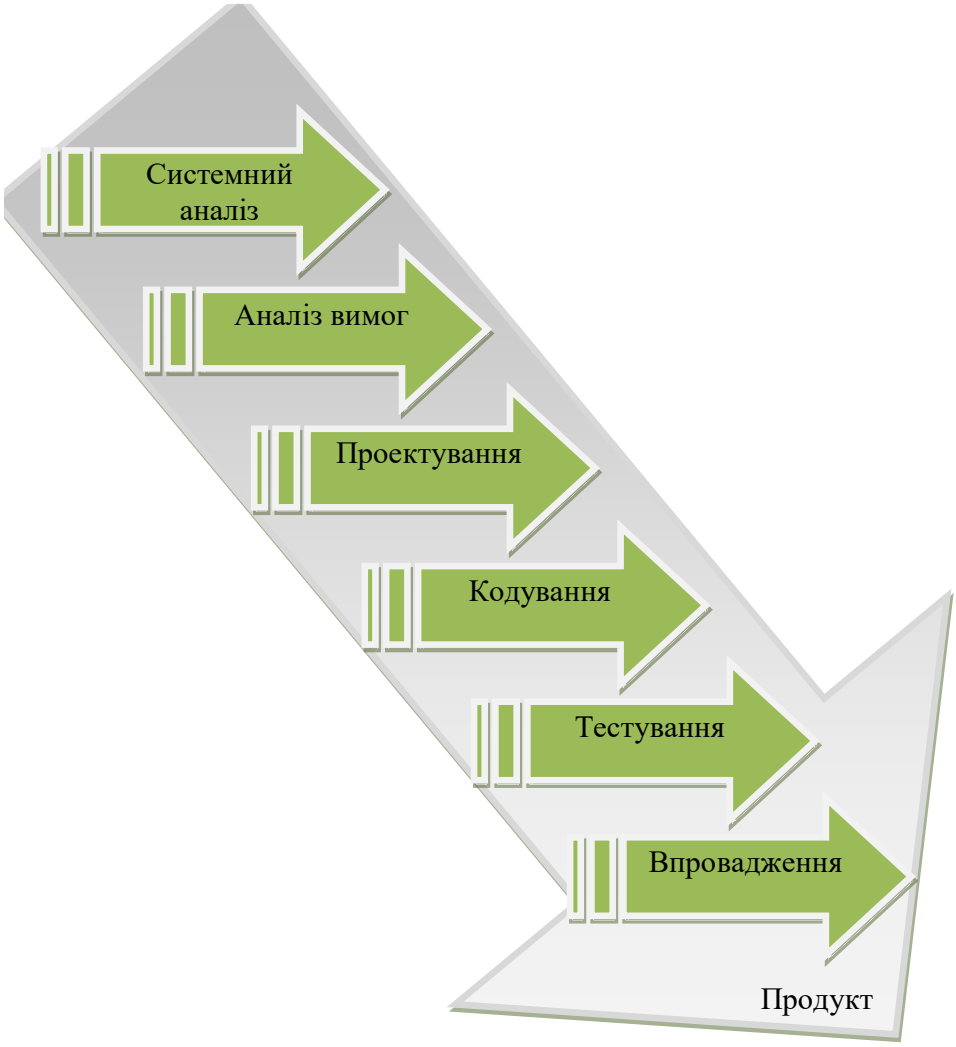

Рисунок 2.4. Каскадна стратегія.

Недоліки моделі:

- реальний процес розробки інформаційної системи не часто повністю відтворює таку не гнучку схему. Особливо це має відношення до розробки не типових і новаторських систем;

- заснована на точному формулюванні всієї групи вихідних вимог до інформаційної системи. В реальності ж на початку проекту вимоги часто визначено не повністю;

- основним недоліком є те, що результат розробки доступний замовнику тільки в кінці проекту. У випадку неточного встановлення вимог або їх частої зміни протягом періоду створення ПП замовник отримає продукт, який не відповідає його потребам.

Інкрементна стратегія (англ. Increment - збільшення, прирощення) має на увазі розробку інформаційної системи з стадіями в лінійній ослідовності, але з декількома інкрементами (версіями), тобто з запланованими поліпшеннями продукту (рис. 2.5).

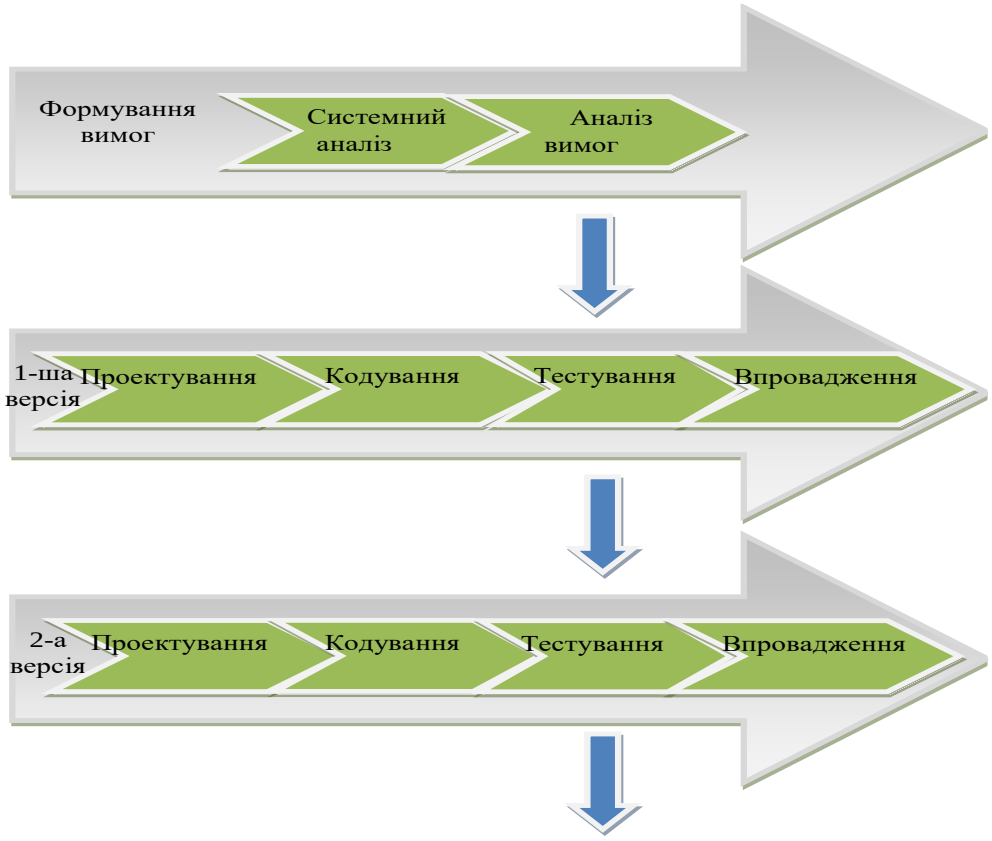

Рисунок 2.5. Інкрементна стратегія.

На початку роботи з проектами мають бути визначені всі основні вимоги замовника до системи, далі виконується власне її розробка як послідовності версій. В той же час, кожна з версій є закінченою і являє собою працездатний продукт. Перша з цих версій реалізує лише частину з можливостей, які були заплановано, кожна наступна інкремента реалізує якісь з додаткових можливості і т. д., до поки не буде отримано готовий програмний продукт [27].

Ця модель життєвого циклу продукту характерна для розробки складної і комплексної системи, для якої наявні чіткі вимоги (як з боку замовника, так і з боку розробника) до того, яким має бути врешті решт кінцевий продукт (інформаційна система) [27]. Розробка версіями через безліч різних причин:

- відсутність у замовника можливості фінансування всього довгострокового проекту відразу;
- відсутність у розробника необхідних ресурсів для реалізації складного проекту в стислі терміни;
- через наявність вимог до поетапного впровадження і освоєння продукту кінцевим користувачем. Впровадження всієї системи відразу може викликати у її користувача відразу і тільки «загальмувавши» процес переходу на нові технології можливо сподобатися.

Переваги і недоліки цієї стратегії такі ж, як і у класичної. Але на відміну від класичної стратегії замовник може раніше побачити результати. Уже за результатами розробки та впровадження першої версії він може незначно змінити вимоги до розробки, відмовитися від неї або запропонувати розробку більш досконалого продукту з укладенням нового договору.

Спіральна стратегія (еволюційна або ітераційна модель, автор Баррі Боем, 1988 г.) має на увазі розробку у вигляді послідовності версій, але на початку проекту визначені не всі вимоги (рис. 2.6). Вимоги уточнюються в результаті розробки версій [27].

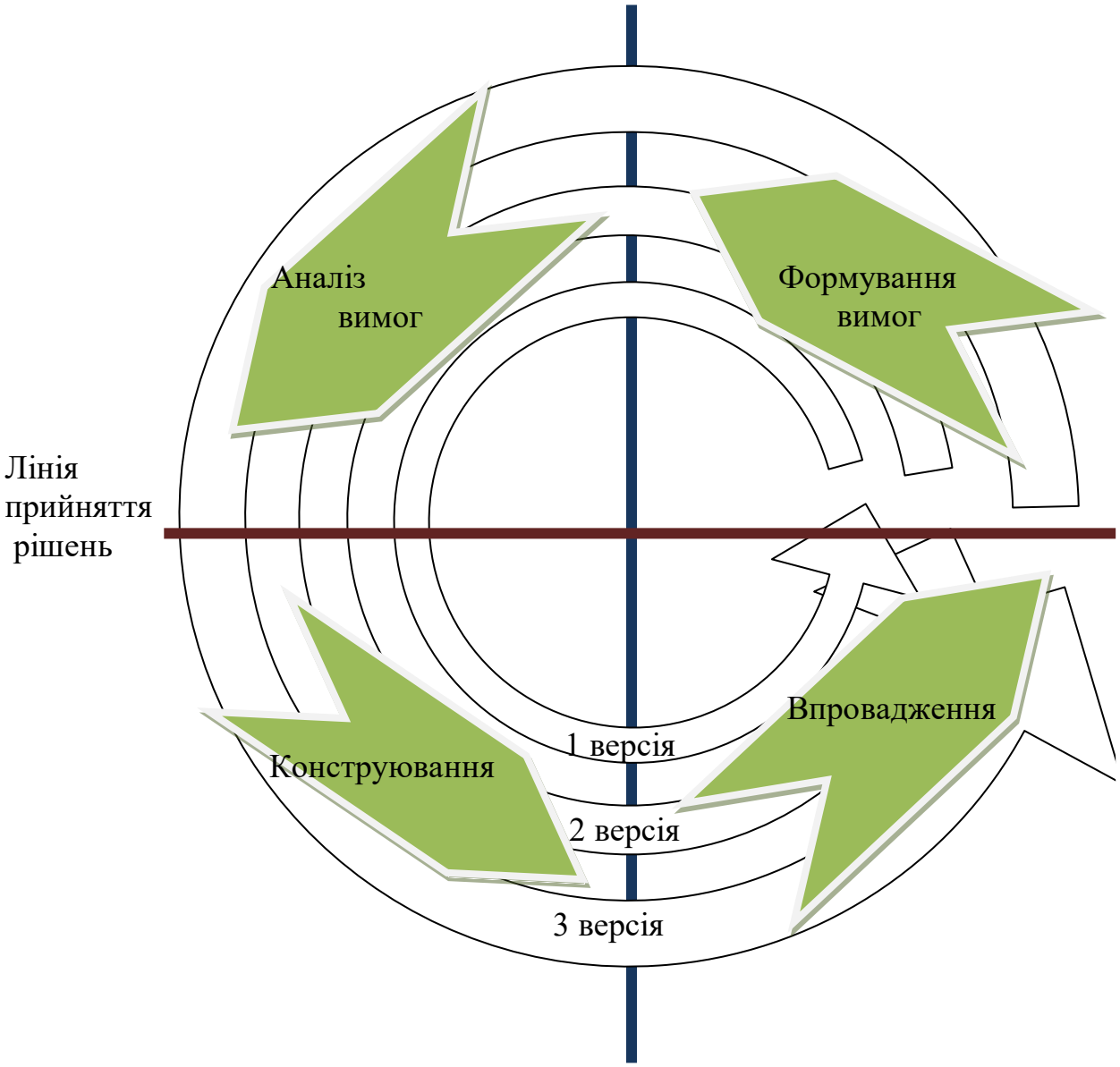

Рисунок 2.6. Спіральна стратегія.

Дана модель життєвого циклу характерна при розробці новаторських (нетипових) систем. На початку роботи над проектом у замовника і розробника відсутнє чітке бачення кінцевого продукту (вимоги не можуть бути чітко визначені) або стовідсоткової впевненості успішної реалізації проекту (ризики дуже великі). У зв'язку з цим приймається рішення розробки системи по частинах з можливістю зміни вимог або відмови від її подальшого розвитку. Як видно з рис.2.6, розвиток проекту може бути завершено не тільки після стадії впровадження, але і після стадії аналізу ризику.

Переваги моделі:

- швидше показує користувачам системи працездатні готові продукти, таким чином, активізує процеси щодо уточнення і доповнення вимог;
- має можливість зміни вимог при розробці інформаційної системи, що є характерним для більшості розробок;
- більша гнучкість відносно управління проектом;
- дозволяє отримати більш надійну і стійку систему. В процесі розвитку системи помилки і слабкі місця виявляються і можуть бути виправлені на будь-якій з ітерацій;
- дозволяє удосконалювати процес розробки - аналіз, проведений в кожній ітерації, дозволяє проводити оцінку того, що повинно бути змінено в організації розробки, і поліпшити її на наступній ітерації;
- зменшуються ризики замовника. Замовник може з мінімальними для себе фінансовими втратами завершити розвиток безперспективного проекту.

Недоліки моделі:

- збільшується невизначеність у розробника в перспективах розвитку проекту. Цей недолік випливає з попереднього гідності моделі;
- ускладнення операцій тимчасового і ресурсного планування всього проекту в цілому. Для знаходження виходу з цієї ситуації необхідно ввести тимчасові обмеження на кожну зі стадій життєвого циклу. Навіть якщо не вся запланована робота виконана, перехід все рівно здійснюється відповідно до плану. План складається на основі статистичних даних, отриманих в попередніх проектах і особистого досвіду розробників.

Знання різних моделей життєвого циклу і вміння їх застосовувати на практиці необхідні будь-якому керівникові проекту. Правильний вибір моделі дозволяє грамотно планувати обсяги фінансування, терміни і ресурси, необхідні для виконання робіт, скоротити ризики як розробника, так і замовника. Це сприяє підвищенню авторитету (іміджу) розробників в

очах замовника і в свою чергу впливає на перспективу подальшої співпраці з ним та іншими замовниками. Вважати, що спіральна модель краще за інших, невірно. Адже на кожен проект полягає окремий договір з певною вартістю. Укладати договір на велику суму з невизначеним підсумковим результатом замовник ніколи не буде (якщо тільки він не альтруїст). У цьому випадку він запропонує вкласти спочатку невелику суму в проект і вже за результатами першої версії (ітерації) буде вирішувати питання про укладення додаткового договору на розвиток системи.

Кожна з моделей має свої переваги і недоліки, а також сфери застосування в залежності від специфіки розробляється, можливостей замовника і розробника і т. п. У табл. 2.1 наводиться порівняльна характеристика розглянутих вище моделей, яка повинна допомогти у виборі стратегії для конкретного проекту.

Таблиця 2.1

Порівняння моделей життєвого циклу

<table>
<tr><td rowspan="2">Характеристика проекту</td><td colspan="3">Модель (стратегія)</td></tr>
<tr><td>Каскадна</td><td>Інкрементна</td><td>Спіральна</td></tr>
<tr><td rowspan="2">Новизна розробки та забезпеченість ресурсами</td><td colspan="2">Типовий. Добре опрацьовані технологія і методи вирішення завдання</td><td>Нетипової (новаторський). нетрадиційний для розробника</td></tr>
<tr><td>Ресурсів замовника і розробника вистачає для реалізації проекту в стислі терміни</td><td colspan="2">Ресурсів замовника або розробника НЕ вистачає для реалізації проекту в стислі терміни</td></tr>
</table>

Продовж. табл. 2.1

Характеристика проекту	Модель (стратегія)		
	Каскадна	Інкрементна	Спіральна
Масштаб проекту	Малі та середні проекти	Середні і великі проекти	Будь-які проекти
Терміни виконання	До року	До декількох років.	
Висновок окремих договорів на окремі версії	Укладається один договір. Версія і є підсумковий результат проекту	На окрему версію або кілька послідовних версій зазвичай укладається окремий договір	
Визначення основних вимог на початку проекту	Так	Так	Немає
Розробка ітераціями (версіями)	Немає	Так	Так
Поширення проміжного програмного забезпечення	Немає	Може бути	Так

У табл. 2.1 не варто вважати «Так» і «Ні» жорсткими вимогами. Під час використання каскадної моделі (наприклад, додавання деяких непередбачених раніше розробником чи замовником функцій) незначна зміна вимог у міру розвитку проекту зустрічається не так уже й рідко і в

разі їх реалізації сприяє поліпшенню взаємин між сторонами. Аналогічно поширення проміжного програмного забезпечення при спіральної моделі необов'язково, а іноді навіть шкідливо відбивається на процесах впровадження і дослідної експлуатації системи.

При розробці системи під підсумковим продуктом і проміжним програмним забезпеченням згідно слід розуміти, що:

- ревізія (виправна або досвідчена) - будь-які оперативні зміни програмного та інформаційного забезпечення, а також БД, необов'язкові в даний момент до передачі на об'єкти впровадження та пов'язані з усуненням помилок і удосконаленням;
- модифікація - будь-які оперативні зміни програмного та інформаційного забезпечення, а також БД, обов'язкові для передачі на об'єкти впровадження і обумовлюють зміну експлуатаційних характеристик без зміни функцій (передбачених ТЗ), а також зміни, пов'язані з усуненням помилок, удосконаленням;
- версія - будь-які зміни програмного та інформаційного забезпечення, а також БД, обов'язкові для передачі на об'єкти впровадження, які дозволяють виконувати заявлені або додаткові функції, а також забезпечують перехід на нові операційні системи і інформаційне середовище;
- розвиток (черга) - планові зміни інформаційної системи, пов'язані з введенням нових функцій і поліпшенням експлуатаційних характеристик, переходом на нову інформаційну середу, впровадженням нових комплексів технічних засобів, нових інформаційних технологій та ін.

Поміж існуючих моделей життєвого циклу (каскадна, гнучка, спіральна та еволюційна) було обрано каскадну модель. Оскільки це поетапне виконання визначених дій. Також в цій моделі значна увага приділяється інженерії вимог та власне проектуванню, що застраховує від вагомих помилок.

В даний час є кілька методологій розробки програмного забезпечення, які можна рекомендувати при використанні каскадної моделі життєвого циклу. Найбільш відомими з них є методологія швидкої розробки додатків (Rapid Application Development, RAD) і екстремальне програмування (eXtreme Programming, XP - автор Кент Бек, 1999).

Методологія RAD. На початковому етапі існування комп'ютерних інформаційних систем їх розробка велася на традиційних мовах програмування і мала на увазі, як правило, ручний набір текстів програм. Однак у міру зростання складності розроблюваних систем і збільшення запитів користувачів потрібні були нові засоби, щоб забезпечити значне скорочення термінів розробки. Це стало передумовою для створення інструментального засобу для швидкої розробки додатків. Розвиток таких напрямків призвело до того, що на ринку з’явилися розробки програмного забезпечення засобів автоматизації практично всіх стадій життєвого циклу [27].

Під RAD-розробкою зазвичай розуміють процес розробки, який складається з трьох елементів:

- невеликої команди з програмістів (до десяти осіб);
- короткого, але ретельно пропрацьованого виробничого графіку (від 2 до 6 місяців);
- повторюваного циклу, під час якого розробник коли додаток починає набувати форми, реалізує в продукті нові вимоги, отримані від замовника.

Крім особливостей, які характеризують спіральну модель життєвого циклу програмної системи, методологія RAD предбачає використання на кожній з ітерацій:

- CASE 2 - засобів формування та аналізу вимог, проектування системи, автоматичної генерації коду програм і структури БД, а також автоматичного тестування програмного забезпечення;

- інструментальних засобів, що забезпечують візуальну розробку (програмування) системи. Середовище розробки додатків дозволяє без написання коду програми створювати («малювати») складні графічні інтерфейси користувача, склад і структуру БД, запити до БД, а також пов'язувати дані з елементами інтерфейсу (перемикачами, полями введення, таблицями і т. Д.);
- інструментальних засобів, що підтримують об'єктно-орієнтований підхід. Ці засоби дозволяють створити опис предметної області у вигляді сукупності об'єктів - сутностей реального світу, характеризуються властивостями (атрибутами) і поведінкою (методами);
- інструментальних засобів, що забезпечують поділ програмування. Кожен об'єкт, що входить до складу програми, може генерувати події і реагувати на події, що генеруються іншими об'єктами;
- шаблонів і бібліотек готових рішень як власної розробки, так і сторонніх виробників.

Умови застосування методології RAD:

- застосовна для відносно невеликих проектів, що розробляються під конкретного замовника;
- непридатна для побудови складних розрахункових програм, операційних систем або програм управління космічними кораблями, тобто програм, що вимагають написання великого обсягу (сотні тисяч рядків) унікального коду;
- непридатна для розробки додатків, в яких відсутня яскраво виражена інтерфейсна частина, наочно визначає логіку роботи системи (наприклад, додатки реального часу);
- непридатна для розробки додатків, від яких залежить безпека людей (наприклад, керування літаком або атомною електростанцією), так як ітеративний підхід припускає, що перші кілька версій, швидше за все не будуть повністю працездатні, що в даному випадку виключається.

Методологія XP [27]. Даний підхід орієнтований на розробку інформаційних систем групами малого і середнього розміру в умовах невизначених або швидко змінюються вимог.

Відмінними рисами XP-розробки є:

- часта зміна версій і модифікацій (тривалість ітерацій аж до годин і хвилин, а зазвичай - 2 тижні; в RAD - мінімум 2 місяці);
- безперервний зв'язок із замовником - в групі весь час знаходиться кваліфікований представник замовника [27]. Це найпотаємніше бажання будь-якого розробника. Як правило, дуже важко переконати замовника виділити такого представника, щоб він цілими днями (тижнями) не виконував своїх прямих обов'язків на роботі. Але ще важче буває знайти саме кваліфікованого фахівця, який може оперативно відповісти на всі збільшує кількість питань з боку команди розробників;
- просте проектування - при розробці завжди вибирається найбільш просте рішення [28]. «Краще зробити просту річ сьогодні і завтра заплатити ще трохи за внесення невеликих змін, ніж робити сьогодні складну річ, яка завтра може не знадобитися». Спірне твердження, на яке можна заперечити, що «скупий платить двічі». Нерідко досвідченому розробнику, особливо непогано розбирається в поставленому завданню, видно, що якщо застосувати більш складну схему реалізації деякої функції або розширити структуру даних, то це збільшить коло вирішуваних завдань, дозволить з меншими витратами адаптувати систему під мінливі або нові вимоги замовника і т.д.;
- простий дизайн - система повинна бути спроектована настільки просто, наскільки це можливо на кожен момент часу. Чим інтерфейс простіше, тим швидше і якісніше йде освоєння системи користувачами [27]. Це не означає, що інтерфейс «командного рядка» самий кращий. Якраз навпаки, «дружній» і простий інтерфейс повинен бути інтуїтивно-зрозумілим для користувача; в ньому мають бути відсутні непотрібні елементи, основна мета яких - зробити візуальне враження на користувача;

додаткові діалогові вікна з введення даних в рядок таблиці, коли це можна зробити безпосередньо в рядку цієї таблиці і т. д .;

- колективне володіння кодом - будь-який, хто бачить можливість поліпшити якусь частину коду, може зробити це в будь-який момент часу [27]. Це має на увазі застосування однакових стандартів і правил оформлення коду з вичерпними коментарями, а також і ведення загальнодоступною історії розвитку системи;
- програмування в парах - на пару програмістів доводиться один комп'ютер. Поки один з них безпосередньо програмує, інший обмірковує питання реалізації вимог (функцій, БД і т.п.) [27]. Сенс положення полягає не в економії на матеріально-технічному забезпеченні команди розробників, а в більш розумному поєднанні різних видів діяльності кожного з них. Як показує досвід, при безпосередньому програмуванні, виправленні помилок і т.п. програміст починає думати односторонньо і все більш вузькими категоріями. Саме тому під час «відпочинку від комп'ютера» приходять найвдаліші рішення;
- безперервне і перехресний проектування, розробка, інтеграція і тестування системи.

2.2.2. Побудова ієрархічної структури та розрахунок нев’язки

Ієрархічна структура робіт (ІСР) (англ. Work Breakdown Structure, WBS, іноді Структура декомпозиції робіт, СДР) - це ієрархічне розбиття всієї роботи, яку необхідно виконати для досягнення цілей проекту, на більш дрібні операції і дії до такого рівня, на якому способи виконання цих дій цілком зрозумілі і відповідні роботи можуть бути оцінені і сплановані. Вона включає також визначення проміжних результатів всіх складових цю структуру робіт.

Work (Робота) - безперервне фізичне або розумове зусилля, спрямоване на подолання перешкод і досягнення цілей або результатів; специфічне завдання, обов'язок, функція або завдання, часто є частиною

фази або іншої, більшої за обсягом роботи; щось, вироблене або виконується в результаті зусилля або застосування навичок (кваліфікації).

Breakdown (Декомпозиція) - поділ на частини або категорії, виділення простих складових.

Structure (Структура) - фіксоване впорядкована множина об'єктів і відносин між ними, класифікація чого-небудь по заданому основи.

Ці визначення означають, що ієрархічна структура робіт має такі характеристики:

1. Описує з необхідною точністю зміст робіт по проекту.

Визначає весь обсяг робіт за проектом.

2. Формується у вигляді ієрархічної структури (проект декомпозіруется на пакети / субпакети).

3. Являє обсяг робіт по пакету як перелік робіт, що мають вимірний або порівнянний результат.

4. Має об'єктивний або вимірний результат, який розглядається як результат роботи по пакету або сукупність результатів робіт.

Розрахуємо нев’язку за допомогою засобів MATLAB (рис. 2.7)

```
>> n = 8; %количество вершин полученного графа
E = 8; %количество ребер полученного графа
Et = n-1; %количество ребер дерева
Ec = n*(n-1)/2; %количество ребер полного графа
Nev = (E-Et)/(Ec-Et); %расчет невязки
Nev %вывод значений невязки

Nev =

    0.0476
```

Рисунок 2.7. Розрахунок нев’язки.

Побудовану ієрархічну структуру зображено на рис. 2.8.

Рисунок 2.8. Ієрархічна структура ПП.

Як можна побачити, ми отримали дуже малу нев’язку, що прагне до нуля. Це свідчить про добре сплановану систему, яку доцільно розробляти.

2.2.3. Діаграма сутність-зв’язок (ERD)

Мета створення ERD-діаграми забезпечити перегляд вимог достатніх для задоволення потреб розробляється ІС (інформаційної системи). Фактично з ERD-діаграми і починається розробка моделі, коли визначається загальний перелік таблиць і зв'язків між ними. Потім вже визначаються ключі, атрибути і т.д., але спершу треба визначитися з таблицями.

Власне 5 моделей, які поступово розробляються, і передбачені в правильному порядку. Фактично Ви розробляєте деякі сутності (таблиці), зв'язку, атрибути (їх називають прикметниками). Формування моделей даних Erwina - це знаходження правильної сукупності іменників, дієслів і прикметників. Під сутностями розуміється безліч екземплярів реальних або абстрактних об'єктів. Наприклад: відомості про рахунки, про людей, станах, події. Кожен реальний або абстрактний об'єктів володіє деякими атрибутами або характеристиками. Будь-який об'єкт повинен бути ідентифікований за допомогою ключа.

У ERD сутність представляється прямокутником, ім'я рекомендується давати в сингулярному вигляді, тобто в однині.

У IDEF1x розрізняються залежні і незалежні сутності. Тип суті визначається зв'язком з іншими сутностями.

Зв'язки або Relationship. По суті зв'язок, яка на поле проектування зображується лінією насправді є логічним відношенням між сутностями. Всередині програми зв'язків у вигляді ліній існувати не може, тобто зв'язок це абстракція, що показує на те, що логічно ці 2 таблиці пов'язані. Зв'язки бувають декількох видів. Зазвичай зв'язок між головною підпорядкованої таблицею зображується суцільною (але бувають випадки, коли і пунктирною лінією). Т.ч. суті представляють собою базові типи даних, що зберігаються в базах даних, а зв'язку насправді показують, як ці дані взаємопов'язані. На логічному рівні можна встановити три типи зв'язків:

ідентифікує зв'язок один-ко-многим (кіностудія з випущеними фільмами), багато-до-багатьох (продавці і покупці) неідентіфіцірующей зв'язок один-ко-многим

Найчастіше між двома сутностями вибирається зв'язок один-ко-многим. Це означає, що один зразок з першої сутності пов'язаний з декількома зразками з іншої сутності. При цьому сутність на одиничному кінці називається батьківської або незалежною. А сутність на manyконце називається дочірньої або залежною. Вони відображаються по-різному.

При цьому незалежна відображається прямокутником, а залежна так само відображається прямокутником, але з округленими кутами. Якщо лінія зв'язку між двома сутностями суцільна, то її називають ідентифікує.

ERD діаграма згідно до завдань, що були сформовані на етапі системного аналізу та аналізу вимог, показана на рис. 2.9.

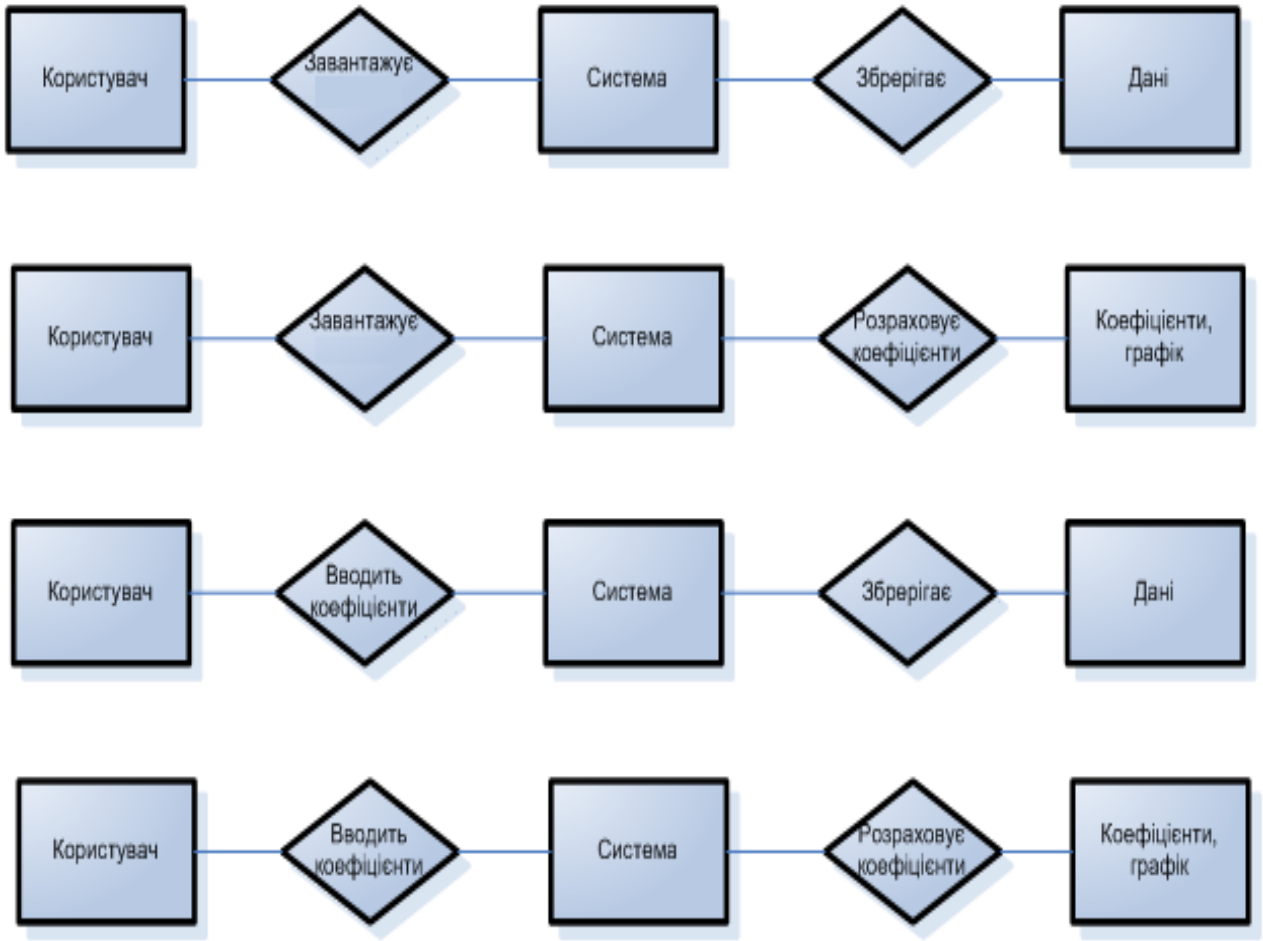

Рисунок 2.9. Діаграма сутність-зв’язок.

2.2.4. Контекстна діаграма IDEF0

Нотація IDEF0 - для документування процесів виробництва і відображення інформації про використання ресурсів на кожному з етапів проектування систем. Функціональне моделювання.

На рис. 2.10 зображена контекстна діаграма IDEF0, що відображає використання всіх ресурсів в проекті. Діяльністю виступає власне створення програмного продукту (ПП).

Управлінням виступають наступні ГОСТи:

Таблиця 2.2

Використані ГОСТи

Позначення	Найменування
Стандарти ISO/IEC (ІСО/МЕК) в області розробки і документування програмних засобів	
ГОСТ Р ІСО/МЕК 12207-02	Інформаційна технологія. Процеси життєвого циклу програмних засобів
ГОСТ Р ІСО/МЕК 9126-93	Інформаційна технологія. Оцінка програмної продукції. Характеристики якості і керівництва щодо їх застосування
ГОСТ Р ІСО/МЕК 12119-94	Інформаційна технологія. Пакети програм. Вимоги до якості і тестування
Комплекс нормативних документів на автоматизовані системи	
ГОСТ 34.601-90	Автоматизовані системи. Стадії створення
ГОСТ 34.602-89	Технічне завдання на створення автоматизованої системи
Комплекс стандартів Єдиної системи програмної документації (ЄСПД)	
ГОСТ 19.201-78	Технічне завдання. Вимоги до змісту та оформлення
ГОСТ 19.701-90 (ІСО/МЕК 5807-85)	Схеми алгоритмів програм, даних і систем. Умовні позначення і правила виконання

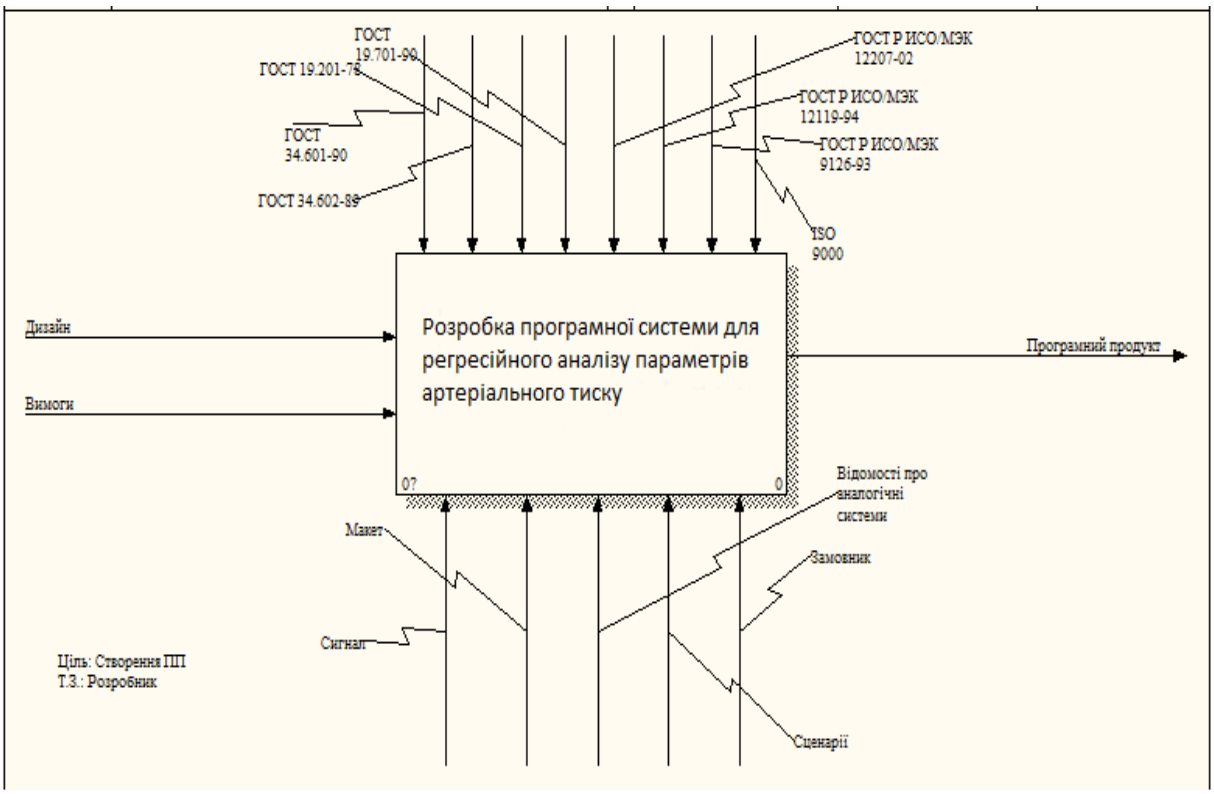

Рисунок 2.10. Контекстна діаграма.

На вхід подаються вимоги до ПП та його дизайн. Ресурсами виступають макет, сценарії, відомості про аналогічні системи, замовник. Виходом є власне ПП.

2.2.5. Діаграма декомпозиції IDEF0

Діаграма першого рівня декомпозиції A0, а також всі наступні діаграми декомпозиції, надають інтерфейсі обмеження (контекст) для дочірніх діаграм.

Було вирішено розбити створення програмного продукту на наступні складові: аналіз, проектування, реалізація та тестування, що зображено на рис. 2.11.

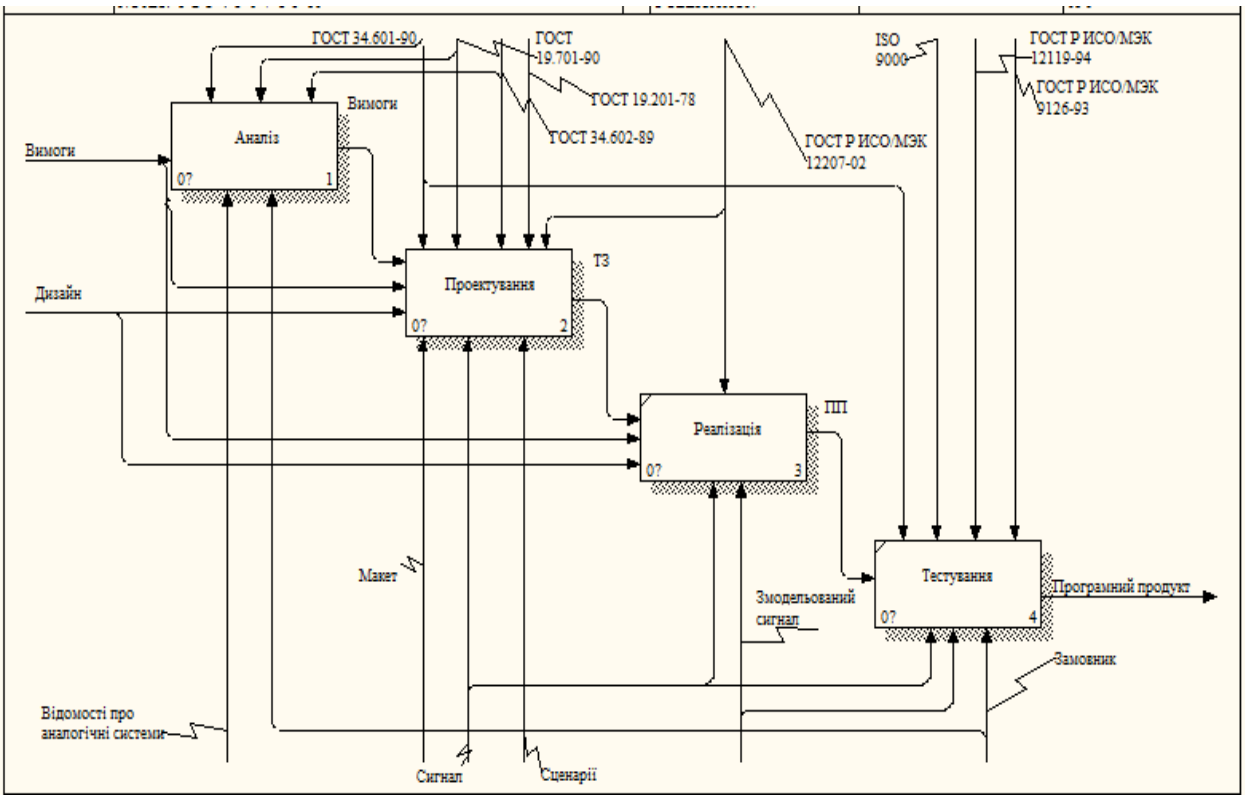

Рисунок 2.11. Діаграма декомпозиції IDEF0.

На рис. 2.12 зображена декомпозиція пункту Аналіз, що дозволяє більш детально розглянути його структуру.

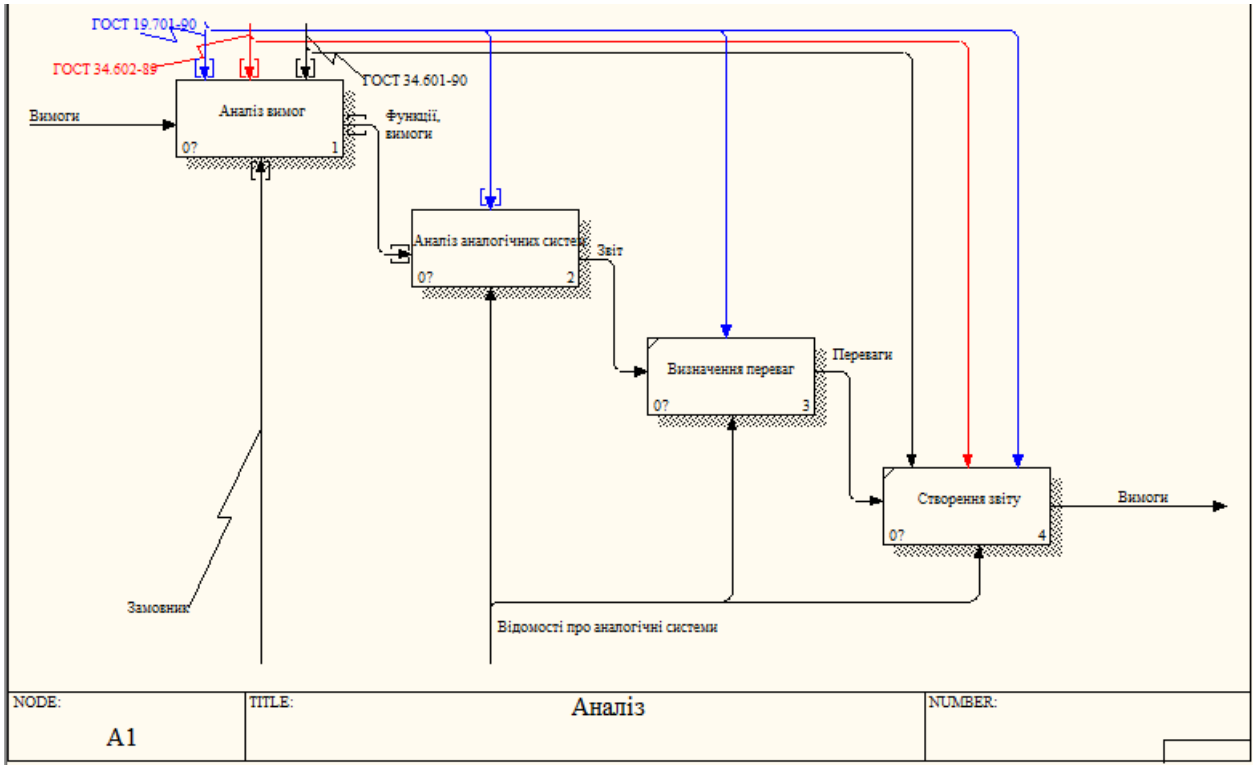

Рисунок 2.12. Діаграма декомпозиції IDEF0 (аналіз).

Не важко помітити, що аналізу вимог та аналізу використаних джерел відводиться важливе місце. Після них потрібно визначити переваги системи в порівнянні з вже існуючими.

2.2.6. Backlog

Планування і визначення пріоритетів є найважливішим елементом створення продукту. Від навичок керівника проекту, в нашому бізнесі - продюсера, залежить наскільки ефективно будуть використані ресурси компанії, який вийде вартість розробки, на скільки продуктивно працюватиме кожен член команди, якими будуть терміни. Всі ці аспекти залежать від грамотного планування і розстановки пріоритетів для кожного завдання.

Список всіх завдань (Backlog) проекту, починаючи від сирих ідей і закінчуючи добре декомпозіровать і описаними, готовими в роботу, планами спринтів. Цей список завдань, буде основним артефактом продюсера, інструментом для досягнення максимальних показників ефективності в роботі над проектом.

Беклог є відкритим, і додавати в нього завдання (ідеї, баги, побажання) може хто завгодно: від Стейкходера до тестувальника. Після того, як завдання потрапляють в беклог, продюсер визначає їх Бізнес цінність, а команда Складність. Саме ці два компоненти визначають, коли завдання йде в роботу. При плануванні, виникнення ідеї або бага в беклог заносяться Епіки - кореневі завдання / цілі. Наприклад в гру потрібно впровадити щоденний бонус, щоб посилити механізми утримання. Продюсер вписує епік «Додати Дейлі бонус» і збирає команду. Разом вони розбивають задачу на підзадачі. Дизайнер інтерфейсів каже, що йому треба намалювати пару скетчів, і просить художника підготувати графіком згідно стилю гри. Програміст просить гейм диза розписати принципи роботи, перед тим як збирати вікно. Бекенда запитує на скільки багато можна додати бонусів, і чи потрібно синхронізувати збір бонусів до

часового поясу. Команда обговорила ідею Дейлі бонусу, і тепер її можна розбити на підзадачі. Підзадач присвоюються мітки, що визначають тип роботи над завданням - програмування, графіка або гейм дизайн. Декомпозіруя епік, менеджер проекту вибудовує ієрархію завдань, які можуть робитися паралельно, і завдань, які виступають блокерами для інших. Наприклад завдання по скетчірованію персонажа є блокером до задачі по анімації персонажа.

Коли в беклоге накопичується достатня для обговорення кількість епіків з високою бізнес цінністю, продюсер проводити предпланірованіе, на якому розбиває пул завдань на підзадачі. Після предпланірованія гейм дизайнери беруться за детальний опис завдань, створюють документ, повністю пояснює як працює новий компонент гри, як виглядає, і як впливає на гру і її процеси. Як тільки опис готове і затверджено продюсером проекту, воно розсилається команді на ознайомлення, разом із запрошенням на планування.

На плануванні, гейм диз і продюсер доносять до команди мети і суть кожної нової задачі, після чого команда зачеплений питання, щоб краще зрозуміти, що ж повинно вийти на виході. Розібравшись з завданнями, кожен член команди дає свою оцінку. Оцінка може бути в робочих годинах або в деяких окулярах - Story Points. Оцінювати завдання потрібно тільки тоді, коли затверджено опис. Кожен член команди оцінює завдання виходячи зі свого досвіду і знань. Оцінюючи завдання, команда бере на себе відповідальність за терміни її виконання. Епіку присвоюється найвища складність всіх підзадач.

Після оцінки завдань Менеджер проекту може включати до складу спринту, враховуючи естімейти команди. Бізнес цінність завдання - це оцінка важливості завдання для поточної стадії проекту. Залежно від стадії проекту, одні і ті ж завдання мають різну бізнес цінність. Наприклад, якщо у гри користувачі відвалюються на туторіали, то завдання по контент апдейт взагалі не важлива і буде мати мінімальну оцінку. А у додатки з

мільярдним DAU, яке на ринку вже деякий час, контент апдейт буде безпосередньо впливати на доходи, відповідно завдання по ньому буде мати високу цінність.

Для оцінки цінності використовуємо той же «Ряд Фібоначчі», помножений на 1000, щоб візуально відрізнявся від оцінки складності. Залежно від сильних і слабких сторін проекту, цінність того чи іншого компонента може змінюватися. Наприклад додаток добре заробляє, але віральності жодної - в цьому випадку завдання на залучення матимуть цінність вище, ніж у завдань по монетизації.

Коли додаток ще не випущено, цінність завдання визначається її найближчими milestones і дорожнечею переробки завдання, в разі помилки. Так би мовити, в роботі є крапки не-повернення, такі як вибір архітектури або сеттинга мають більш високу цінність, ніж впровадження соціальних компонентів або обважень гри. Обвіс завжди можна переробити, і це не займе багато часу. А ось переробити архітектуру - це практично не реально і дуже дорого. Після того, як у кожного завдання в беклоге є оцінка складності і бізнес цінність, можна легко розставити пріоритети за наступним принципом. Спочатку в роботу йдуть завдання з найвищої бізнес цінність. Після цього в роботу йдуть завдання з найменшою складністю при однаковій бізнес цінності. Якщо в спринті залишилися заплановані 1-2 дня і не одне із завдань з високою цінністю не проходить за термінами, то в роботу йдуть завдання з меншою цінністю але підходящої складністі.

Плануючи роботу над проектом ми весь час сфокусовані на завданнях, які принесуть найбільшу цінність для бізнесу найменшими зусиллями.

Backlog ПП показаний на рис. 2.13. Розглянуто програмну систему з погляду користувача та розробника. Визначено важливість етапів та їх опис.

Ім'я	Важливість	"Story points"	Опис	Примітка
1) Графік	2	10	Як корстувач я хочу бачити графічне зображення заванажених даних, щоб оцінити коректність роботи.	Створення графічного поля.
2) GUI-інтерфейс	1	5	Як користувач я хочу увійти в систему, матиможливість завантажувати, генерувати та створювати самостійно дані для експерименту, зберігати дані	Створення зручного інтерфесу для внесення даних та збереження
3) Програмна частина	5	3	Як розробник я хочу реалізувати зручну програму для дослідження збіжності алгоритмів класифіції	Створення програмної системи

Рисунок 2.13. Backlog.

2.2.7. Методологія IDEF3

Для опису логіки взаємодії інформаційних потоків більше підходить IDEF3, звана також workflow diagramming, - методологія моделювання, що використовує графічний опис інформаційних потоків, взаємин між процесами обробки інформації та об'єктів, що є частиною цих процесів (рис. 2.14).

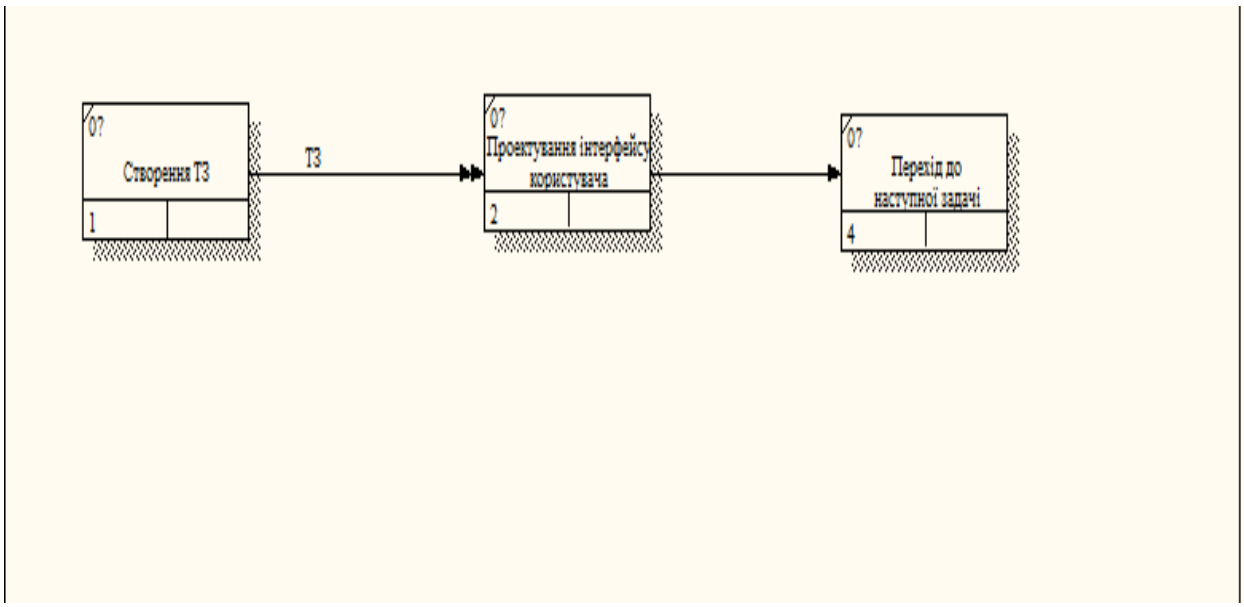

Рисунок 2.13. Діаграма IDEF3 (проектування інтерфейсу).

2.2.8. Методологія DFD

Найважливішим способом опису процесу є діаграми потоків даних (інформації) DFD (Data Flow Diagram). На рис.2.14. зображено декомпозицію пункту «Аналіз вимог». Зовнішнім посиланням тут виступає замовник, котрий надає свої потреби. Сховищем даних є репозиторій ГОСТів.

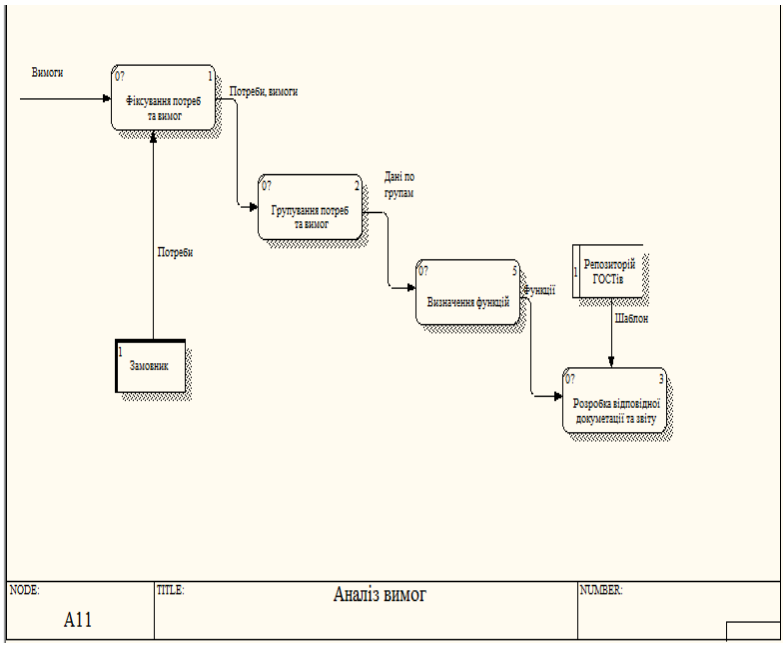

Рисунок 2.14. Діаграма DFD (аналіз вимог).

На рис. 2.15 зображено декомпозицію пункту «Аналіз аналогічних систем».

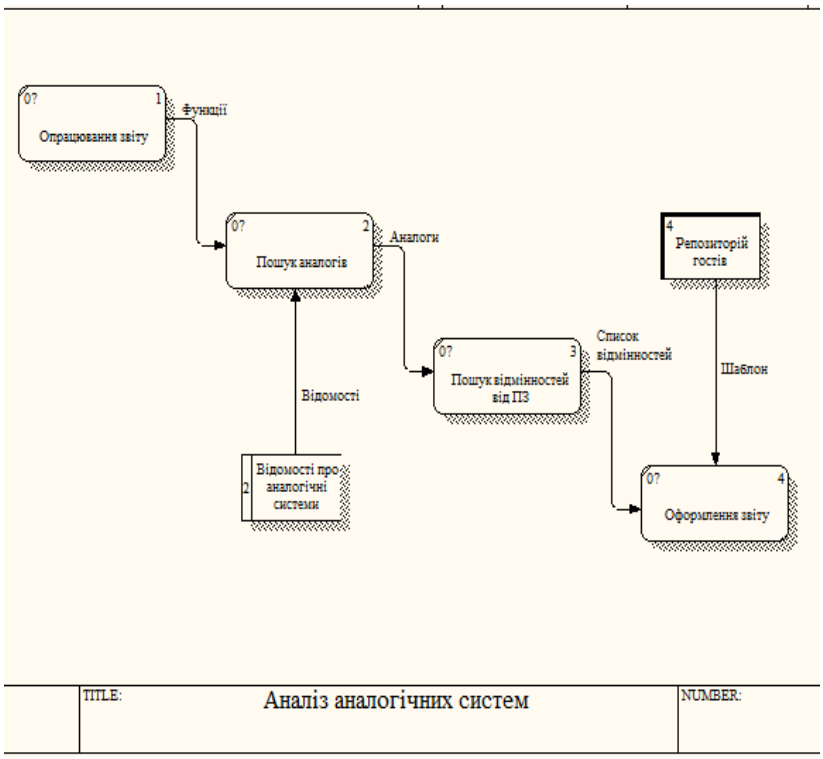

Рисунок 2.15. Діаграма DFD (аналіз аналогічних систем).

На рис. 2.16 зображена діаграма декомпозицій «Проектування», яка має такі основні блоки:

- проектування інтерфейсу;
- проектування архітектури.

Для проектування інтерфейсу необхідно мати затверджений дизайн замовником та створений на основі побажань замовника макет.

Під час проектування архітектури необхідно знати вимоги замовника та можливі сценарії роботи з програмною системою.

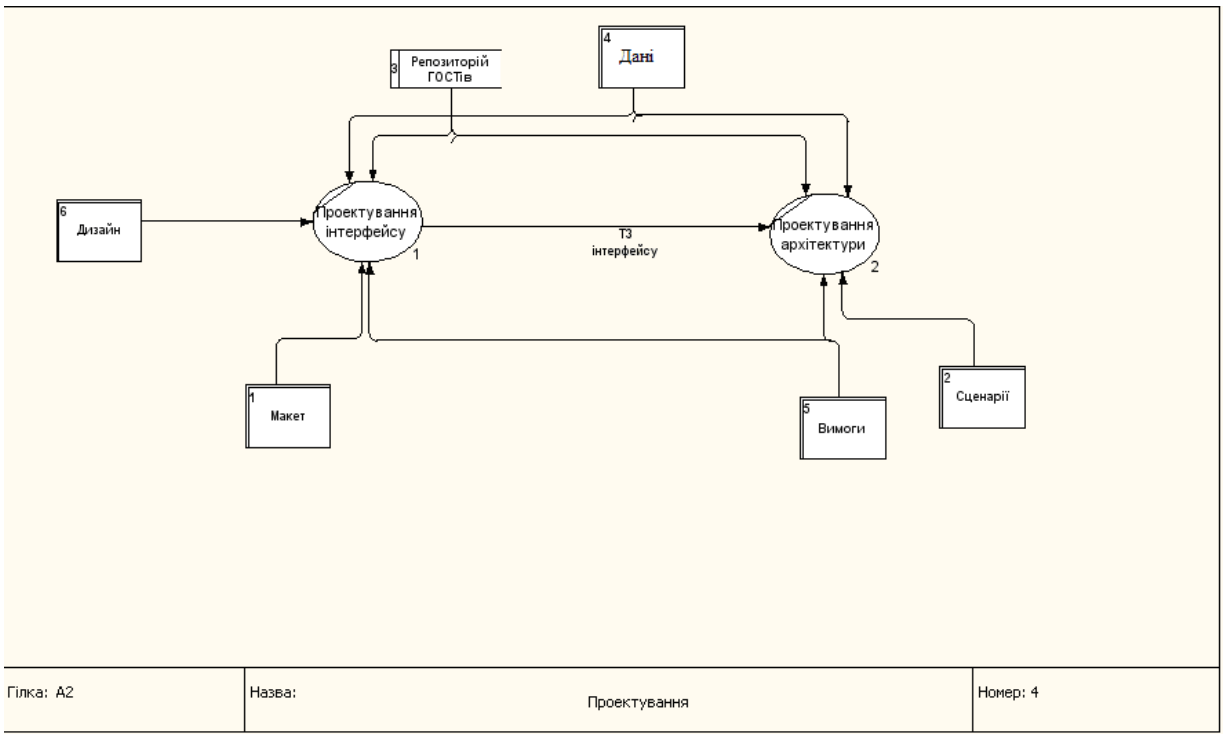

Рисунок 2.16. Діаграма декомпозиції «Проектування» (DFD).

Далі черга за реалізацією (рис. 2.17), яка складається з наступних блоків:

- реалізація інтерфейсу, модель котрого була побудована на етапі проектування;
- реалізація алгоритму лінійного регресійного аналізу;
- реалізація функціоналу роботи з базою даних, фільтрації вхідних даних;
- реалізація збереження даних у базу.

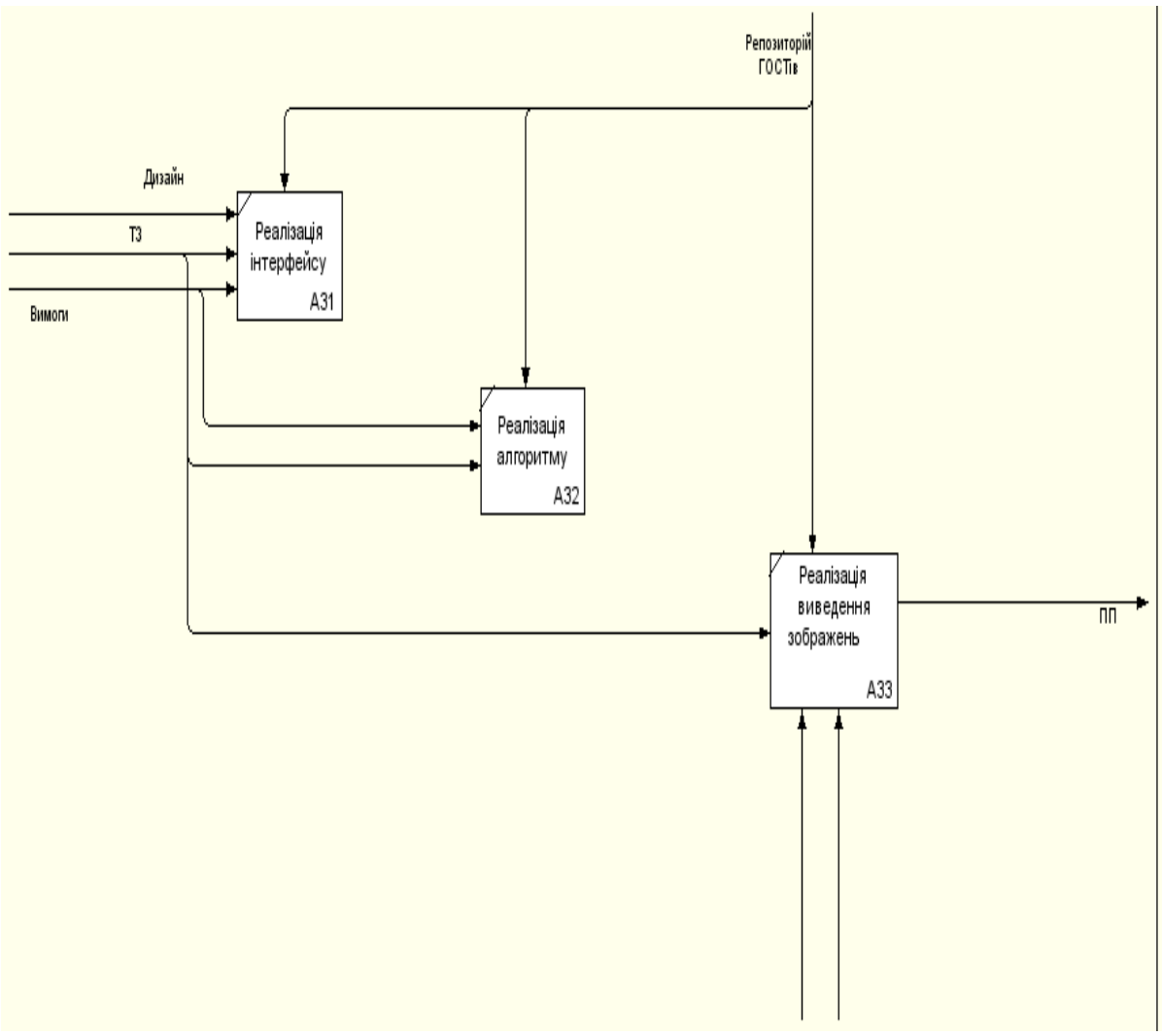

Рисунок 2.17. Діаграма декомпозиції «Реалізація» (IDEF0).

Чи не найважливіше місце в розробці програмної системи варто віддавати тестуванню (рис. 2.18). Оскільки необхідно створити якісну інструкцію з використання, протестувати інтерфейс та власне роботу системи. Крім того варто зробити гарний обробник помилок, особливо під час зчитування/запису файлу.

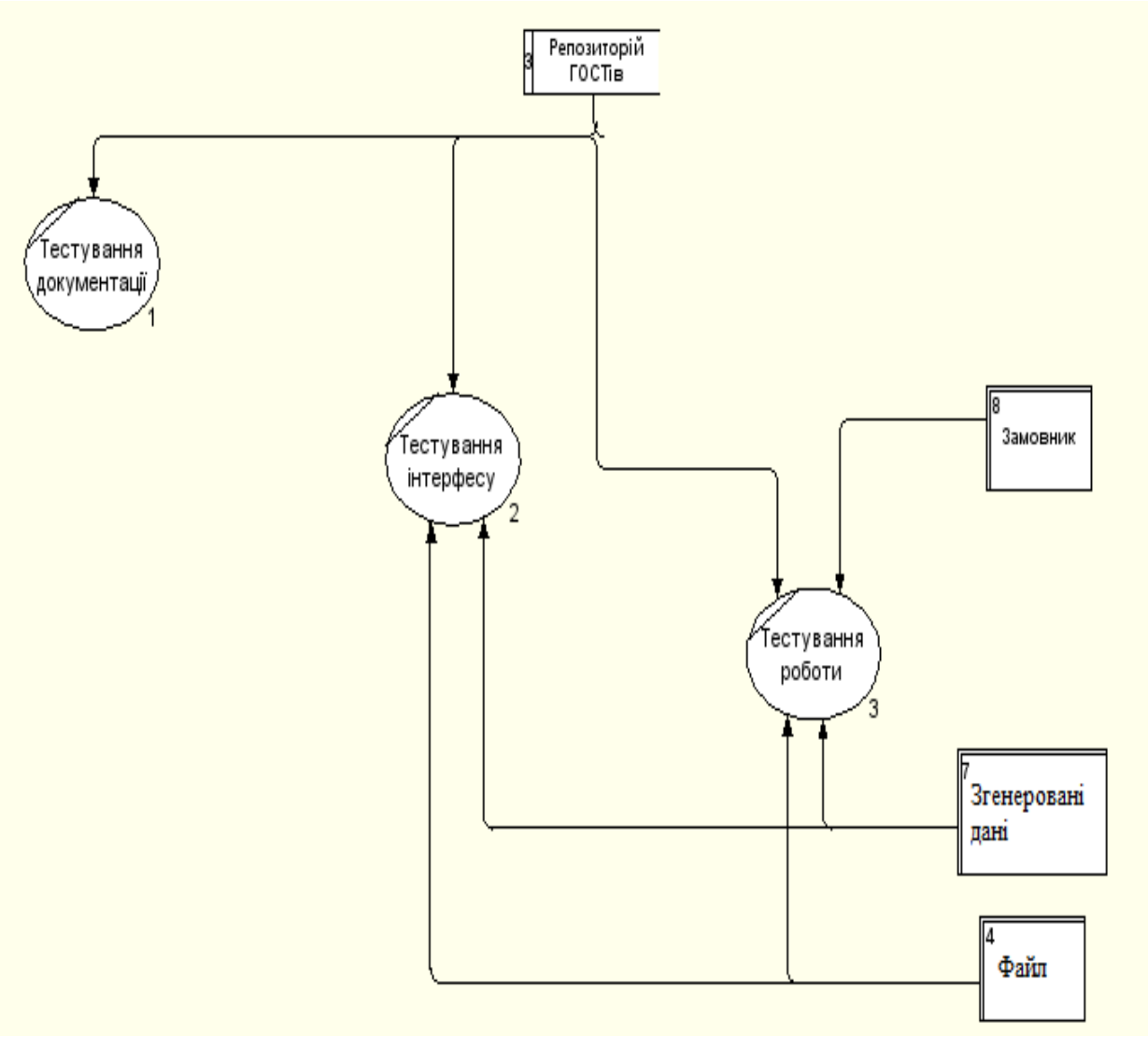

Рисунок 2.18. Діаграма декомпозиції «Тестування» (DFD).

2.2.9. Діаграма класів

Діаграми класів (Class diagrams) – головний тип діаграм UML, які відображають логічну структуру програмної системи та суттєво впливають на процес генерації програмного коду. Основними елементами діаграми класів у Rational Rose є безпосередньо класи (classes) та відношення між ними (relations).

Було виділено наступні класи: ПП, Алгоритми, Користувач, Вхідні дані, Вихідні дані. Їхні методи, атрибути та відношення між ними наведені на рис. 2.19.

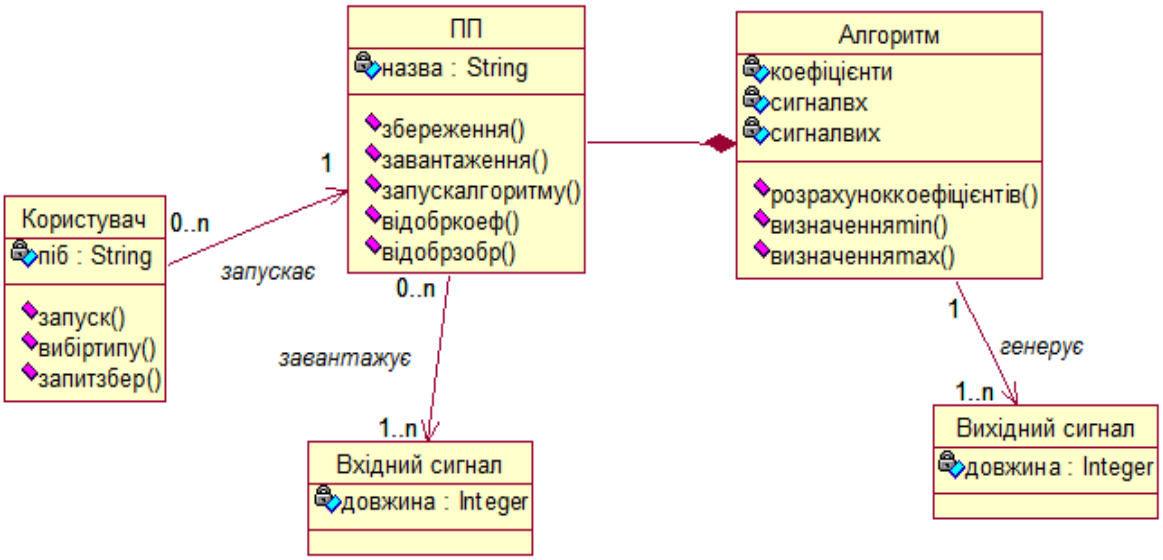

Рисунок 2.19. Діаграма класів.

2.2.10. Діаграми реалізації

Діаграма компонент — в UML, діаграма, на якій відображаються компоненти, залежності та зв'язки між ними.

Діаграма компонент (рис. 2.20) відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись. Модуль програмного забезпечення може бути представлено як компоненту. Деякі компоненти існують під час компіляції, деякі — під час компонування, а деякі під час роботи програми.

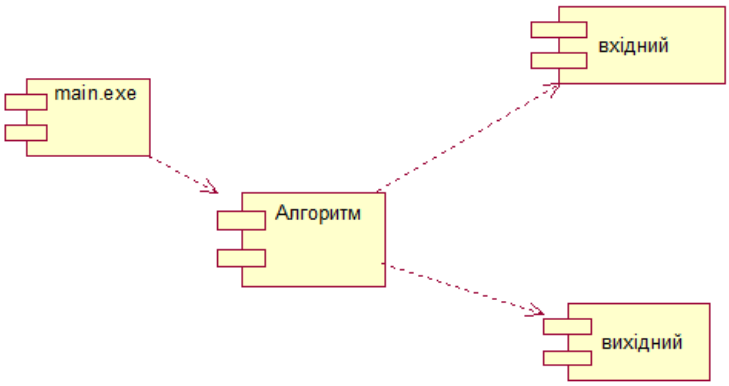

Рисунок 2.20. Діаграма компонентів.

2.2.11. Діаграма діяльності

Діаграми діяльності (activity diagrams) відображають послідовність дій, що виконується в процесі реалізації певного варіанта використання або функціонування системи в цілому (рис. 2.21).

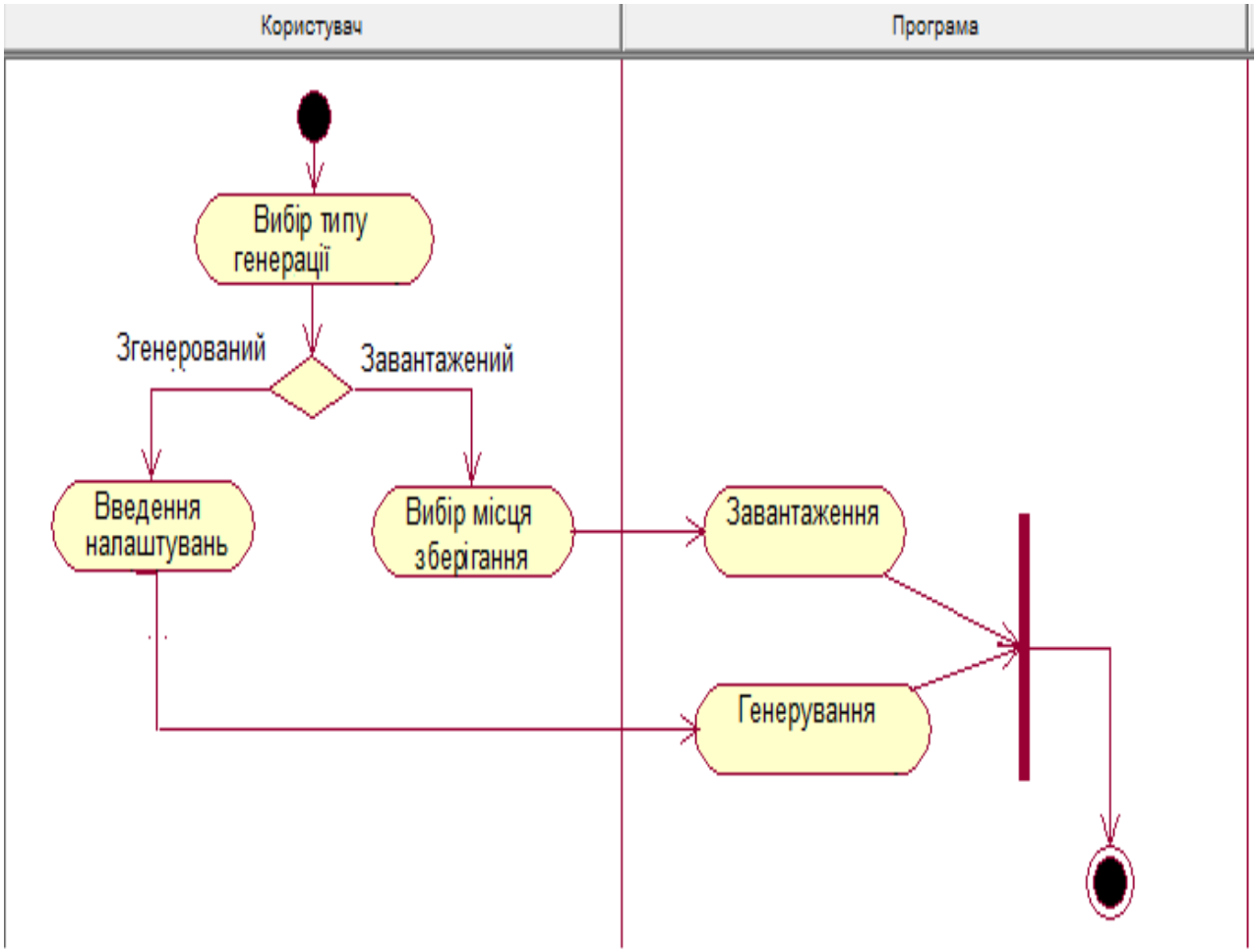

Рисунок 2.21. Діаграма діяльності «Вибір типу генерації».

Після запуску програми користувач обирає використовувати завантажену з файлу вибірку чи згенерувати її автоматично чи мануально. У разі використання генерованого користувач має налаштувати програму для кращої вибірки, після чого сигнал генерується програмою. Якщо ж вибрали завантажений, то програма завантажує файл. Після закінчення виконання операцій гілки поєднуються в один потік, в якому програма запускає алгоритми, алгоритми знаходять коефіцієнти прямих класифікації, генерує лінії. Програма виводить зображення (рис. 2.22).

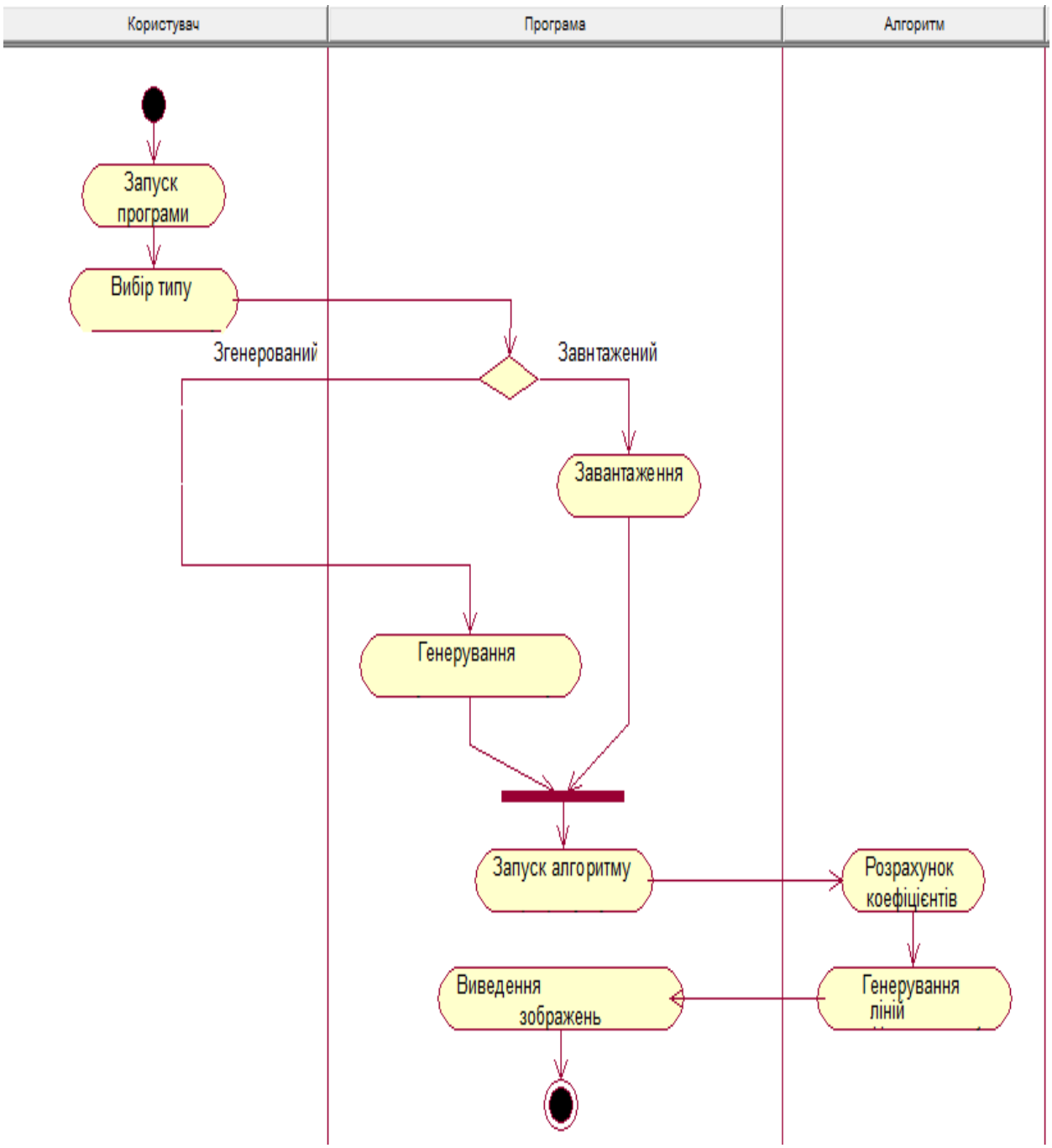

Рисунок 2.22. Діаграма діяльності ПП.

2.2.12. Діаграма дерева вузлів

Діаграма дерева вузлів показує ієрархічну залежність робіт, але не взаємозв'язки між роботами (рис. 2.23).

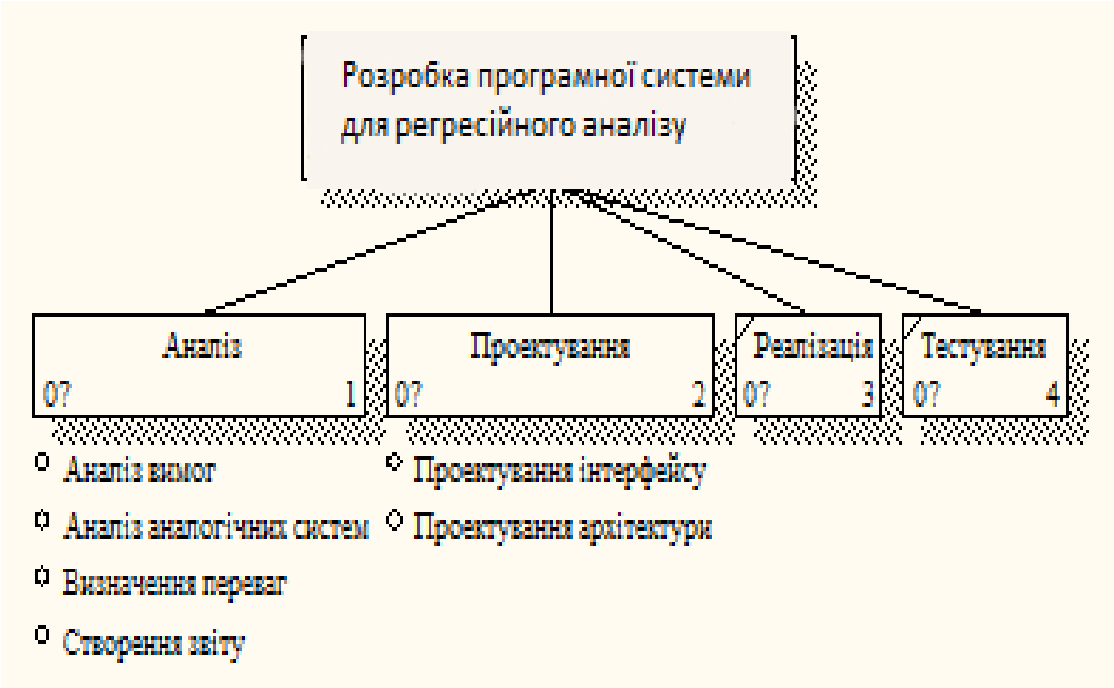

Рисунок 2.23. Діаграма дерева вузлів.

2.2.13. Діаграма варіантів

Діаграми варіантів використання (usecase diagrams) використовуються для відображення сценаріїв використання системи (usecases) та користувачів системи (actors), які використовують її функції.

Актором виступає користувач. Сценарії наступні: застосування фільтрів (filter data) чотирма різними способами:

• Поле пошуку (search field);

• Фільтр ділянки тіла (body part filter);

• Фільтр типу активності (activity type filter).

• Фільтр типу тиску (pressure type filter).

Під час запуску програми користувач має можливість сормувати вибірку даних для аналізу за допомогою чотирьох фільтрів (рис. 2.24).

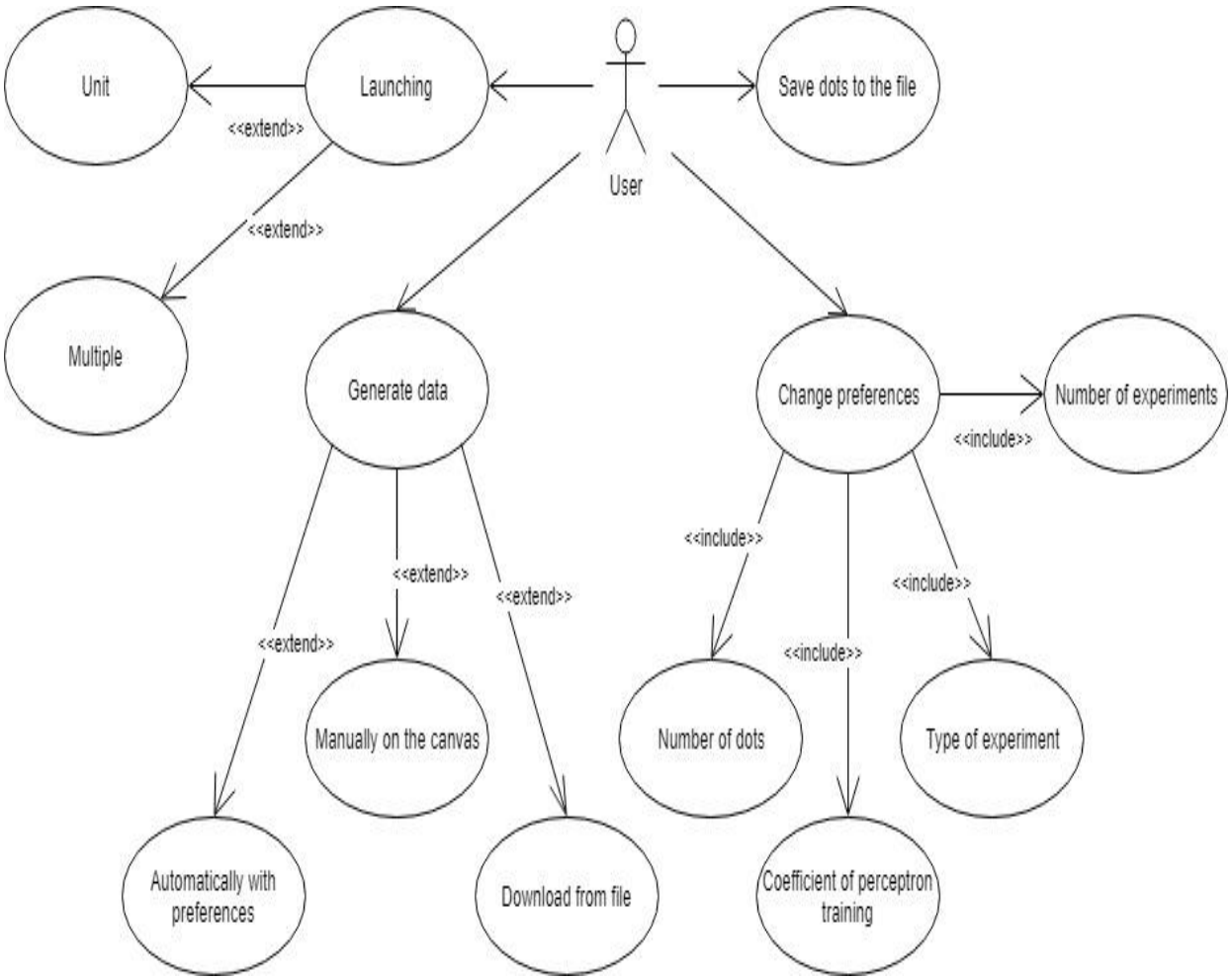

Рисунок 2.24. Use-case діаграма.

2.3. Реалізація програмного продукту

Розроблений програмний продукт написано мовою java з використанням IntelliJ IDEA 2017 community edition [39].

Метою розробки є створення програмної системи для регресійного аналізу параметрів артеріального тиску. Програмне забезпечення дозволить досліджувати взаємозв`язок між тиском у лівому плечі та тиском в інших ділянках тіла з метоюдослідження стану студентів на заняттях фізичного виховання.

Розроблений інтерфейс користувача є досить простим і інтуїтивно зрозумілим.

Вхідні дані при роботі з розробленою програмою:

- база даних;
- фільтри.

Основні можливості системи, що повинні бути розроблені:

- можливість завантаження бази даних у форматі .xls;
- можливість пошуку за ім`ям;
- можливість пошуку за номером кластеру;
- можливість обрати ділянку тіла на якій вимірювався тиск;
- можливість отбрати тип активності;
- можливість обрати тип тиску;
- розрахунок коефіцієнтів лінійної регресії;
- розрахунок коефіцієнту кореляції;
- запис результатів у базу даних;
- можливість використання програмного продукту на різних операційних системах.

Для реалізації програмного забезпечення було обрано IntelliJ IDEA 2017 community edition, тому що дане програмне забезпечення має необхідний інструментарій для виконання поставлених задач замовником.

Висновки до розділу 2

В ході проектування ПП була створена ієрархічна структура та розрахована нев’язка, яка свідчить про добре сплановану систему ПП, яку буде доцільно розробляти. Здійснено проектування програмного продукту у вигляді діаграм, які описують розробку та функціонування програми.

РОЗДІЛ 3
РОБОТА З РОЗРОБЛЕНИМ ПРОГРАМНИМ ПРОДУКТОМ

Після запуску програми з’являється інтерфейс представлений на рис. 3.1.

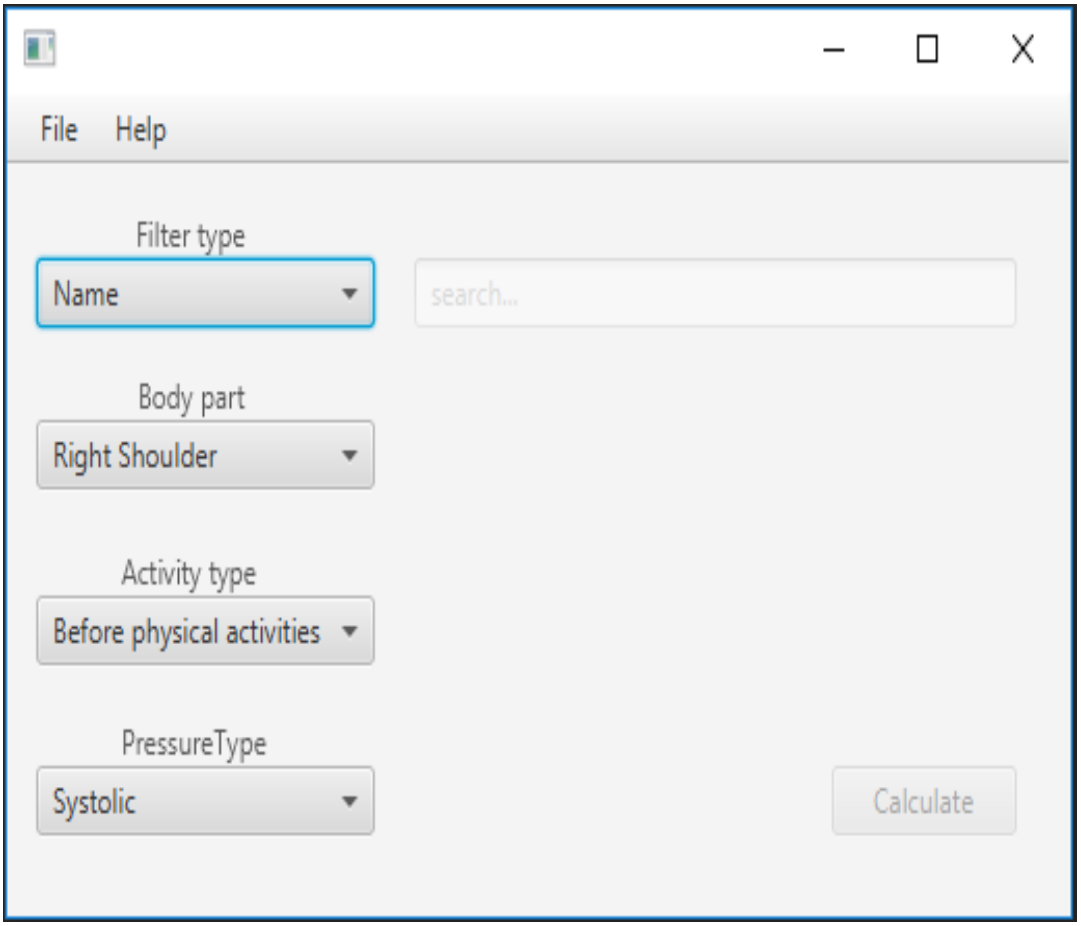

Рисунок 3.1. Інтерфейс програми.

Перш за все необхідно завантажити базу даних. В разі відсутності бази даних, або якщо файл не відповідає вимогам формату поле для пошуку та кнопка для розрахунку неактивні. Після вдалого завантаження бази даних користувач отримує сповіщення та може працювати з функціоналом фільтрації даних. Також варто зазначити, що файл з разширенням, яке не підтримується програмною системою завантажити неможливо. Програма працює із заздалегідь створеним шаблоном бази даних, який надається разом з програмним додатком і може бути

розширений користувачем. В разі невідповідності файлу вимогам, користувач отримає сповіщення та матиме можливість завантажити новий файл.

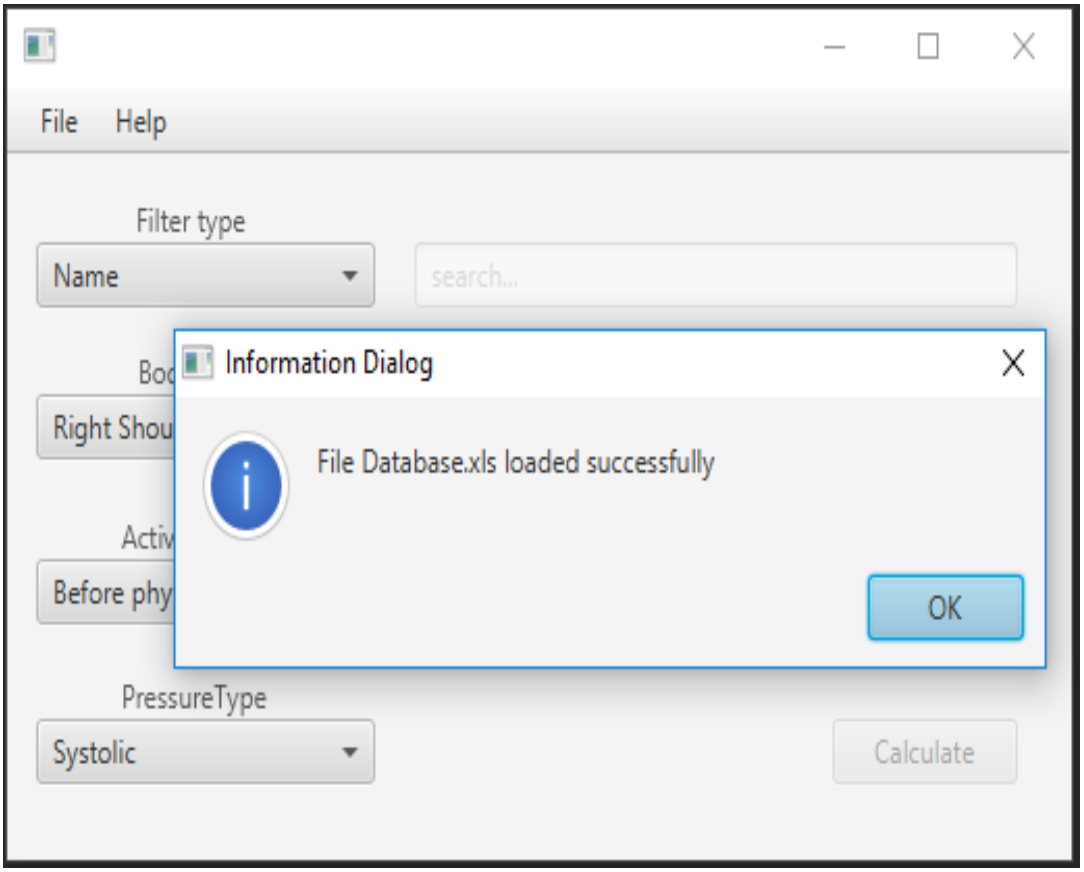

Рисунок 3.2. Вдале завантаження бази даних.

Після введення даних у поле для пошуку кнопка для розрахунку стає активною. Кнопка автоматично стає неактивною, якщо користувач видаляє введене значення з поля пошуку. Поле для пошуку дозволяє проводити пошук за ім`ям та за номером кластеру.

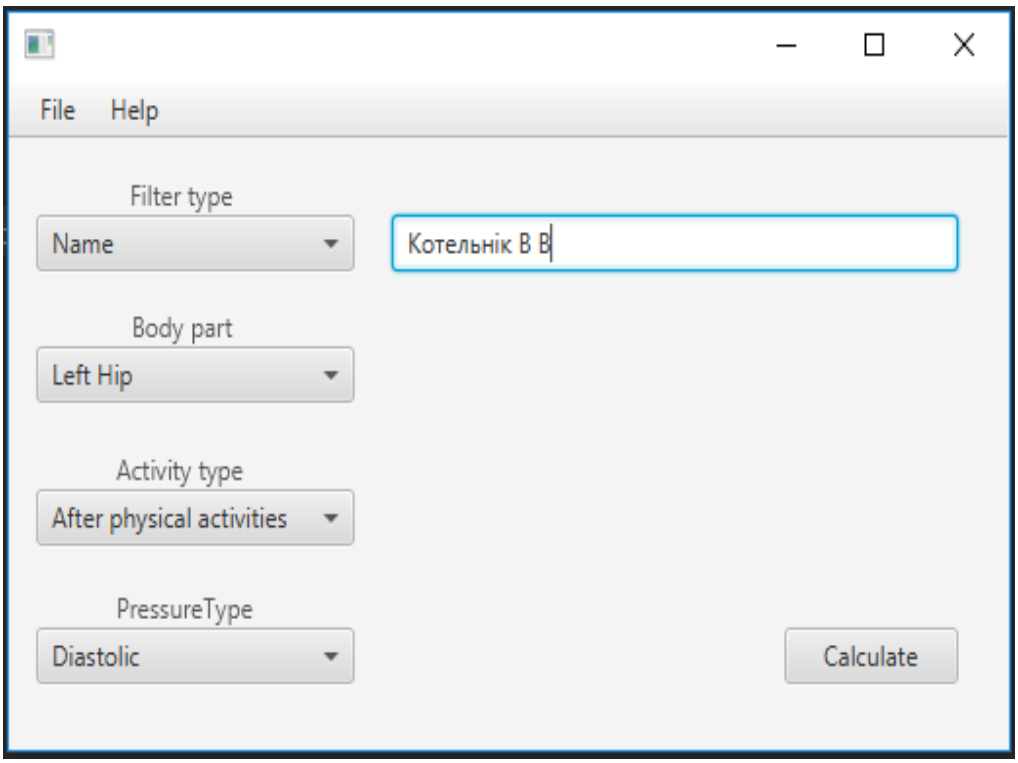

Рисунок 3.3. Застосування фільтрації за ім`ям.

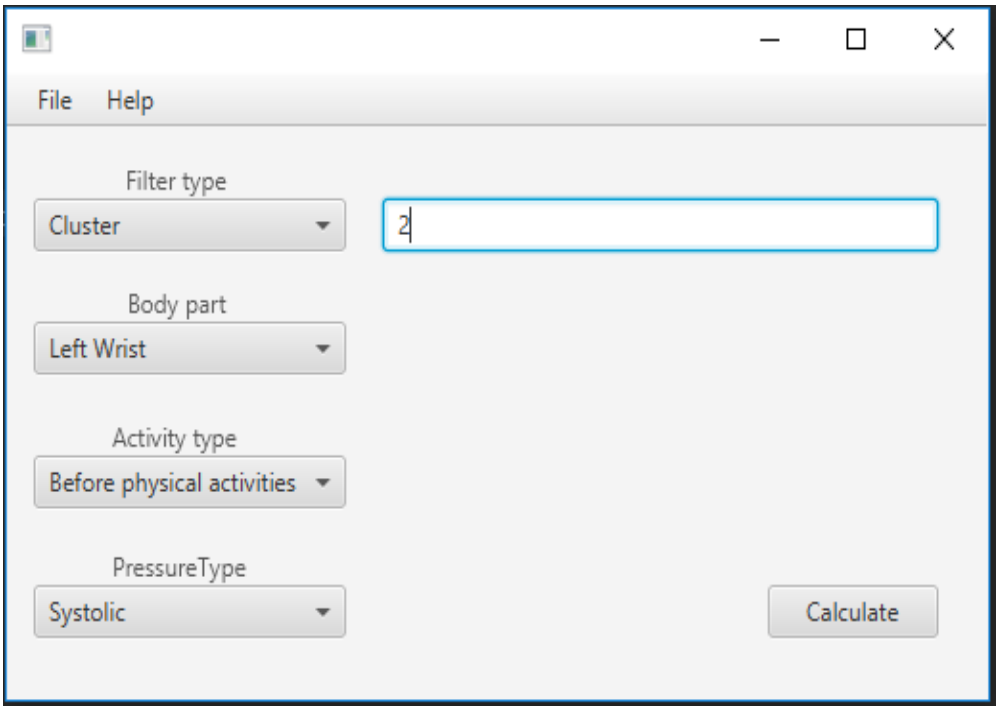

Рисунок 3.4. Застосування фільтрації за номером кластеру.

В програмній системі реалізовано функціонал валідації вхідних даних, який підтримує введення номеру кластеру цілими числами та символи крилиці та латиниці для фільтрації за ім’ям.

Після настискання кнопки “Calculate” програма проводить лінійний регресійний аналіз та сповіщує користувача про завершення розрахунків.

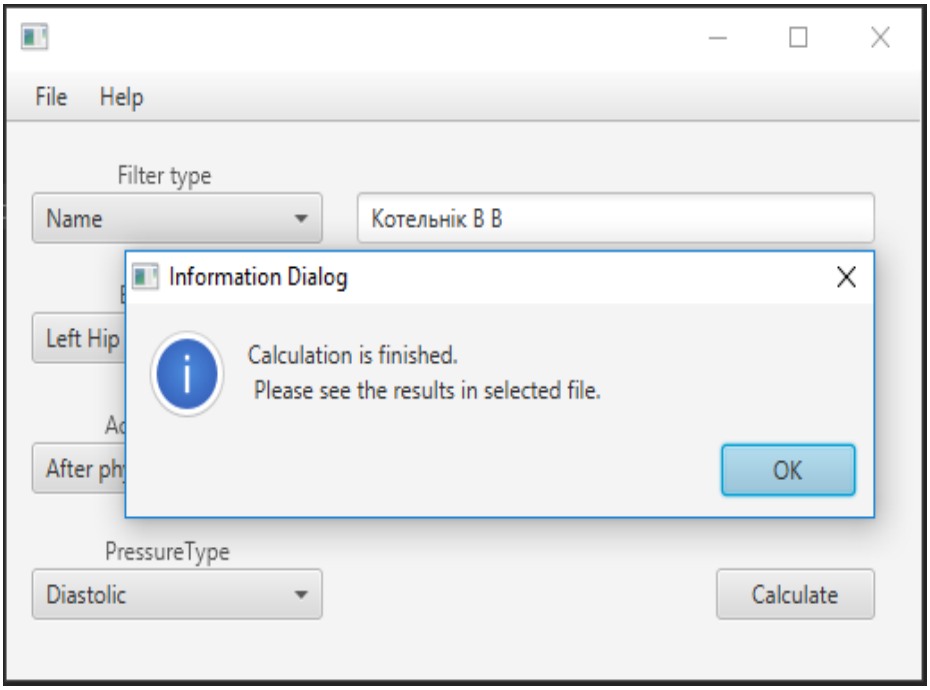

Рисунок 3.5. Сповіщення про успішне завершення розрахунків для фільтрації за ім`ям.

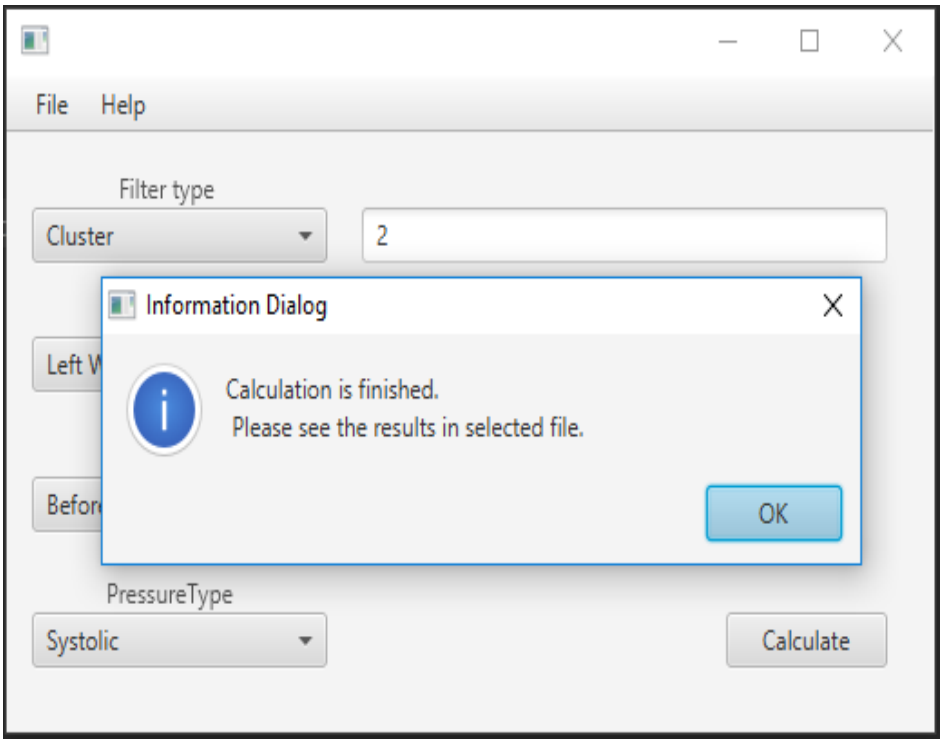

Рисунок 3.6. Сповіщення про успішне завершення розрахунків для фільтрації за номером кластеру.

Після завершення розрахунків результати записуються в базу даних, з якою працював користувач на новий лист. Лист містить параметри задані у фільтрах в назві для зручної навігації при наявності декількох розрахунків.

A	B	C	D
Searched value	Body part	Pressure type	Activity type
2	Left Wrist	Systolic	Before physical activities
Equation	R^2	Std. error of a	Std. error of b
y = 0.7619082731160595 * x +37.87383866686331	0.2824626214878097	0.2532098193731967	32.30334132572973
	SSTO(total sum of squares)	SSE(error sum of squares)	SSR(regression sum of squares)
	3344.64	2399.904217667011	944.7357823329878

Лист1 | Котельнік В В Left Hip Diastoli | 2 Left Wrist Systolic Before ph

Рисунок 3.7. Результати роботи програми при фільтрації за номером кластеру.

A	B	C	D
Searched value	Body part	Pressure type	Activity type
Мороз Ю А	Right Shin	Systolic	After physical activities
Equation	R^2	Std. error of a	Std. error of b
y = 2.1 * x -153.16666666666669	0.4914561664190193	2.1361959960016152	267.1668746100251
	SSTO(total sum of squares)	SSE(error sum of squares)	SSR(regression sum of squares)
	448.6666666666667	228.16666666666663	220.5

Лист1 | Мороз Ю А Right Shin Systolic A | 2 Right Hip Diastolic Before ph

Рисунок 3.8. Результати роботи програми при фільтрації за ім`ям.

Висновки до розділу 3

В розділі детально описано всі можливі маніпуляції з програмою. Наведено знімки програми з поясненнями для кращого розуміння можливих дій. Розглянуто вибір даних для алгоритмів, запуск експерименту та налаштування програми.

РОЗДІЛ 4
РЕЗУЛЬТАТИ СТАТИСТИЧНИХ ЕКСПЕРИМЕНТІВ

Розглянемо результати, отримані при експериментальному дослідженні параметрів артеріального тиску студента Мороз Ю.А.

A	B	C	D
Searched value	Body part	Pressure type	Activity type
Мороз Ю А	Right Shin	Systolic	After physical activities
Equation	R^2	Std. error of a	Std. error of b
y = 2.1 * x -153.16666666666669	0.4914561664190193	2.1361959960016152	267.1668746100251
	SSTO(total sum of squares)	SSE(error sum of squares)	SSR(regression sum of squares)
	448.6666666666667	228.16666666666663	220.5

Лист1 | Мороз Ю А Right Shin Systolic A | 2 Right Hip Diastolic Before ph

Рисунок 4.1. Результати розрахунків для студента Мороз Ю.А.

Для даного студента проводилося вимірювання тиску чотири рази. В даному експерименті була поставлена мета встановити взаємозв`язок між значенням тиску у лівому плечі та правій гомілці. Аналіз проводився для систолічного тиску одразу після навантаження. В результаті розрахунків було отримане рівняння лінійної регресії, розраховане значення R^2, розраховані стандартні помилки для коефіцієнтів a, b. Також була обчислена загальа сума квадратів, регресійна сума квадратів та помилка суми квадратів.

Коефіцієнт кореляції Пірсона(позначають «r») — в статистиці, показник кореляції (лінійної залежності) між двома змінними X та Y, який набуває значень від −1 до +1 включно. Він широко використовується в

науці для вимірювання ступеня лінійної залежності між двома змінними. Показник був розроблений Карлом Пірсоном (Karl Pearson) зі схожої ідеї, представленої Френсісом Гальтоном в 1880-х рр. Різні автори пропонують різні підходи до інтерпретації значення коефіцієнта кореляції.

В той же час, всі критерії є певною мірою умовними, і не повинні трактуватися надто прискіпливо. Інтерпретація кореляції залежить від контексту та мети. Наприклад, показник кореляції 0.9 може бути дуже низьким у випадку дослідження законів фізики з використанням високоякісного обладнання, проте може трактуватися як дуже високий в гуманітарних науках, де існує вплив багатьох інших факторів.

Кореляція	Негативна	Позитивна
Відсутня	−0.09 до 0.0	0.0 до 0.09
Низька	−0.3 до −0.1	0.1 до 0.3
Середня	−0.5 до −0.3	0.3 до 0.5
Висока	−1.0 до −0.5	0.5 до 1.0

Рисунок 4.2. Значення показників кореляції.

За результатами розрахунків R^2 = 0.49, отже r = 0.7, що свідчить про досить високе значення лінійної залежності показників артеріального тиску в лівому плечі та в правій гомілці.

Розглянемо результати, отримані при експериментальному дослідженні параметрів артеріального тиску для кластеру №2.

A	B	C	D
Searched value	Body part	Pressure type	Activity type
2	Left Wrist	Systolic	Before physical activities
Equation	R^2	Std. error of a	Std. error of b
y = 0.7619082731160595 * x +37.87383866686331	0.2824626214878097	0.2532098193731967	32.30334132572973
	SSTO(total sum of squares)	SSE(error sum of squares)	SSR(regression sum of squares)
	3344.64	2399.904217667011	944.7357823329878

Лист1 | Котельнік В В Left Hip Diastoli | 2 Left Wrist Systolic Before ph

Рисунок 4.3. Результати розрахунків для кластеру №2.

Для даного кластера в базі зафіксовано 25 вимірювань тиску. В даному експерименті була поставлена мета встановити взаємозв`язок між значенням тиску у лівому плечі та лівому зап`ясті. Аналіз проводився для систолічного тиску у стані спокою. В результаті розрахунків було отримане рівняння лінійної регресії, розраховане значення R^2, розраховані стандартні помилки для коефіцієнтів a, b. Також була обчислена загальа сума квадратів, регресійна сумквадратів та помилка суми квадратів.

За результатами розрахунків R^2 = 0.28, отже r = 0.53, що свідчить про досить високе значення лінійної залежності показників артеріального тиску в лівому плечі та в лівому зап`ясті.

Також було проведено дослідження окремо для жінок та чоловіків до та після навантаження. Для цього база даних була розбита на 2 кластери за полем стать. Чоловіки – кластер №1, жінки – кластер № 2. Нижче наведені результати, отримані в ході дослідження взаємозв’язку між показниками

тиску в референтній точці (ліве плече) та показниками тиску в правому стегні.

На рисунках зображені результати аналізу показників систолічного тиску для жінок у стані спокою.

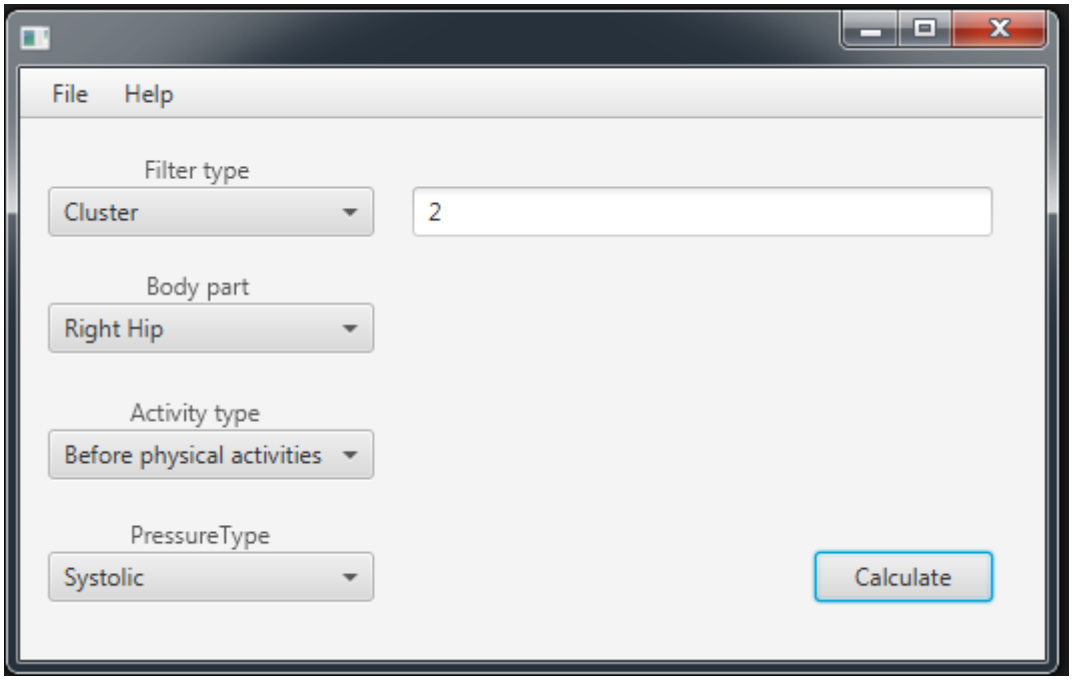

Рисунок 4.4. Налаштування фільтрів для аналізу систолічного тиску жінок у стані спокою.

A	B	C	D
Searched value	Body part	Pressure type	Activity type
2	Right Hip	Systolic	Before physical activities
Equation	R^2	Std. error of a	Std. error of b
y = 0.06426678445229674 * x +111.2126766784452	0.002347092655072581	0.34210983566722103	39.33213016573028
	SSTO(total sum of squares)	SSE(error sum of squares)	SSR(regression sum of squares)
	1406.1176470588236	1402.817358657244	3.3002884015797047

Рисунок 4.5. Результати розрахунків аналізу систолічного тиску жінок у стані спокою.

На рисунках зображені результати аналізу показників діастолічного тиску для жінок у стані спокою.

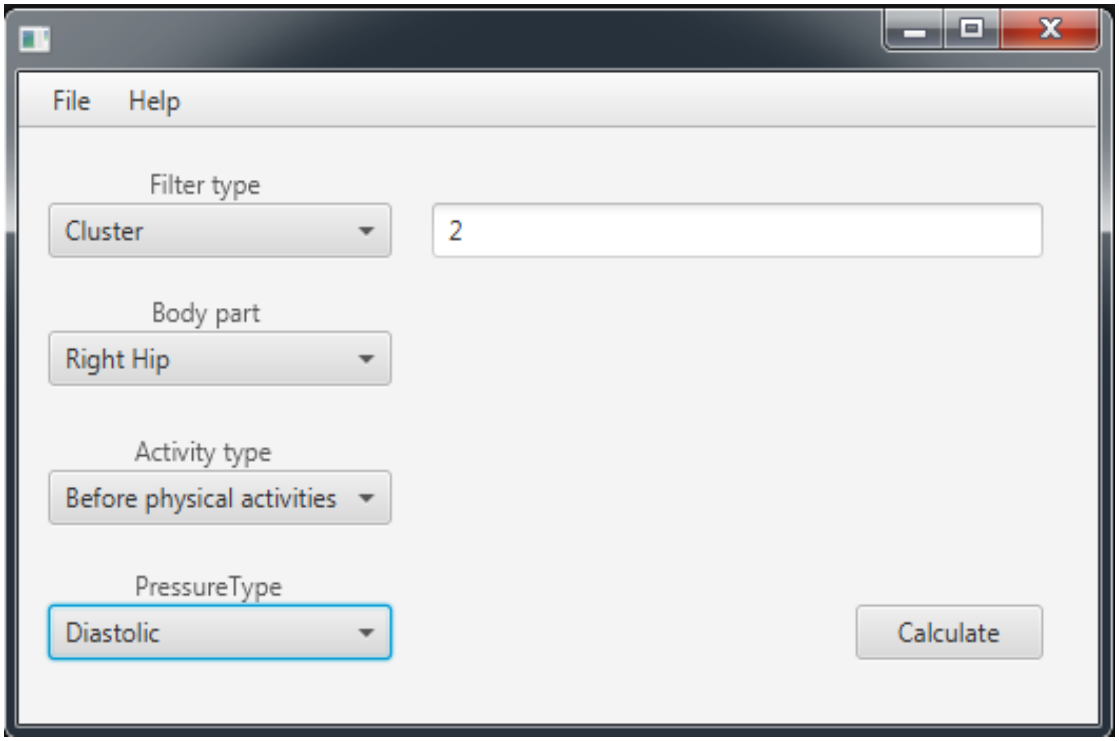

Рисунок 4.6. Налаштування фільтрів для аналізу діастолічного тиску жінок у стані спокою.

A	B	C	D
Searched value	Body part	Pressure type	Activity type
2	Right Hip	Diastolic	Before physical activities
Equation	R^2	Std. error of a	Std. error of b
y = 0.643029087261785 * x +19.547141424272844	0.32353315636754854	0.24007605673233304	16.432500220010525
	SSTO(total sum of squares)	SSE(error sum of squares)	SSR(regression sum of squares)
	1499.0588235294117	1014.0635907723172	484.9952327570945

Рисунок 4.7. Результати розрахунків аналізу діастолічного тиску жінок у стані спокою.

На рисунках зображені результати аналізу показників систолічного тиску для жінок після фізичного навантаження.

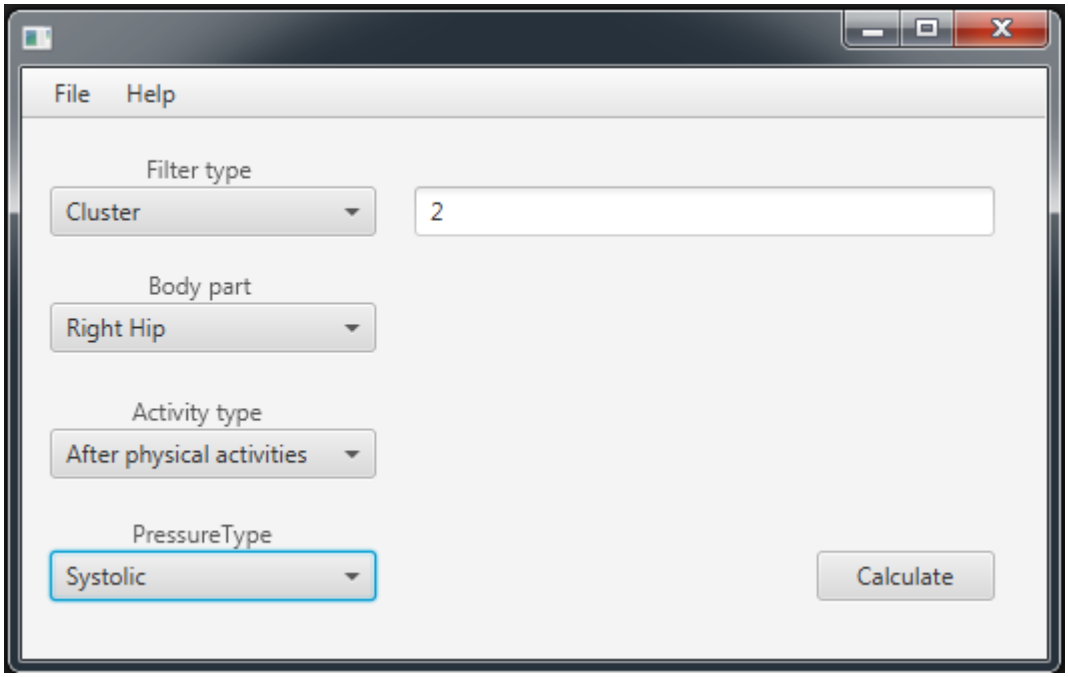

Рисунок 4.8. Налаштування фільтрів для аналізу систолічного тиску жінок після фізичного навантаження.

Searched value	Body part	Pressure type	Activity type
2	Right Hip	Systolic	After physical activities
Equation	R^2	Std. error of a	Std. error of b
y = 0.2645026724025550 4 * x + 85.80615304393169	0.022207892175154557	0.45316256721530507	53.94750983908717
	SSTO(total sum of squares)	SSE(error sum of squares)	SSR(regression sum of squares)
	2843.0588235294117	2779.9204797288485	63.13834380056294

Рисунок 4.9. Результати розрахунків аналізу систолічного тиску жінок після фізичного навантаження.

На рисунках зображені результати аналізу показників діастолічного тиску для жінок після фізичного навантаження.

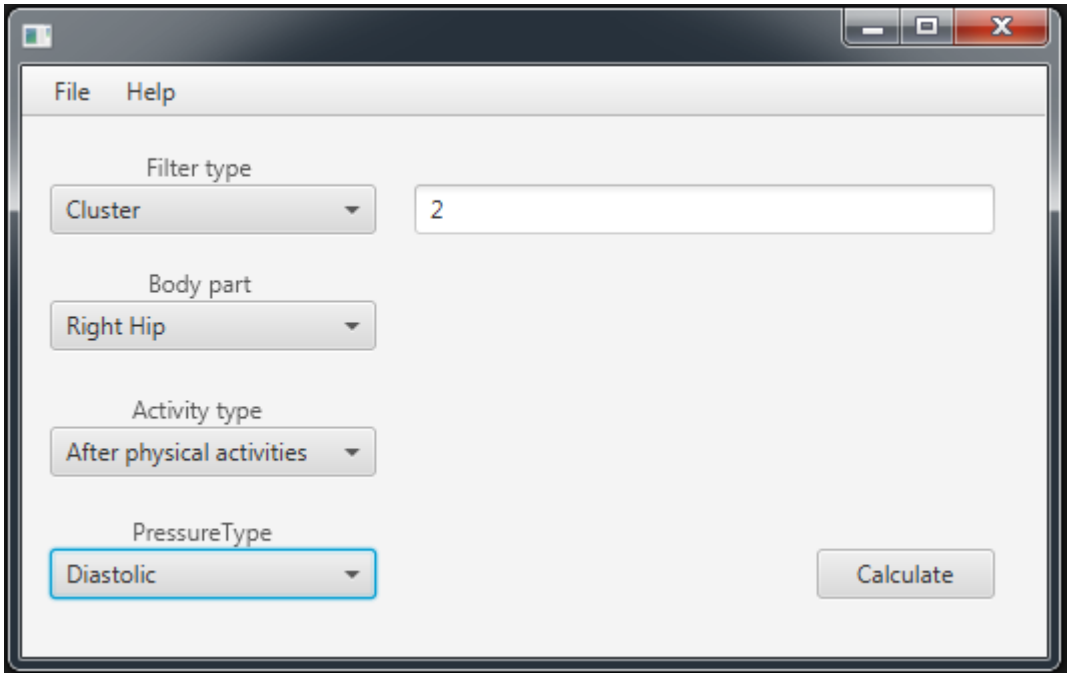

Рисунок 4.10. Налаштування фільтрів для аналізу діастолічного тиску жінок після фізичного навантаження.

A	B	C	D
Searched value	Body part	Pressure type	Activity type
2	Right Hip	Diastolic	After physical activities
Equation	R^2	Std. error of a	Std. error of b
y = 0.033058852022281406 * x +55.5191329619762	0.0018042916037750525	0.2007689853134091	14.076839517791463
	SSTO(total sum of squares)	SSE(error sum of squares)	SSR(regression sum of squares)
	588.4705882352943	587.4088156938726	1.0617725414215076

Рисунок 4.11. Результати розрахунків аналізу дістолічного тиску жінок після фізичного навантаження.

На рисунках зображені результати аналізу показників систолічного тиску для чоловіків у стані спокою.

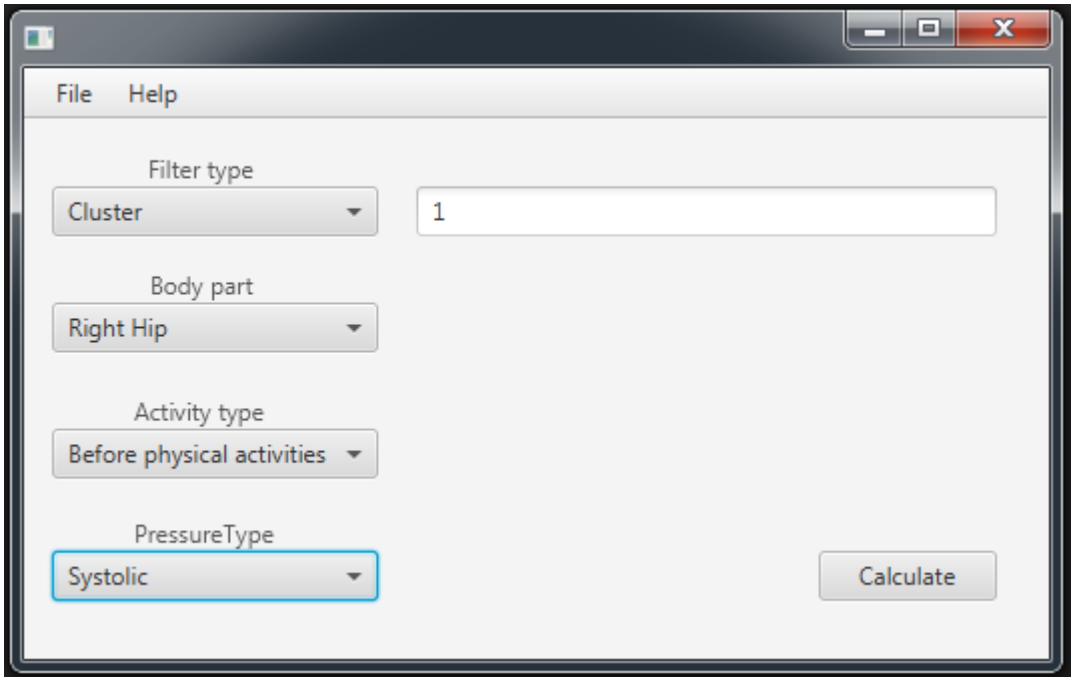

Рисунок 4.12. Налаштування фільтрів для аналізу систолічного тиску чоловіків у стані спокою.

A	B	C	D
Searched value	Body part	Pressure type	Activity type
1	Right Hip	Systolic	Before physical activities
Equation	R^2	Std. error of a	Std. error of b
y = 0.6665815269250416 * x +41.24304825977583	0.22508008949464403	0.15460486542342444	19.53390495535138
	SSTO(total sum of squares)	SSE(error sum of squares)	SSR(regression sum of squares)
	12413.0303030303	9619.104331524513	2793.9259715057883

Рисунок 4.13. Результати розрахунків аналізу систолічного тиску чоловіків у стані спокою.

На рисунках зображені результати аналізу показників діастолічного тиску для чоловіків у стані спокою.

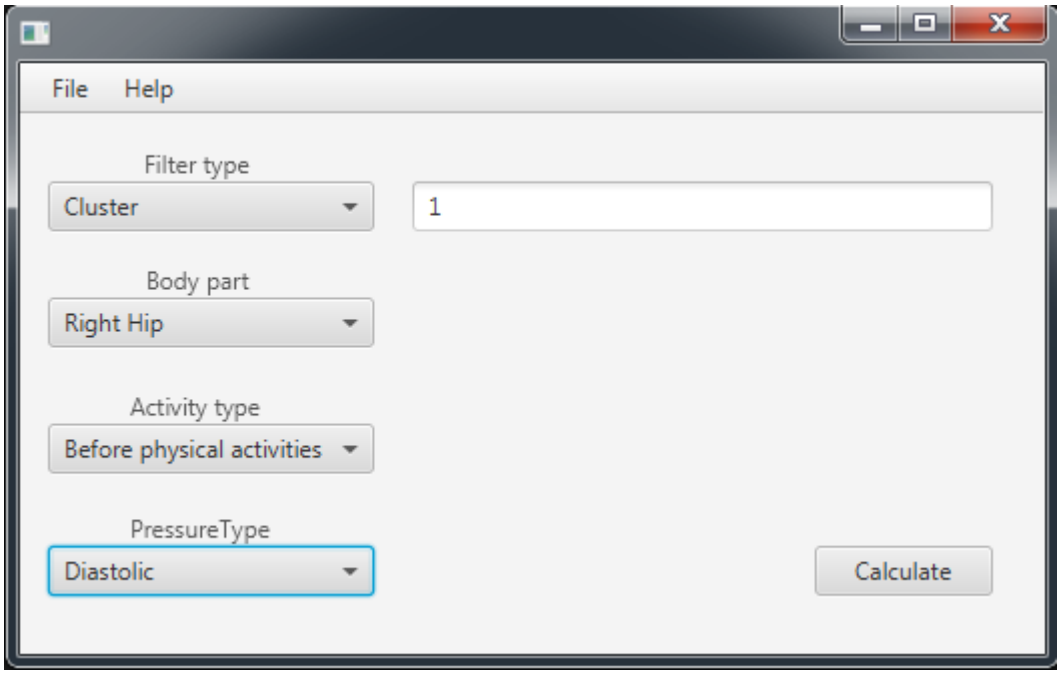

Рисунок 4.14. Налаштування фільтрів для аналізу діастолічного тиску чоловіків у стані спокою.

A	B	C	D
Searched value	Body part	Pressure type	Activity type
1	Right Hip	Diastolic	Before physical activities
Equation	R^2	Std. error of a	Std. error of b
y = 0.49294002516426666 * x +24.37703188744710	0.275080165400858	0.10002750790110009	6.990798936497151
	SSTO(total sum of squares)	SSE(error sum of squares)	SSR(regression sum of squares)
	3159.272727272727	2290.2194629081246	869.0532643646015

Рисунок 4.15. Результати розрахунків аналізу діастолічного тиску чоловіків у стані спокою.

На рисунках зображені результати аналізу показників систолічного тиску для чоловіків після фізичного навантаження.

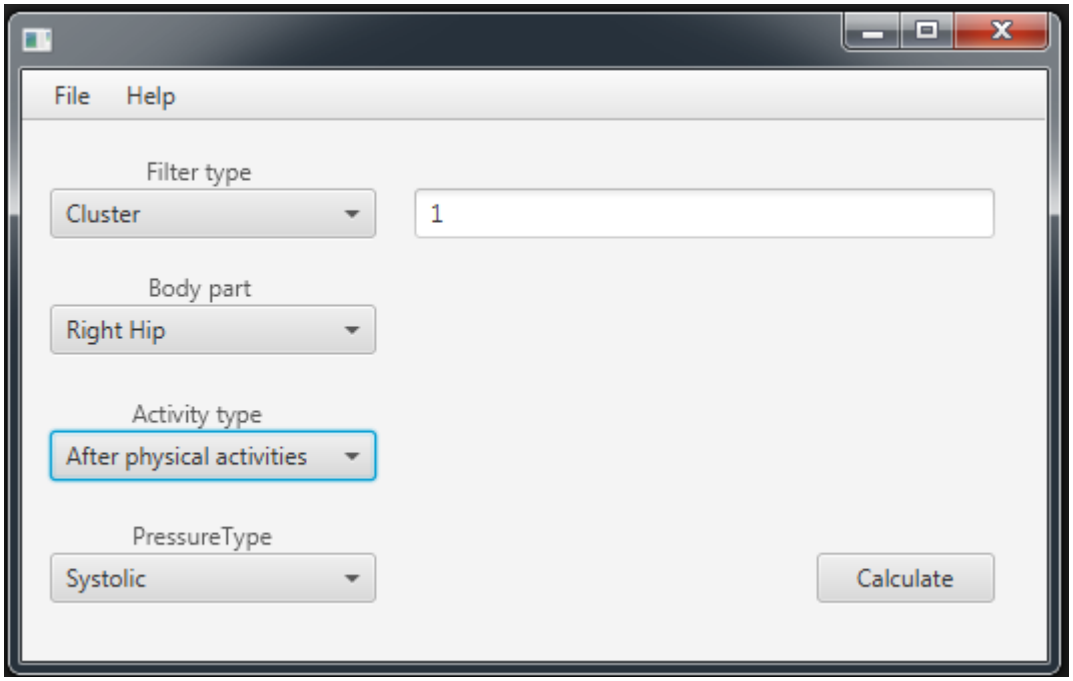

Рисунок 4.16. Налаштування фільтрів для аналізу систолічного тиску чоловіків після фізичного навантаження.

A	B	C	D
Searched value	Body part	Pressure type	Activity type
1	Right Hip	Systolic	After physical activities
Equation	R^2	Std. error of a	Std. error of b
y = 0.4759884151680477 * x +71.67158348487906	0.0651515385375441	0.22537963262476915	29.73217817605241
	SSTO(total sum of squares)	SSE(error sum of squares)	SSR(regression sum of squares)
	28162.12121212122	26327.315686670707	1834.8055254505039

Рисунок 4.17. Результати розрахунків аналізу систолічного тиску чоловіків після фізичного навантаження

На рисунках зображені результати аналізу показників діастолічного тиску для чоловіків після фізичного навантаження.

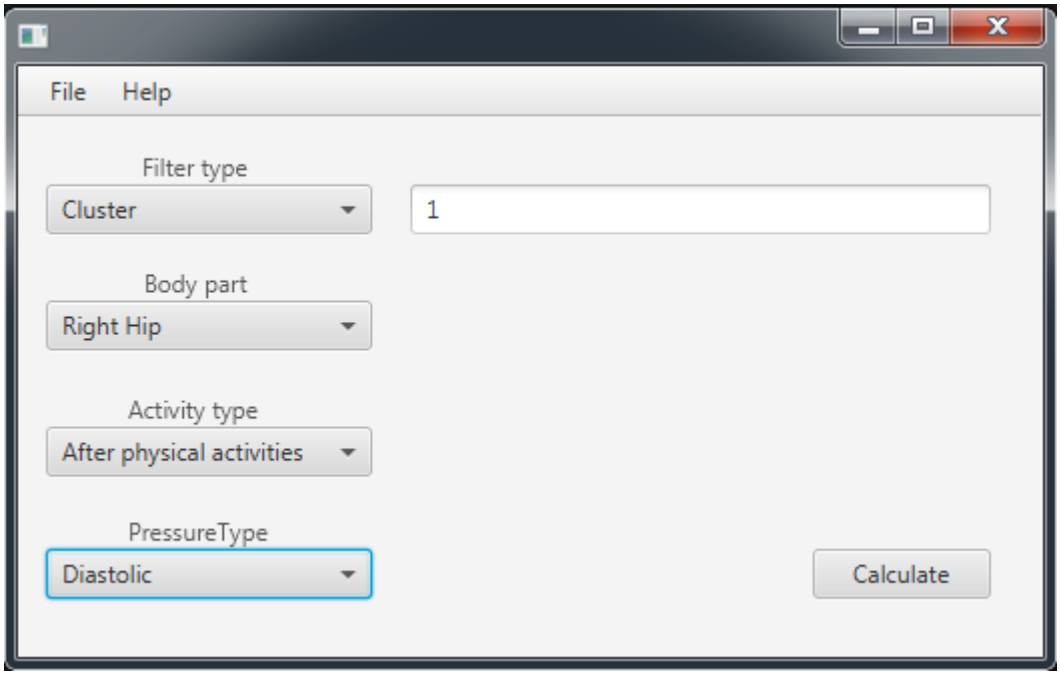

Рисунок 4.18. Налаштування фільтрів для аналізу діастолічного тиску чоловіків після фізичного навантаження.

A	B	C	D
Searched value	Body part	Pressure type	Activity type
1	Right Hip	Diastolic	After physical activities
Equation	R^2	Std. error of a	Std. error of b
y = 0.2010285524690789 * x +45.918941586330206	0.01985269418887505	0.17656462945977075	12.416870072704654
	SSTO(total sum of squares)	SSE(error sum of squares)	SSR(regression sum of squares)
	6692.984848484849	6560.1110670771	132.87378140774393

Рисунок 4.19. Результати розрахунків аналізу дістолічного тиску чоловіків після фізичного навантаження.

У підсумку маємо значення коефіцієнтів детермінації та коефіцієнтів лінійної регресії для всіх досліджень.

Жінки до навантаження:

Систолічний тиск:

y = -0.21033469596320903 * x +142.47023505365357

R^2 = 0.03224544477685507

Діастолічний тиск:

y = 0.5855978260869563 * x +22.615489130434796

R^2 = 0.3295899546418025

Жінки після навантаження:

Систолічний тиск:

y = 0.7498508887033281 * x +29.4736967672671

R^2 = 0.13804951635394533

Діастолічний тиск:

y = 0.12916860195076635 * x +48.53516024152346

R^2 = 0.03321171246533664

Чоловіки до навнтаження:

Систолічний тиск:

y = 0.7061939502082155 * x +36.36094060332607

R^2 = 0.24890989625497414

Діастолічний тиск:

y = 0.5033311505882851 * x +23.86707082655625

R^2 = 0.25600280310033896

Чоловіки після навантаження:

Систолічний тиск:

y = 0.47169721590670055 * x +71.8446003661885

R^2 = 0.06589082165599175

Діастолічний тиск:

y = 0.16915698398101905 * x +48.216827804845835

R^2 = 0.0128970013113166

Аналізуючи результати, можна сказати, що в стані спокою взаємозв’язок між показниками тиску в лівому плечі та показниками тиску

в правому стегні майже відсутній. При помірному навантаженні, ситуація не змінюється, що свідчить про те, що організм добре адаптований під навантаження такої інтенсивності. При сильному навантаженні можна спостерігати кореляцію між показниками тиску в даних точках. Це свідчить про те, що організм не пристосований під навантаження такої інтенсивності. Таким чином, можна індивідуально для кожної людини встановити гранично допустиму інтенсивність навантаження, при якій організму не буде завдано шкоди.

Висновки до розділу 4

В розділі детально описано всі проведені експерименти, що дозволили виявити взаємозв`язок між параметрами артеріального тиску у лівому плечі та інших точках верхніх та нижніх кінцівок. Дослідження проводилися у стані спокою та після навантаження. Отримані результати дозволять встановити гранично допустиму інтенсивність навантаження індивідуально для кожної людини.

РОЗДІЛ 5
СТАРТАП-ПРОЕКТ

Програмна система дозволяє проводити лінійний регресійний аналіз параметрів артеріального тиску до та після навантаження, що дозволяє моніторити стан студентів на заняттях фізичного виховання.

Складові:

1) підсистема зчитування даних;

2) підсистема обробки даних;

3) підсистема розрахунку алгоритму;

4) підсистема виведення результату.

Наявна можливість гнучкої роботи з базою даних завдки наявності функціоналу фільтрації.

Бізнес-модель: орієнтація на дослідні інститути.

Цінний продукт.

1. Сукупність товарів-послуг (покращення товару-послуг)

Комплекс послуг: оцінка взаємозв’язку показників тиску в референтній точці(ліве плече) та інших точках нижніх та верхніх кінцівок.

2. Вирішує такі проблеми клініки:

оцінка стану студентів на заняттях фізичного виховання;

3. Підтримка і сервіс програмної системи:

Додатковий сервіс надається шляхом підтримки системи розробником.

4. Продуктова лінійка (унікальність):

Гнучкий функціонал фільтрації вхідних даних.

Унікальність пропозиції (новизна, продуктивність, дизайн, ціна, економія на витратах, зниження ризику, доступність, зручність)

На ринку не існує аналогів у даної програмної системи.

Сегмент споживачів.

1. Ринок

Ринок: середнє сегментування – надання послуг дослідним інститутам.

В майбутньому планується розширення списку сегментів (міжнародний ринок).

Канали збуту.

1. Прямі – прямий продаж через сайт.

Фріміум – частина послуг (наприклад, інформаційна) безкоштовна, частина – платна (або Shareware - 1 місяць безкоштовне надання послуг, далі – платно).

2. Функції.

Продажні (велике рішення) – в системі присутні всі модулі.

Надання рекомендацій в реальному часі – модуль вартістю 1000 грн.

Взаємодія із споживачами

Залучення закладів, їх утримання.

Заклад купує систему.

Підтримка (пошта, телефон, особисто, форум)

Наприклад, на форумі можна повідомляти про внесення в систему вдосконалень (при цьому необхідні повторні продажі)

Залучення клієнтів завдяки науковим та медичним конференціям.

Повторний продаж системи завдяки реалізації додаткових модулів-послуг.

Дохід (монетизація).

За що і скільки готовий платити клієнт?

За систему, яка буде працювати та підтримуватися розробником.

Ключові види діяльності.

1. Процес створення цінності

Адміністрування, розробка системи (в тому числі і наукова діяльність), навчання користувачів системи, створення додаткових модулів (за потребою)

2. Виробництво товару-послуг, продаж.

3. Підтримка рішення

Ключові ресурси.

1. Матеріальні (в т.ч. комп’ютерна техніка) - 3 персональні комп’ютери, принтер.

2. Інтелектуальні (в т.ч. патенти та ліцензії).

3. Людські:

- Бухгалтер веде всю фінансову діяльність фірми (нарахування і сплата податків, розподіл прибутку, розрахунок і видача зарплати).

- Старший програміст здійснює розробку програм, програмно-технічних засобів і контролює їх якісне виконання. Під його контролем працюють два фахівці в даній області, які й реалізують успішне виконання проекту.

- Маркетолог – за дослідження потреб, доцільності подальшого розширення системи,

- Генеральний директор займається кадрами, укладає договори на поставку продукції в організації та установи, відвідує виставки, конференції з обміну досвідом, відповідає за поставку обладнання у випадку його зносу, технічного старіння. Забезпечує регулярну поставку сировини, проводить дослідження ринку, виконує розрахунки, пов'язані зі змінами в технології.

4. Фінансові - фінансування всіх членів команди (в т.ч., програмістів-розробників)

5. Час реалізації проекту

Ключові партнери.

1. Забезпечення ресурсами (комп’ютерною технікою та обладнанням).

2. Оптимізація та економія в продажах.

3. Зниження ризиків і невизначеності.

4. Подальший розвиток проекту.

Витрати.

Таблиця 5.1

Витрати на паливо й енергію на технологічні цілі

Назва	Кількість	Потужність обладнання, кВт	Корисний фонд часу, годин	Коефіцієнт за часом	Коефіцієнт за потужністю	Тарифи за одну кВт/год. енергії, грн.
Комп'ютер	4	0,40	0,01	0,50	0,50	140,70
Всього	4	0,40	0,01	0,50	0,50	140,70

Таблиця 5.2

Розрахунок собівартості одиниці продукції

Найменування статей калькуляції	Всього, грн.	Питома вага, %
Попутні комплектуючі вироби і напівфабрикати	166,4500	76,4470
Паливо й енергія на технологічні цілі	1,9698	0,9047
Основна заробітна плата	11,0000	5,0521
Додаткова заробітна плата	3,3000	1,5156
Відрахування на соціальне страхування	5,3768	2,4695
Відшкодування зносу спеціальних інструментів	0,0019	0,0009

Продовж. табл. 5.2

Найменування статей калькуляції	Всього, грн.	Питома вага, %
Витрати на утримання і експлуатацію обладнання	0,1670	0,0767
Загальновиробничі витрати	12,2837	5,6416
Виробнича собівартість	200,5492	92,1080
Адміністративні витрати	11,1670	5,1288
Інші витрати	2,0055	0,9211
Витрати на збут	4,0110	1,8422
Повна собівартість	217,7327	100,0000

Планується продаж 2000 ліцензій в місяць, вартість яких становить 435465,3726 грн.

Фінансовий план

Витрати

Таблиця 5.3

Витрати на місяць

Статті витрат	Сума, грн
1. Постійні витрати	62370,21
1.1 Орендна плата	7680,00
1.2 Заробітна плата працівникам	33000,00
1.3 Нарахування на заробітну плату	17028,00
1.4 Плата за телефон та інтернет	119,81
1.7 Витрати на електроенергію	3939,60
1.8 Амортизаційні відрахування	602,80
2. Змінні витрати (на перший місяць)	396834,35
2.1. Витрати на закупку матеріалів та техніки	391834,35

Продовж. табл. 5.3

Статті витрат	Сума, грн
2.2. Витрати на рекламу	4000,00
2.3. Інші невраховані витрати	1000,00
Вартість устаткування	40000,00
Всього	499204,56

Постійні витрати:

- орендна плата - 7680 грн/міс;
- заробітна плата працівникам;
- генеральний директор – 5000 грн;
- бухгалтер – 4000 грн;
- старший програміст – 4000 грн;
- старший інженер – 4000 грн;
- програміст – 3500 грн;
- прибиральниця – 2000 грн;

Всього: 33000 грн.

Відрахування з заробітної плати:

- до пенсійного фонду – 35,2%;
- єдиний соціальний внесок – 16,4%;

Всього: 51,6% (51,6% від 33000,00грн = 17 028 грн).

Амортизаційні відрахування:

- зношення обладнання становить 10% від балансової вартості обладнання на рік – 4000 грн., а в місяць – 334 грн;
- зношення споруди 3.5% від балансової вартості на рік – 3225,6 грн, а в місяць - 268, 8 грн;

Визначення місячної виручки

Пристрій ми плануємо продавати в офісі за ціною 300 грн за 1 шт (300, 00 • 2000 = 600000 грн).

Розрахунок точки беззбитковості

Точка беззбитковості дозволить визначити, коли проект перестане бути збитковим (рис. 5.1).

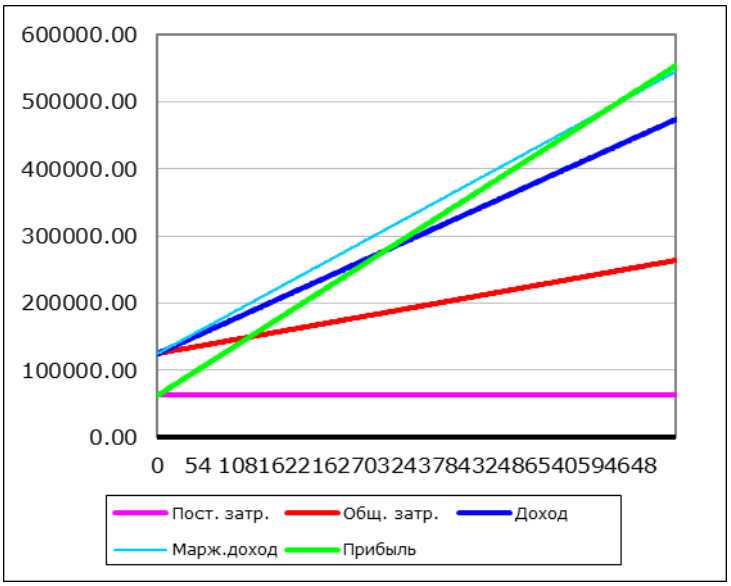

Рисунок 5.1. Точка беззбитковості.

Вийшло, що величина точки беззбитковості дорівнює 613,98 штук, тобто необхідно випустити 614 ліцензій, і тільки після цього підприємство стане отримувати прибуток.

Окупність.

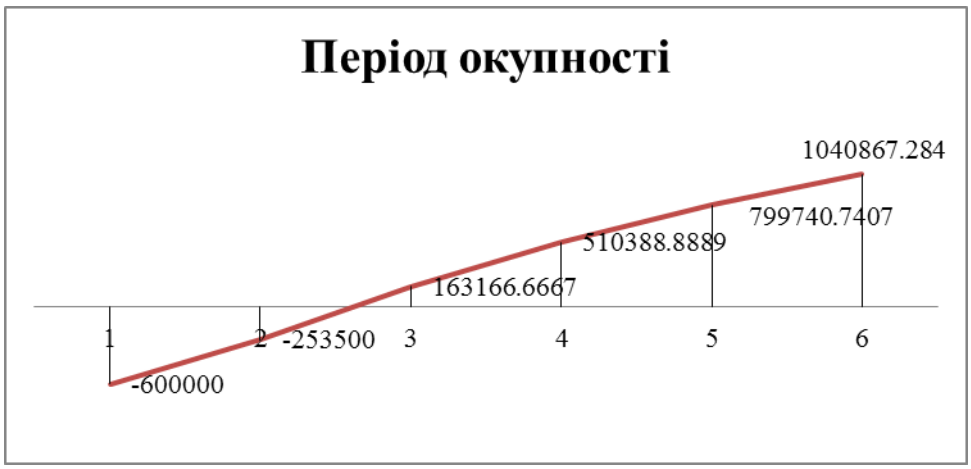

Рисунок 5.2. Період окупності.

Дисконтований період окупності 2,6084 місяці.

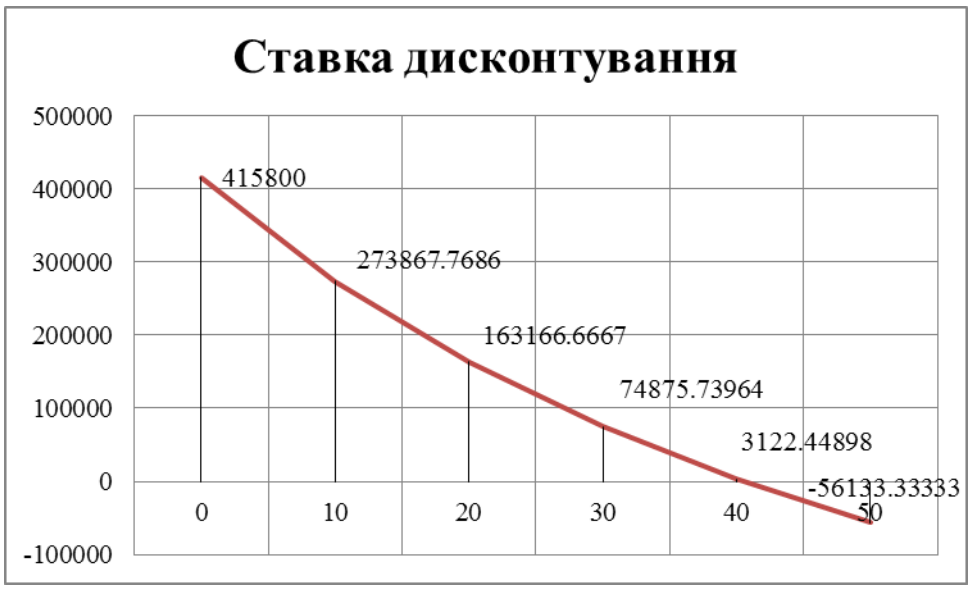

Рисунок 5.3. Ставка дисконтування.

IRR ≈ 41 % (ставка рентабельності за якої чиста приведена вартість 0).

Споживчі властивості товару:

1. Базові (очікувані) властивості

- придбавши систему клініка очікує на підвищення відсотку вірно встановлених діагнозів.

2. Основні (бажані) властивості

- зручність під час використання
- точність результату
- надійність у роботі

Дослідження ринку

Під час початкового періоду дослідження ринку потрібно відвідувати медичні конференції, на яких розглядаються дані проблеми.

Дослідження конкурентного оточення

Повний доступ до всіх ресурсів можна здійснити одразу після встановлення додатку на комп’ютер (тобто процедура реєстрації не потрібна). Користувачам видається повний доступ до інформації зі змогою редагування даних.

Конкурентів у програмної системи немає.

План розвитку товару

- надійність товару в споживанні: зведення помилок програми до мінімуму;
- ергономічні властивості: зручність експлуатації товару - зручний інтерфейс системи;
- естетичні властивості: здатність товару виражати свою соціокультурну значимість, ступінь корисності та досконалість;
- безпека споживання: система є безпечною при використанні;
- як зміна характеристик продукту змінює споживчі властивості: додавання модулів до системи розширює споживчі властивості. Збільшення функцій (модулів) збільшує ціну товару.

Додаток використовують в комплексі із комп'ютером.

Управління ціною:

1. Формування ціни на продукт (витрати на розробку системи, організаційні заходи, рекламні заходи).

2. Формування системи лояльності: системи знижок, заохочень (для збільшення продажів, повторних продажів, партнерських продажів): робота із БД клінік для покращення алгоритмів.

Висновки до розділу 5

Створено стартап-проект на основі магістерської дисертації. Розраховано його фінансовий плвн, точку беззбитковості. Створено план розвитку товару. Розраховано собівартість одиниці товару.

ВИСНОВКИ

Програмна система надає користувачу можливість формувати вибірку вхідних даних для аналізу. Даний функціонал реалізований за допомогою набору з чотирьох фільтрів для роботи з базою даних Excel.

Поле для текстового пошуку дозволяє проводити пошук за ім'ям або за номером кластеру. Фільтр ділянки тіла дозволяє задати одну з семи точок, в якій проводилося вимірювання тиску. Фільтр типу активності дозволяє обирати показники, отримані у стані спокою або після фізичного навантаження. Фільтр типу тиску дозволяє проводити аналіз для систолічного або діастолічного тиску. Таким чином, наприклад, можна провести регресійний аналіз параметрів систолічного тиску для певної особи у стані спокою, з метою встановлення взаємозв'язку між показниками тиску на лівому плечі та правому стегні.

Результати розрахунків записуються в базу даних, з якою працював користувач на новому листі. Лист містить інформацію щодо застосованих фільтрів, що надає користувачу зручну навігацію між розрахунками при множинному використанні програмної системи.

Спроектований програмний продукт був реалізований за допомогою мови програмування java в середовищі IntelliJ IDEA 2017 community edition. Він являє собою зручну програму для регресійного аналізу параметрів артеріального тиску.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кушаковский М.С. Гипертоническая болезнь, 4-е издание [Текст] / М.С. Кушаковский. - СПб.: Сотис, 1995. - 311 с.

2. Janeway T. Important contributions to clinical medicine during the past 30 years from the study of human blood pressure [Text] / T. Janeway. - New York: Bull Johns Hopkins Hosp., 1915. - 342 p.

3. Pescatello L., Franklin B. American College of Sports Medicine position stand. Exercise and hypertension [Text] / L. Pescatello, B. Franklin. - New York: Med Sci Sports Exerc., 2004. - 543 p.

4. Демиденко Е.З. Линейная и нелинейная регрессия [Текст] / Е.З. Демиденко. - Москва: Финансы и статистика, 1981. - 302 с.

5. Линник Ю. В. Метод наименьших квадратов и основы математико-статистической теории обработки наблюдений, 2-е издание [Текст] / Ю.В. Линник. - Москва: Слово, 1962. - 437 с.

6. J. Wolberg Data Analysis Using the Method of Least Squares: Extracting the Most Information from Experiments [Text] / J. Wolberg. - Berlin: Springer, 1982. - 475 p.

7. Björck, Åke Numerical methods for least squares problems [Text] / Björck, Åke. – Philadelphia: Union, 2001. - 637 p.

8. Fox J. Applied Regression Analysis, Linear Models, and Related Methods [Text] / J. Fox. - Thousand Oaks, CA: SAGE Publications, 1997. - 544 p.

9. Lai T.L. Strong consistency of least squares estimates in multiple regression [Text] / T.L. Lai. - Madrid: Springer, 1978. - 514 p.

10. Айвазян С. А., Мхитарян В. С. Прикладная статистика и основы эконометрики [Текст] / С.А. Айвазян, В.С. Мхитриян. - М.: ЮНИТИ, 1998. – 1022 с.

11. Misuno I.S. Experimental investigation of handwritten digit classification [Text] / I.S. Misuno. – Madrid: System Technologies, 2005. - 133 p.

12. Галямина И.Г. Управление процессами: Учебник для вузов. Стандарт третьего поколения [Текст] / И.Г. Галямина. - СПб.: Питер, 2013. - 304 с.

13. Репин В.В., Елиферов В.Г. Процессный подход к управлению. Моделирование бизнес-процессов [Текст] / В.В. Репин, В.Г. Елиферов. - М.: Мани, Иванов и Фербер, 2013. - 544 с.

14. Леоненков А.В. Самоучитель UML 2 [Текст] / А.В. Леоненков. - СПб.: БХВ-Петербург, 2007. - 576 с.

15. Колесов Ю.Б. Моделирование систем. Объектно-ориентированный подход. Учебное пособие [Текст] / Ю.Б. Колесов, Ю.Б Сениченков. - СПб.: БХВ-Петербург, 2012. - 192 с.

16. Шмуллер Джозеф. Освой самостоятельно UML за 24 часа, 3-е издание [Текст] / Д. Шмуллер. - М.: Издательский дом "Вильямс", 2005. - 416 с.

17. Дорот В.Л., Новиков Ф.А. Толковый словарь современной компьютерной лексики. - 3-е изд., перераб. и доп [Текст] / В.Л. Дорот, Ф.А. Новиков - СПб.: БХВ-Петербург, 2004. - 608 с.

18. Основні елементи і поняття IDEF0 [Електронний ресурс]. - Режим доступу : URL : http://um.co.ua/3/3-13/3-132841.html

19. Andrew Filev Professional UML Using Visual Studio .Net; Publilkat - Москва, 2012. - 360 с.

20. Герберт Шилдт Java 8. Полное руководство, 9-е издание [Текст] / Гербердт Штлдт. - М.: «Вильямс», 2015. - 1376 с.

21. Кей С. Хорстманн. Java SE 8. Вводный курс [Текст] / Кей С. Хорстман. - М.: «Вильямс», 2014. - 208 с.

22. Фрэд Лонг Руководство для программиста на Java: 75 рекомендаций по написанию надежных и защищённых программ [Текст] / Фрэд Лонг. - М.: «Вильямс», 2014. - 256 с.

23. Кей С. Хорстманн Java. Библиотека профессионала, том 1. Основы. 9-е издание [Текст] / Кей С. Хорстман. - М.: «Вильямс», 2013. - 864 с.

24. Кей С. Хорстманн Java. Библиотека профессионала, том 1. Основы. 9-е издание [Текст] / Кей С. Хорстман. - М.: «Вильямс», 2017. - 864 с.

25. Кей С. Хорстманн Java. Библиотека профессионала, том 2. Основы. 9-е издание [Текст] / Кей С. Хорстман. - М.: «Вильямс», 2017. - 864 с.

26. Барри Берд Java 8 для чайников [Текст] / Барри Берд. - М.: «Диалектика», 2015. - 400 с.

27. Джеймс Гослинг Язык программирования Java SE 8. Подробное описание, 5-е издание [Текст] / Джеймс Гослинг. - М.: «Вильямс», 2015. - 672 с.

28. Джошуа Блох Java. Эффективное программирование [Текст] / Джошуа Блох. - М.: Лори, 2002. - 224 с.

29. Монахов Вадим Язык программирования Java и среда NetBeans [Текст] / Монах Вадим. - СПб.: БХВ-Петербург, 2011. - 704 с.

30. Брюс Эккель Философия Java [Текст] / Брюс Эккель. - СПб.: Питер, 2003. – 976 с.

31. Hastie T. The elements of statistical learning [Text] / T. Hastie, R. Tibshirani, J. Friedman. – New York: Springer, 2014. – 739 p.

32. Bishop C. M. Pattern recognition and machine learning [Text] / C. M . Bishop. – New York: Springer, 2006. – 738 p.

33. Merkov A.B. Image recognition: Introduction to statistical learning methods [Text] / A.B . Merkov. – Moscow: URSS, 2011. – 256 p.

34. Vapnik V. The nature of statistical learning theory [Text] / V. Vapnik. – New York: Springer-Verlag, 1995. – 188 p.

35. Gori M. Machine Learning: A constraint-based approach [Text] / M. Gori. – Waltham: Morgan Kaufmann; 2017. – 580 p.

36. Хайкин С. Нейронные сети: полный курс, 2-е издание [Текст] / С. Хайкин. – Издательский дом Вильямс, 2008. – 1103 с.

37. Харарі Ю.Н. Людина розумна. Історія людства від минулого до майбутнього [Текст] / Ю.Н. Харарі. – Family Leisure Club, 2016. – 413 с.

38. Терехов С.А. Лекції з теорії і додатків штучних нейронних мереж. Лабораторія штучних нейронних мереж НТО – 2 [Текст] / С.А. Терехов. – Снежинск – ВНДІТФ, 2010. – 104 с.

39. Kodratoff Y. Machine learning: an artificial intelligence approach [Text] / Y. Kodratoff, R. S. Michalski., Vol. 3. – Moskow: Elsevier; 2014. – 825 p.

40. Camastra F. Machine learning for audio, image and video analysis: Theory and Applications [Text] / F. Camastra, A. Vinciarelli. – Madrid: Springer; 2015. – 561 p.

Додаток А

Лістинг коду

```
public class LinearRegressionCalculator {
    public Map<String, String> calculate(Map<List<Integer>, List<Integer>> data) {
        // first pass: read in data, compute xbar and ybar
        Double sumX = 0.0;
        Double sumY = 0.0;
        Double sumX2 = 0.0;
        List<Integer> x = new ArrayList<>(data.keySet()).get(0);
        List<Integer> y = new ArrayList<>(data.values()).get(0);
        for (Integer i : x) {
            sumX += i;
            sumX2 += i * i;
        }
        for (Integer i : y) {
            sumY += i;
        }
        Double xBar = sumX / x.size();
        Double yBar = sumY / y.size();
        // second pass: compute summary statistics
        Double xXBar = 0.0, yYBar = 0.0, xYBar = 0.0;
        for (int i = 0; i < x.size(); i++) {
            xXBar += (x.get(i) - xBar) * (x.get(i) - xBar);
            yYBar += (y.get(i) - yBar) * (y.get(i) - yBar);
            xYBar += (x.get(i) - xBar) * (y.get(i) - yBar);
        }
        double beta1 = xYBar / xXBar;
        double beta0 = yBar - beta1 * xBar;
        String sign = beta0 >= 0.0 ? "+" : " ";
        String equation = "y = " + beta1 + " * x " + sign + beta0;
        int df = x.size() - 2;
        double rss = 0.0;      // residual sum of squares
        double ssr = 0.0;      // regression sum of squares
        for (int i = 0; i < x.size(); i++) {
            double fit = beta1 * x.get(i) + beta0;
            rss += (fit - y.get(i)) * (fit - y.get(i));
            ssr += (fit - yBar) * (fit - yBar);
        }
        double R2 = ssr / yYBar;
        double svar = rss / df;
        double svar1 = svar / xXBar;
        double svar0 = svar / x.size() + xBar * xBar * svar1;
        svar0 = svar * sumX2 / (x.size() * xXBar);
        Map<String, String> result = new HashMap<>();
        result.put("Equation", equation);
        result.put("R2", String.valueOf(R2));
        result.put("ErrorA", String.valueOf(Math.sqrt(svar1)));
        result.put("ErrorB", String.valueOf(Math.sqrt(svar0)));
        result.put("SSTO", String.valueOf(yYBar));
```

```
        result.put("SSE", String.valueOf(rss));
        result.put("SSR", String.valueOf(ssr));
        return result;
    }
}
public class Controller {
    private Stage stage;
    private List<ExcelDatabaseDTO> listExcelRows;
    private File file;
    private Map<String, List<ExcelDatabaseDTO>> dataFilteredByName;
    private Map<String, List<ExcelDatabaseDTO>> dataFilteredByCluster;

    @FXML
    private ChoiceBox<String> filterTypeChoiceBox;
    @FXML
    private ChoiceBox<String> bodyPartChoiceBox;
    @FXML
    private ChoiceBox<String> pressureTypeChoiceBox;
    @FXML
    private ChoiceBox<String> activityTypeChoiceBox;
    @FXML
    private TextField searchField;
    @FXML
    private Button calculateButton;

    public void init(Stage stage) {
        this.stage = stage;
    }
    public void loadDatabase() {
        System.out.println("open file");
        FileChooser fileChooser = new FileChooser();
        fileChooser.setTitle("Open file");
        fileChooser.getExtensionFilters().addAll(
                new FileChooser.ExtensionFilter("Excel files", "*.xls", "*.xlsx")
        );
        File file = fileChooser.showOpenDialog(stage);
        ExcelDatabaseToObjectConverter converter = new ExcelDatabaseToObjectConverter();
        if (Objects.nonNull(file)) {
            this.file = file;
            listExcelRows = converter.convertDatabaseToListOfObjects(file);
            dataFilteredByName = converter.convertExcelRowsToFilteredMap(listExcelRows,
FilterType.NAME.getValue());
            dataFilteredByCluster = converter.convertExcelRowsToFilteredMap(listExcelRows,
FilterType.CLUSTER.getValue());
            Alert alert = new Alert(Alert.AlertType.INFORMATION);
            alert.setTitle("Information Dialog");
            alert.setHeaderText(null);
            alert.showAndWait();
            System.out.println("File: " + file);
            searchField.setDisable(false);
        }
    }
    public void enableCalculateButton() {
```

```
        if (Objects.nonNull(searchField.getText()) && !searchField.getText().isEmpty()) {
            calculateButton.setDisable(false);
        }else {
            calculateButton.setDisable(true);
        }
        System.out.println("enableCalculateButton() function");
    }
    public void calculateLinearRegression() throws ClassNotFoundException,
NoSuchMethodException, IllegalAccessException, InvocationTargetException, IOException {
        LinearRegressionCalculator calculator = new LinearRegressionCalculator();
        ExcelWriter writer = new ExcelWriter();
        RegressionAnalysisDataPreparation dataPreparation = new
RegressionAnalysisDataPreparation();
        switch (filterTypeChoiceBox.getSelectionModel().getSelectedItem()) {
            case "Name":
                Map<String, String> resultForName =
calculator.calculate(dataPreparation.getDataByName(dataFilteredByName, bodyPartChoiceBox,
activityTypeChoiceBox, pressureTypeChoiceBox, searchField));
                writer.writeIntoExcel(resultForName, file, searchField, bodyPartChoiceBox,
activityTypeChoiceBox, pressureTypeChoiceBox);
                break;
            case "Cluster":
                Map<String, String> resultForCluster =
calculator.calculate(dataPreparation.getDataByCluster(dataFilteredByCluster,
bodyPartChoiceBox, activityTypeChoiceBox, pressureTypeChoiceBox, searchField));
                writer.writeIntoExcel(resultForCluster, file, searchField, bodyPartChoiceBox,
activityTypeChoiceBox, pressureTypeChoiceBox);
                break;
        }
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Information Dialog");
        alert.setHeaderText(null);
        alert.setContentText("Calculation is finished. \n Please see the results in selected
file.");
        alert.showAndWait();
        System.out.println("calculate button clicked");
    }
}
public class ExcelWriter {
    public void writeIntoExcel(Map<String, String> result, File file, TextField searchField,
ChoiceBox<String> bodyPartChoiceBox, ChoiceBox<String> activityTypeChoiceBox,
ChoiceBox<String> pressureTypeChoiceBox) throws IOException {
        FileInputStream xlsFile = new FileInputStream(file);
        XSSFWorkbook workbook = new XSSFWorkbook(xlsFile);
        XSSFSheet sheet = workbook.createSheet(searchField.getText() + " " +
bodyPartChoiceBox.getValue() + " " + pressureTypeChoiceBox.getValue() + " " +
activityTypeChoiceBox.getValue());
        Object[][] datatypes = {
                {"Searched value", "Body part", "Pressure type", "Activity type"},
                {searchField.getText(), bodyPartChoiceBox.getValue(),
pressureTypeChoiceBox.getValue(), activityTypeChoiceBox.getValue()},
                {" ", " ", " ", " ", " ", " ", " "},
                {"Equation", "R^2", "Std. error of a", "Std. error of b"},
```

```
                {result.get("Equation"), result.get("R2"), result.get("ErrorA"),
result.get("ErrorB")},
                {" ", " ", " ", " ", " ", " ", " "},
                {" ", "SSTO(total sum of squares)", "SSE(error sum of squares)",
"SSR(regression sum of squares)"},
                {" ", result.get("SSTO"), result.get("SSE"), result.get("SSR")}
        };
        XSSFCellStyle cellStyle;
        cellStyle = workbook.createCellStyle();
        XSSFFont xSSFFont = workbook.createFont();
        xSSFFont.setFontName(XSSFFont.DEFAULT_FONT_NAME);
        xSSFFont.setFontHeightInPoints((short) 14);
        cellStyle.setFont(xSSFFont);
        int rowNum = 0;
        System.out.println("Creating excel");
        for (Object[] datatype : datatypes) {
            Row row = sheet.createRow(rowNum++);
            int colNum = 0;
            for (Object field : datatype) {
                sheet.setColumnWidth(colNum, 60 * 256);
                Cell cell = row.createCell(colNum++);
                cell.setCellStyle(cellStyle);
                if (field instanceof String) {
                    cell.setCellValue((String) field);
                } else if (field instanceof Integer) {
                    cell.setCellValue((Integer) field);
                }
            }
        }
        try {
            FileOutputStream outputStream = new FileOutputStream(file);
            workbook.write(outputStream);
            workbook.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.println("Done");
    }
}
public class RegressionAnalysisDataPreparation {
    public Map<List<Integer>, List<Integer>> getDataByName(Map<String,
List<ExcelDatabaseDTO>> dataFilteredByName, ChoiceBox<String> bodyPartChoiceBox,
ChoiceBox<String> activityTypeChoiceBox, ChoiceBox<String> pressureTypeChoiceBox, TextField
searchField) throws ClassNotFoundException, NoSuchMethodException, InvocationTargetException,
IllegalAccessException {

        Map<List<Integer>, List<Integer>> result = new HashMap<>();
        UIValuesToDTOValuesConverter converter = new UIValuesToDTOValuesConverter();
        String bodyPart = bodyPartChoiceBox.getSelectionModel().getSelectedItem();
        String activityType = activityTypeChoiceBox.getSelectionModel().getSelectedItem();
        String presureType = pressureTypeChoiceBox.getSelectionModel().getSelectedItem();
```

```
        String name = searchField.getCharacters().toString();
        String filter = converter.convertToFilterTitle(bodyPart, activityType, presureType);
        String filterReferent = converter.convertToFilterTitle("LeftShoulder", activityType,
presureType);
        List<Integer> referentPressure = getPressureValues(name, filterReferent,
dataFilteredByName);
        List<Integer> selectedPressure = getPressureValues(name, filter, dataFilteredByName);
        result.put(referentPressure, selectedPressure);
        return result;
    }
    public Map<List<Integer>,List<Integer>> getDataByCluster(Map<String,
List<ExcelDatabaseDTO>> dataFilteredByCluster, ChoiceBox<String> bodyPartChoiceBox,
ChoiceBox<String> activityTypeChoiceBox, ChoiceBox<String> pressureTypeChoiceBox, TextField
searchField) throws ClassNotFoundException, NoSuchMethodException, IllegalAccessException,
InvocationTargetException {
       Map<List<Integer>, List<Integer>> result = new HashMap<>();

        UIValuesToDTOValuesConverter converter = new UIValuesToDTOValuesConverter();
        String bodyPart = bodyPartChoiceBox.getSelectionModel().getSelectedItem();
        String activityType = activityTypeChoiceBox.getSelectionModel().getSelectedItem();
        String presureType = pressureTypeChoiceBox.getSelectionModel().getSelectedItem();
        String cluster = searchField.getCharacters().toString();
        String filter = converter.convertToFilterTitle(bodyPart, activityType, presureType);
        String filterReferent = converter.convertToFilterTitle("LeftShoulder", activityType,
presureType);
        List<Integer> referentPressure = getPressureValues(cluster, filterReferent,
dataFilteredByCluster);
        List<Integer> selectedPressure = getPressureValues(cluster, filter,
dataFilteredByCluster);
        result.put(referentPressure, selectedPressure);
        return result;
    }
    private List<Integer> getPressureValues(String filterType, String methodName, Map<String,
List<ExcelDatabaseDTO>> dataFilteredByType) throws ClassNotFoundException,
NoSuchMethodException, InvocationTargetException, IllegalAccessException {
        List<ExcelDatabaseDTO> filteredData = dataFilteredByType.get(filterType);
        Class<?> c = Class.forName("sample.dto.ExcelDatabaseDTO");
        Method method = c.getDeclaredMethod("get" + methodName);
        List<Integer> result = new ArrayList<>();
        for (ExcelDatabaseDTO dto : filteredData) {
            String pressureValue = (String) method.invoke(dto);
            result.add(Integer.valueOf(pressureValue));
        }
        return result;
    }
}
```