МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

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

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

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «магістр»

на тему: Нелінійна регресійна модель для оцінювання тривалості проєктів зі створення програмного забезпечення за методологією Agile та розробка програми для її реалізації

Виконав: студент 6151мз групи
Каіров В.О.
(підпис, ПІБ)

Керівник роботи:
доцент, к.т.н., доцент
(посада, науковий ступень вчене звання)
Макарова Л.М.
(підпис, ПІБ)

Миколаїв – 2021 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

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

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

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

Студенту Каірову Володимиру Олексійовичу _____
(Прізвище, ім'я, по батькові)

1. Тема роботи: Нелінійна регресійна модель для оцінювання тривалості проєктів зі створення програмного забезпечення за методологією Agile та розробка програми для її реалізації

Керівник роботи Макарова Лідія Миколаївна, к.т.н., доцент _____
Затверджені наказом ректора № 1239уч від «13» _____10_____ 2021 р.
2. Термін подання роботи: 06.12.2021 р._____
3. Вихідні дані по роботі: _____
4. Перелік питань, що належать до розробки (найменування розділів) _____
Вступ (Актуальність теми. Зв'язок роботи з науковими програмами, планами, темами. Мета і завдання дослідження. Об'єкт дослідження. Предмет дослідження. Методи дослідження. Наукова новизна одержаних результатів. Практичне значення одержаних результатів. Особистий внесок здобувача. Апробація результатів досліджень**.** Публікації.); Огляд літератури та обґрунтування необхідності проведення досліджень за обраною темою; Викладення результатів власних досліджень з висвітленням того нового, що пропонується; Проєкт програмного забезпечення; Організаційно-економічний розділ; Розділи з охорони праці та охорони навколишнього середовища; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма і методика випробувань програмного забезпечення)

5. Перелік презентаційних матеріалів: Актуальність теми, Об'єкт і предмет дослідження, Методи дослідження, Наукова новизна одержаних результатів, Практичне значення одержаних результатів, Апробація результатів досліджень та публікації, Нелінійна регресійна модель, Лінійне рівняння регресії, довірчий інтервал та інтервал прогнозування, Нелінійне рівняння регресії, довірчий інтервал та інтервал прогнозування, Метрики якості, Діаграма варіантів використання, Діаграма діяльності, Концептуальна модель даних, Діаграма послідовності для варіанту використання «Визначення метрик якості», Діаграма компонентів ПЗ, Інтерфейс користувач, Висновки
_____

6. Консультанти розділів роботи

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

7. Дата видачі завдання 13.10.2021 р. _____

К А Л Е Н Д А Р Н И Й П Л А Н

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

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

Студент _____ Каіров В.О.
(підпис) (ПІБ)

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

РЕФЕРАТ

Каіров Володимир Олексійович

Нелінійна регресійна модель для оцінювання тривалості проєктів зі створення програмного забезпечення за методологією Agile та розробка програми для її реалізації.

Кваліфікаційна робота на здобуття освітнього рівня магістра зі спеціальності 121 – «Інженерія програмного забезпечення». Національний університет кораблебудування імені адмірала Макарова. Миколаїв, 2021 р.

Обсяг роботи: 104 стор., 9 табл., 11 рис., 27 використаних джерел, 5 додатків.

Актуальність теми. При розробці програмних проєктів виконавці вирішують і задачі оцінювання. Тематика оцінювання тривалості та інших ресурсів розробки ПЗ в повній мірі ще не розкрита. При вирішенні задач оцінювання тривалості нових проєктів сама проєктна діяльність обумовлює необхідність використання методів аналізу емпіричних даних. Щоб поліпшити точність алгоритмічних моделей, необхідно пристосувати їх до конкретних обставин. Проте не було знайдено моделей для оцінювання тривалості розробки ПЗ, створеного за методологією Agile. Тому розробка такої моделі є актуальною задачею.

Мета і завдання дослідження. Метою роботи є підвищення достовірності оцінювання тривалості розробки програмного забезпечення, яке створюється за методологією Agile. Для досягнення поставленої мети необхідно вирішити наступні завдання: проаналізувати існуючі моделі оцінювання тривалості розробки ПЗ; обґрунтувати необхідність проведення досліджень; побудувати нелінійну регресійну модель оцінювання тривалості розробки ПЗ, довірчий інтервал та інтервал передбачення; розробити програму для реалізації побудованої моделі.

Об’єктом дослідження є процес оцінювання тривалості розробки програмного забезпечення, створеного за методологією Agile.

Предметом дослідження є регресійна модель оцінювання тривалості розробки програмного забезпечення, створеного за методологією Agile.

Методи дослідження: при проведенні досліджень використовувались методи теорії ймовірностей, математичної статистики, регресійного аналізу.

Наукова новизна одержаних результатів: удосконалено регресійну модель для оцінювання тривалості розробки програмного забезпечення, яке створюється за методологією Agile, з використанням нормалізуючого перетворення на основі десяткового логарифму, що дозволяє підвищити достовірність оцінювання тривалості розробки ПЗ в порівнянні з лінійною моделлю.

Практичне значення отриманих результатів. ПЗ для оцінювання тривалості розробки програмного забезпечення, створеного за методологією Agile, яке розроблене в рамках магістерської роботи, дозволило автоматизувати та скоротити час відповідних розрахунків, а також підвищити достовірність оцінювання.

Апробація результатів роботи. Результати досліджень, викладених у роботі, були оприлюднені на XIII міжнародній науково-практичній конференції «FREE AND OPEN SOURCE SOFTWARE» (м. Харків, 16-18.11.2021).

Ключові слова: ОЦІНЮВАННЯ ТРИВАЛОСТІ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, МЕТОДОЛОГІЯ AGILE, НЕЛІНІЙНА РЕГРЕСІЙНА МОДЕЛЬ, НОРМАЛІЗУЮЧЕ ПЕРЕТВОРЕННЯ

ABSTRACT

Kairov Volodymyr

A nonlinear regression model for estimating the duration of software development projects using Agile metodology and developing the software for its implementation

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

The work contains: 104 pages, 9 tables, 11 figures, 27 references, 5 appendices.

Relevance of the topic: When developing software projects, performers also solve evaluation problems. The topic of estimating the duration and other resources of software development is not yet fully disclosed. When solving problems of estimating the duration of new projects, the project activity itself necessitates the use of methods of empirical data analysis. To improve the accuracy of algorithmic models, it is necessary to adapt them to specific circumstances (language, environment, development technology). However, no models have been found to estimate the duration of Agile software development. Therefore, the development of such a model is an relevant task.

Purpose and tasks of the research. The aim of the work is to increase the reliability of estimating the duration of software development, which is created according to the Agile methodology. To achieve this goal it is necessary to solve the following tasks: to analyze existing models for estimating the duration of software development; justify the need for research; build a nonlinear regression model for estimating the duration of software development, confidence interval and prediction interval; develop a program to implement the built model.

The object of the research is the process of estimating the duration of software development created according to the Agile methodology.

The subject of the research is a regression model for estimating the duration of software development created using the Agile methodology.

Research methods: methods of probability theory, mathematical statistics, regression analysis were used in the research.

Scientific novelty of the obtained results: improved regression model for estimating the duration of software development, which is created by the Agile methodology, using a normalizing transformation based on the decimal logarithm, which increases the reliability of estimating the duration of software development compared to the linear model.

The practical value of the results. Software for estimating the duration of software development, created according to the Agile methodology, which was developed as part of the master's thesis, allowed to automate and reduce the time of relevant calculations, as well as increase the reliability of estimation.

Approbation of research results. The results of the research presented in the paper were published at the XIII International Scientific and Practical Conference "FREE AND OPEN SOURCE SOFTWARE" (Kharkov, 16-18.11.2021).

Keywords: ESTIMATING THE DURATION OF SOFTWARE DEVELOPMENT, AGILE METHODOLOGY, NONLINEAR REGRESSION MODEL, NORMALIZING TRANSFORMATION

ЗМІСТ

ВСТУП

Актуальність теми. При оцінюванні інвестиційної привабливості ІТ-проєкту найбільш важливим і одночасно найбільш складним для розрахунку показником є тривалість його розробки. Це пов'язано з тим, що вона використовується при розрахунку багатьох показників проєкту в якості основної змінної. Однак на етапі прийняття рішення про початок інвестування відсутня велика кількість інформації про характеристики проєкту, на основі яких зазвичай розраховують тривалість розробки [1].

Недооцінка тривалості, вартості та інших ресурсів, необхідних для створення програмного проєкту, призводить до недостатньої чисельності проєктної команди, надмірно стислих термінів розробки і, як результат, до втрати довіри до розробників у разі порушення графіка. З іншого боку, перестраховка і переоцінка, тобто якщо для проєкту виділено більше ресурсів, ніж реально необхідно, причому без належного контролю за їх використанням, призводить до збільшення тривалості і вартості розробки та відмови замовників від контракту на цих умовах [2].

Всі моделі оцінювання тривалості та інших ресурсів розробки програмного забезпечення можна розділити на неалгоритмічні та алгоритмічні [3]. До неалгоритмічних належать: оцінювання по аналогії, експертне оцінювання, принцип Паркінсона та інші. Відомо, що неалгоритмічні моделі мають істотні недоліки. Це робить обгрунтованим вибір алгоритмічних моделей [4-6].

В алгоритмічних моделях математичний апарат варіюється від простих лінійних формул до складних регресійних моделей і диференціальних рівнянь. Зазвичай, щоб поліпшити точність алгоритмічних моделей, необхідно пристосувати їх до конкретних обставин (мови, середовища, технології розробки). Цього можна досягти, наприклад, за допомогою розрахунку основних параметрів і коефіцієнтів. Таким чином, тематика оцінювання

тривалості та інших ресурсів розробки програмного забезпечення в повній мірі ще не розкрита [7-10].

Тому задача побудови нелінійної регресійної моделі для оцінювання тривалості розробки програмного забезпечення за методологією Agile являється актуальною.

Мета і завдання дослідження. Метою роботи є підвищення достовірності оцінювання тривалості розробки програмного забезпечення, яке створюється за методологією Agile.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

– проаналізувати існуючі моделі оцінювання тривалості розробки програмного забезпечення, порівняти їх;

– обґрунтувати необхідність розробки нелінійної регресійної моделі для оцінювання тривалості розробки програмного забезпечення за методологією Agile;

– обрати в якості нормалізуючого перетворення для емпіричних даних, отриманих із попередньо завершених проєктів, десятковий логарифм;

– нормалізувати отримані емпіричні дані;

– перевірити нормалізовані дані на викиди;

– побудувати лінійну регресійну модель, довірчий інтервал та інтервал прогнозування для нормалізованих даних;

– перейти від лінійної регресійної моделі до нелінійної та побудувати нелінійну регресійну модель, довірчий інтервал та інтервал прогнозування для вихідних даних;

– побудувати лінійну регресійну модель та довірчий інтервал і інтервал передбачення для неї без нормалізації даних в припущенні про нормальність вихідних даних;

– виконати порівняння лінійної моделі з нелінійною;

– виконати оцінювання тривалості розробки нового проєкту;

– розробити ПЗ для оцінювання тривалості розробки програмного забезпечення, створеного за методологією Agile, за розробленою методикою.

Об’єктом дослідження є процес оцінювання тривалості розробки програмного забезпечення, створеного за методологією Agile.

Предметом дослідження є регресійна модель оцінювання тривалості розробки програмного забезпечення, створеного за методологією Agile.

Методи дослідження: при проведенні досліджень використовувались методи теорії ймовірностей, математичної статистики та регресійного аналізу.

Наукова новизна одержаних результатів: удосконалено регресійну модель для оцінювання тривалості розробки програмного забезпечення, яке створюється за методологією Agile, з використанням нормалізуючого перетворення на основі десяткового логарифму, що дозволяє підвищити достовірність оцінювання тривалості розробки програмного забезпечення в порівнянні з лінійною моделлю.

Практичне значення одержаних результатів: ПЗ для оцінювання тривалості розробки програмного забезпечення, створеного за методологією Agile, яке розроблене в рамках магістерської роботи, дозволило автоматизувати та скоротити час відповідних розрахунків, а також підвищити достовірність оцінювання.

Особистий внесок здобувача. Основні положення кваліфікаційної роботи, що представлені до захисту, отримані здобувачем самостійно. У праці, яка опублікована у співавторстві [11], здобувачу належить розробка нелінійної моделі для оцінювання тривалості розробки програмного забезпечення, створеного за методологією Agile.

Апробація результатів роботи. Результати досліджень, викладених у роботі, були оприлюднені на XIII міжнародній науково-практичній конференції «FREE AND OPEN SOURCE SOFTWARE» (м. Харків, 16-18.11.2021).

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

1 AGILE-МЕТОДОЛОГІЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Цінності та принципи Agile Software Development

Agile – це здатність реагувати на зміни. Це спосіб впоратися з невизначеністю та змінами і в кінцевому підсумку досягти успіху.

В 2001 з'явився маніфест для Agile Software Development. Автори Agile-маніфесту обрали «Agile» як символ, через те, що цей термін означає пристосовуваність та реагування на зміни, що були настільки важливими для їхнього підходу.

Що таке Agile Software Development?

Гнучка розробка ПЗ – це більше, ніж Scrum, Extreme Programming або Feature-Driven Development (FDD).

Гнучка розробка ПЗ – це більше, ніж парне програмування, розробка на основі тестів, стендапи, сеанси планування та спринти.

Гнучка розробка ПЗ – це набір методологій і практик, основаних на цінностях і принципах Маніфесту для Agile розробки програмного забезпечення.

Розглянемо цінності гнучкої розробки по Agile-маніфесту [12]:

1) Люди та взаємодія важливіші за процеси та інструменти.

2) Працюючий продукт важливіший за вичерпну документацію.

3) Співпраця із замовником важливіша за погодження умов контракту.

4) Готовність до змін важливіша за слідування початковому плану.

В Agile-маніфесті визначили наступні принципи гнучкої розробки [12]:

1) Найвищий пріоритет – задовольняти клієнта шляхом своєчасної та постійної поставки ПЗ.

2) Дозволяти змінювання вимог навіть на пізніх етапах розробки. Гнучкі процеси використовують зміни для конкурентної переваги клієнта.

3) Постачати версії програмного забезпечення часто, з інтервалом від кількох тижнів до кількох місяців, причому перевагу віддають коротшим термінам.

4) Співпрацювати замовникам та розробникам щодня протягом усього проєкту.

5) Створювати проєкти навколо мотивованих людей. Давати їм необхідне середовище та підтримку та довіряти їм виконання роботи.

6) Приймати спілкування віч-на-віч в якості найефективнішого методу передачі інформації команді розробників та всередині неї.

7) Вважати працююче програмне забезпечення основним показником прогресу.

8) Підтримувати постійний темп спонсорами, розробниками та користувачами. Гнучкі процеси сприяють сталому розвитку.

9) Приділяти постійну увагу технічній досконалості та гарному дизайну як засобам покращення гнучкості.

10) Робити робочий процес простим та зрозумілим.

11) Дозволяти учасникам проєкту приймати рішення.

12) Постійно працювати командою над підвищенням ефективності роботи.

Таким чином, розробка Agile ведеться швидкими циклами, більше схожими на спринти. Результатом є невеликі версії програмного продукту, у кожному з яких додаються нові функції, порівняно з попереднім продуктом. В ідеалі команди виконують велику роботу за менший час. Цей метод виявився найбільш підходящим для продуктів із критичними вимогами до термінів, коли клієнт доступний та готовий спілкуватися протягом усього життєвого циклу розробки. Для Agile потрібна команда, що легко пристосовується, готова швидко реагувати і змінювати продукт на основі результатів тестування та відгуків.

1.2 Плюси та мінуси гнучких методологій

Переваги гнучких методологій [13]:

1) Покращена якість

Розділяючи проєкт на керовані елементи, команда може зосередити свою увагу на якості процесів розробки, тестування та спільної праці.

Якість ПЗ буде кращою, якщо на кожній стадії проводити збирання та тестування. За допомогою таких перевірок можна швидко знайти і відразу виправити помилки та невідповідності на ранніх етапах.

2) Спільна робота з зацікавленими особами

Гнучка методологія надає широке коло можливостей для спільної роботи з зацікавленими особами та командою до, у процесі та після закінчення кожного спринту.

Коли замовник особисто проходить кожен етап створення проєкту, є більша ймовірність того, що між командою розробників та ним виникне високий рівень співробітництва; це, у свою чергу, допоможе групі більш зрозуміти та побачити кінцевий результат.

Коли робоче ПЗ надається у визначені терміни, замовники виявляють до команди більше довіри та більшу активність у проєкті.

3) Орієнтація на замовників

Орієнтація на замовників дозволяє виконувати бета-тести ПЗ після закінчення кожного спринту, отримуючи при цьому важливі відгуки на старті проєкту та надаючи можливість редагування вимог.

4) Прозорість

За методикою Agile, замовники можуть брати участь у проєкті: розставляти пріоритети, визначати набір функцій, складати план ітерації та огляди, виконувати збірки програмного забезпечення, в яких містяться нові функції.

Але, при цьому, замовник повинен розуміти, що йому демонструється незавершений проєкт в обмін на можливість брати активну участь у його плануванні та розробці.

5) Орієнтація на важливість для розвитку бізнесу

Дозволяючи замовнику брати участь у визначенні пріоритетних цілей, розробники розуміють, що їхня важливість для замовника висока, оскільки вони здатні надати корисний набір функцій.

6) Ранній випуск з високим рівнем передбачуваності

Оскільки використовується фіксований розклад спринтів (1-4 тижні), новий функціонал надається швидко, часто та передбачувано. Це також дозволяє виконати бета-тести ПЗ раніше, ніж було заплановано, якщо в цьому є цінність для проєкту.

7) Можливість редагування

Коли команда повністю сконцентрована на впровадженні раніше визначених функцій продукту на всіх ітераціях, незавершені задачі продукту можна постійно вдосконалювати і перерозподіляти.

Нововведені або відредаговані елементи незавершених задач можна запланувати на наступну ітерацію, що дозволить вносити зміни протягом найближчих кількох тижнів.

8) Передбачувані витрати і час реалізації

Оскільки всі спринти фіксовані за тривалістю, то й обсяг витрат передбачуваний та обмежений кількістю робіт, які співробітники можуть виконати за певний часовий проміжок.

За оцінками, що надаються перед кожним спринтом, замовник може приблизно розрахувати середню вартість кожного завдання. Це сприяє поліпшенню прийняття рішень щодо важливості завдань та потреби додаткових ітерацій.

Недоліки методології Agile [13]:

1) Менше передбачуваності

Для деяких програмних продуктів розробники не можуть повною мірою кількісно оцінити необхідні зусилля, особливо на початку життєвого циклу розробки великих продуктів. Команди, які не мають досвіду гнучкої розробки, бояться цих невідомих, що призводить до розчарування, поганих практик і часто до поганих рішень.

2) Більше часу та уваги

Тестувальники, замовники та розробники повинні постійно взаємодіяти один з одним. Потрібні численні бесіди віч-на-віч, оскільки вони є кращою формою спілкування. Щодня замовники повинні бути доступні для оперативного тестування та затвердження на кожному етапі, так щоб розробники змогли відзначити його як виконаний, перш ніж перейти до наступної функції. Завдяки цьому продукт краще відповідає очікуванням користувачів, але потребує більше часу та енергії всіх учасників.

3) Підвищені вимоги до замовників

Принципи Agile вимагають тісної співпраці та активної участі користувача. Замовники повинні пройти навчання, щоб допомогти у розробці продукту. Будь-який недолік участі замовника впливатиме на якість програмного забезпечення та кінцевий успіх.

4) Відсутність необхідної документації

Оскільки вимоги до ПЗ уточнюються якраз під час розробки, документація не надто докладна. Це означає, що якщо до команди приєднаються нові члени, вони не знатимуть подробиць, деяких особливостей та своїх дій. Це створює непорозуміння та труднощі.

5) Проєкт легко збивається зі шляху

Цей метод не вимагає детального планування для початку роботи та передбачає, що потреби замовника постійно змінюються. Такі недостатні вихідні дані можуть обмежити використання Agile. Потім, якщо зворотний зв'язок замовника не реалізується, розробник може сфокусуватися на розробці у неправильному напрямку. Крім того, є небезпека неконтрольованого

розширення меж проєкту, і продукт, що постійно змінюється, стає нескінченним.

Незважаючи на існуючі недоліки методологія Agile не втратила своєї актуальності. Agile дає можливість отримати більший результат за менший час та підвищити ефективність своєї команди.

2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ТА МОДЕЛЕЙ ОЦІНЮВАННЯ ТРИВАЛОСТІ РОЗРОБКИ ПЗ

2.1 Існуючі методи та моделі оцінювання тривалості розробки програмного забезпечення

При оцінюванні інвестиційної привабливості ІТ-проєкту найбільш важливим і одночасно найбільш складним для розрахунку показником є тривалість його розробки. Це пов'язано з тим, що вона використовується при розрахунку багатьох показників проєкту в якості основної змінної. Однак на етапі прийняття рішення про початок інвестування відсутня велика кількість інформації про характеристики проєкту, на основі яких зазвичай розраховують тривалість розробки. Тому використовують різні підходи до оцінювання тривалості розробки [1].

На сьогоднішній день моделі оцінювання тривалості розробки програмного забезпечення можна поділити на дві основні групи: алгоритмічні та неалгоритмічні. Визначимо ж різницю між ними. Алгоритмічні моделі базуються на оцінювані та аналізі кількісних характеристик програми. Неалгоритмічні ж покладаються на використання попередньо визначених схем та принципів або на думку кваліфікованих експертів [3-6].

До неалгоритмічних методів відноситься метод розбиття на частини. Менеджер проєкту повинен розділити усі завдання, що складають більше ніж 10% від загального обсягу проєкту, на частини, що має допомогти більш точно визначити тривалість виконання великих завдань. Перевагою цього методу є його відносна простота та невеликий обсяг робіт. Недоліком є недостатній рівень точності та залежність від професіонального рівня програміста та менеджера проєкту. Також цей метод лише допомагає розділити великий обсяг роботи на малі та більш зрозумілі етапи, але все ж залишається проблема оцінювання тривалості на цих етапах.

Найбільш популярною неалгоритмічною моделлю є експертні оцінки. Цей метод найчастіше застосовується до проєктів, що мають справу з інноваційними завданнями або базуються на нових, інноваційних технологіях. У цьому випадку не існує достатньої кількості емпіричних даних для оцінювання тривалості або існуючих аналогів для застосування методу аналогій. Цей метод є дуже популярним, оскільки значна кількість створюваних програмних продуктів є інноваційними. Для визначення тривалості певного програмного продукту кожен з групи досвідчених експертів дає свої оцінки і за допомогою них формується інтегрована консенсусна оцінка.

Основним недоліком методів експертного оцінювання є наявність так званого «людського» фактору. Ми не можемо повністю довіряти інформації, що отримана від експерта, через її суб'єктивність. Це зумовлено неформалізованістю критеріїв, через що важко достовірно встановити кваліфікацію того чи іншого експерта. Також ми не можемо бути повністю впевнені у відсутності у них певних упередженостей.

Метод оцінки по аналогії є одним з різновидів експертної оцінки. Для оцінки тривалості у цьому методі використовуються певні емпіричні дані про попередньо завершені проєкти такого ж типу як і даний проєкт.

Алгоритмічні моделі у свою чергу базуються на певних математичних моделях. Для них є характерним постійний процес удосконалення, що призводить до підвищення точності оцінювання [7-10, 14, 15].

Program (Project) Evaluation and Review Technique (PERT) використовується у складних та масштабних проєктах. В першу чергу PERT використовується для визначення тривалості виконання кожної окремої задачі та проєкту в цілому. Цей метод припускає наявність певної невизначеності, що в свою чергу дозволяє, не маючи повної інформації про усі деталі та точної тривалості виконання кожного завдання, розробити графік проєкту. Найпопулярнішою частиною PERT є метод критичного шляху, який полягає у побудові мережевого графіку (мережеві діаграми PERT). Перевагами цього

методу є можливість визначити області, де необхідно зібрати додаткову інформацію, можливість графічного відображення проєкту та його головних складових, можливість виявити ключові завдання, які можуть викликати затримки у проєкті та завдання, що мають резервний час, а це в свою чергу допомагає розподілити ресурси більш раціонально. Недоліками цього методу є те, що при розробці мережевої діаграми деякі важливі дії можуть бути пропущені, а також те, що зв'язки та послідовність дій бувають відображені недостатньо чітко.

Модель Путнем (SLIM) в своїй основі має твердження, що витрачені на розробку програмного забезпечення ресурси розподіляються згідно з кривими Нордена-Рейлі. До недоліків цієї моделі можна віднести високу чутливість до значень технологічних факторів, при помилці у яких модель видаватиме недостовірні дані. Також дана модель застосовується для оцінювання тривалості проєктів, що мають більш ніж 70000 рядків програмного коду. Це робить її непридатною для оцінювання невеликих проєктів, що є суттєвим недоліком цього методу. Серед переваг цієї моделі можна відмітити невелику кількість параметрів, необхідних для оцінювання тривалості, що надає цій моделі значних переваг у порівняні з COCOMO та COCOMO II [14].

У 1991р. QSM Inc. представила власну реалізацію моделі SLIM, яка була покладена в основу програмного продукту для оцінювання тривалості розробки програмного забезпечення, що отримав назву SLIM Estimate. Ця нова модель поділяє багато недоліків та переваг з моделлю SLIM. Вона може бути використана лише тоді, коли трудомісткість розробки програмного забезпечення становить більш ніж 20 людино-місяців, що також робить її непридатною для оцінювання тривалості менших проєктів.

Найпопулярнішим сімейством алгоритмічних моделей є COCOMO (Constructive Cost Model) [15]. В основі даної моделі лежить формула регресії з параметрами. Ці параметри визначають за допомогою даних, отриманих із попередньо завершених проєктів. Модель складається із ієрархії 3 послідовно доповнюваних і розширюваних форм. Базовий етап (Basic) виконує

розрахунок тривалості виключно залежно від розміру програми, який визначається як кількість рядків програмного коду. Середній рівень (Intermediate) додатково також враховує додаткові фактори такі як: характеристики проєкту, кваліфікація персоналу та стан апаратного забезпечення. Детальний рівень (Advanced/Detailed) включає в себе характеристики середнього рівня та досліджує їх вплив на кожен з етапів створення програмного забезпечення. До переваг цієї методології можна віднести прозорість та зрозумілість, ми можемо бачити як вона працює на відміну від інших моделей, таких як SLIM. До недоліків цієї моделі можна віднести: неможливість створити точну оцінку тривалості на початковому етапі проєкту, тоді, коли вона є найбільш необхідною, результат використання цієї моделі дуже сильно залежить від правильності налаштування її до потреб саме даної організації, також для отримання оцінки необхідно використовувати історичні дані по попередньо завершеним проєктам, потрібно докласти багато зусиль для їх отримання, якщо кількість створених компанією проєктів не є достатньою для використання у цій моделі.

У 1999 р. була створена модифікація моделі COCOMO, що отримала назву COCOMO II. В моделі COCOMO II існують 2 основні етапи: попередня оцінка перед початком розробки програмного забезпечення та більш детальна оцінка, яка проводиться після створення архітектури. Суттєвими відмінностями моделі COCOMO II від своєї попередниці є використання функціональних точок та об’єктно-орієнтовані підходи до оцінювання.

При аналізі останніх досліджень та публікацій були знайдені нелінійні регресійні моделі [7 - 11] для оцінювання тривалості розробки програмного забезпечення. Проте не було знайдено моделей для оцінювання тривалості розробки ПЗ, створеного за методологією Agile. Саме тому виникає необхідність удосконалення математичної моделі оцінювання тривалості розробки програмного забезпечення, яке створюється за Agile-методологією. Для нормалізації негаусівських даних було прийнято рішення використовувати метод десяткового логарифму. Розроблена модель дасть

можливість підвищити достовірність оцінювання тривалості розробки програмного забезпечення порівняно з існуючими моделями.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

– обґрунтувати необхідність розробки нелінійної регресійної моделі для оцінювання тривалості розробки програмного забезпечення за методологією Agile;

– обрати в якості нормалізуючого перетворення для емпіричних даних, отриманих із попередньо завершених проєктів, десятковий логарифм;

– нормалізувати отримані емпіричні дані;

– перевірити нормалізовані дані на викиди;

– побудувати лінійну регресійну модель, довірчий інтервал та інтервал прогнозування для нормалізованих даних;

– перейти від лінійної регресійної моделі до нелінійної та побудувати нелінійну регресійну модель, довірчий інтервал та інтервал прогнозування для вихідних даних;

– побудувати лінійну регресійну модель та довірчий інтервал і інтервал передбачення для неї без нормалізації даних в припущенні про нормальність вихідних даних;

– виконати порівняння лінійної моделі з нелінійною;

– розробити ПЗ для оцінювання тривалості розробки програмного забезпечення, створеного за методологією Agile за розробленою методикою.

Для вирішення поставлених задач будуть використані методи теорії ймовірностей та математичної статистики, регресійного аналізу, побудови нелінійних регресійних моделей на основі нормалізуючих перетворень.

2.2 Перевірка якості математичної моделі для оцінювання тривалості розробки програмного забезпечення

Для перевірки якості регресійної моделі використаємо коефіцієнт детермінації R^2 [16]:

$$R^2 = 1 - \left(\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 / \sum_{i=1}^{n} (y_i - \bar{y})^2 \right), \quad (2.1)$$

де y_i – емпіричне значення y;

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

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

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

Значення $R^2 \geq 0{,}5$ являється прийнятним. При $R^2 \geq 0{,}8$ модель вважається достатньо ефективною та результативною можна. $R^2 = 1$ означає, що модель являється достовірною та адекватною.

Величина коефіцієнта детермінації використовується для оцінки якості як лінійних так і нелінійних моделей.

Також для перевірки якості лінійних і нелінійних моделей, окрім критерію R^2, використовується сума квадратів відхилень S_y [17]:

$$S_y = \sum_{i=1}^{n} [y_i - \hat{y}_i]^2, \quad (2.2)$$

де y_i – фактичне значення випадкової величини y;

$\hat{y}_i$ – розрахункове значення згідно рівняння регресії.

Також для перевірки якості регресійного рівняння використаємо величину відносної похибки MMRE та рівень прогнозування PRED(x). Для обчислення цих величин використовується величина відносної похибки MRE, яка обчислюється із співвідношення:

$$MRE_i = \left| \frac{y_i - \hat{y}_i}{y_i} \right|, \tag{2.3}$$

де $\hat{y}_i$– значення y, розраховане за рівнянням регресії;

y_i – емпіричні значення залежної змінної y.

Співвідношення для обчислення MMRE наведене в (2.4):

$$MMRE = \frac{1}{N} \sum_{i=1}^{N} MRE_i \tag{2.4}$$

PRED(l) обчислюється за формулою 2.5:

$$PRED(l) = \frac{k}{N}, \tag{2.5}$$

де k – кількість тих MRE_i, які не перевищують l.

2.3 Обґрунтування необхідності проведення досліджень

Із аналізу методів та моделей для оцінювання тривалості розробки програмного забезпечення за методологією Agile випливає, що вони не дають змогу оцінити в повній мірі тривалості розробки програмного забезпечення, щоб при цьому одночасно враховувати і методологію Agile і нелінійні зв’язки у вихідних даних. А отже є необхідність дослідити способи побудови математичної моделі, яка б покращила достовірність передбачення та дала

змогу швидко та надійно оцінити трудомісткість розробки програмного забезпечення за методологією Agile.

В реальних проєктах майже неможливо отримати данні, які будуть розподілені за нормальним законом, і зв’язки між цими даними не завжди є лінійними. Виходячи з цього, у математичній моделі, яка розробляється, необхідно брати до уваги нелінійність зв’язків вихідних даних і виконувати нормалізацію даних.

3 ПОБУДОВА НЕЛІНІЙНОЇ РЕГРЕСІЙНОЇ МОДЕЛІ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, СТВОРЕНОГО ЗА МЕТОДОЛОГІЄЮ AGILE, З ВИКОРИСТАННЯМ НОРМАЛІЗУЮЧОГО ПЕРЕТВОРЕННЯ НА ОСНОВІ ДЕСЯТКОВОГО ЛОГАРИФМУ

Для побудови нелінійних регресійних моделей можна використовувати методи простого перебору або нормалізуючих перетворень.

Метод простого перебору полягає у заданні різних видів рівняння регресії, з яких потім, використовуючи певні критерії, обирається найкраще наближення. Метод має значний недолік. Він потребує значних ресурсів часу, так як кількість обчислень досить значна.

При використанні методу нормалізуючих перетворень, необхідно знайти таке перетворення, з допомогою якого перейдемо від ненормалізованих емпіричних даних до нормалізованих. Потім для нормалізованих вихідних даних необхідно побудувати лінійну регресійну модель, з якої отримаємо нелінійну модель через зворотне перетворення.

Використовуючи метод нормалізуючих перетворень, ми можемо обробляти дані з нелінійною залежністю, і тому ми будемо використовувати саме цей метод. Для нормалізації даних було обрано перетворення десяткового логарифму.

Аналіз існуючих методів оцінювання тривалості розробки програмного забезпечення в умовах гнучкої методології розробки показав, що розроблювана модель відповідає вимогам та зумовлює пришвидшення та спрощення виконання необхідних розрахунків. Ці вимоги стосуються факторів, які використовуються для побудови моделі [18].

3.1 Побудова нелінійної регресійної моделі оцінювання тривалості розробки програмного забезпечення за методологією Agile

Нелінійна регресійна модель оцінювання тривалості розробки програмного забезпечення за методологією Agile використовує параметричне оцінювання тривалості розробки ПЗ на основі наступних факторів:

- трудомісткість розробки ПЗ (X) – загальна кількість зусиль у годинах, записаних для проєкту;
- тривалість розробки ПЗ (Y) – загальний час виконання проєкту в календарних місяцях.

Тому тривалість розробки ПЗ може бути розрахована як функція від одного фактору:

$$Y = f(X) \tag{3.1}$$

Емпіричні дані для побудови моделі були обрані із репозитарію International Software Benchmarking Standards Group з релізу 2021 року [19], який отримано згідно з ліцензійною угодою, що визначає порядок аналізу отриманої інформації. Було відібрано проєкти з методологією розробки Agile (Development Methodologies – Agile) та типом розробки – нові розробки (Development Type – New). За цими критеріями відібрано 39 проєктів з атрибутами трудомісткість (Summary Work Effort) та тривалість розробки (Project Elapsed Time).

Емпіричні дані не підпорядковуються нормальному закону розподілу. Це було виявлено шляхом використання критерію χ^2 Пірсона.

Це означає, що для побудови нелінійної регресійної моделі необхідно виконати нормалізацію емпіричних даних [20, 21].

Нормалізацію емпіричних даних було виконано за допомогою десяткового логарифму. В результаті було отримано нормалізовані величини Z_Y та Zx.

Обрані дані було перевірено на наявність викидів. В результаті 10 ітерацій було вилучено 9 викидів. При вилученні викидів використовували квадрат відстані Махаланобіса, критерій χ^2 та інтервал прогнозування.

Після видалення викидів залишилося 30 проєктів, по яким буде будуватися модель.

Нелінійна регресія будується на основі лінійної регресійної моделі. Лінійна регресійна модель з двома параметрами виглядає наступним чином [22-24]:

$$\hat{Z}_y = \mathrm{b}_0 + \mathrm{b}_1 \cdot Z_x + \varepsilon\,, \tag{3.2}$$

де $\mathrm{b}_0, \mathrm{b}_1$ – коефіцієнти лінійної регресії;
ε – випадкова помилка.

Значення коефіцієнтів були розраховані за методом найменших квадратів:

b_1 = 0,483; b_0 = -1,017.

Після виконання нормалізації емпіричних даних та отримання лінійної регресійної моделі, перевірили залишки на нормальність розподілу. Так як залишки мають гаусівський розподіл, то можна побудувати довірчий інтервал для цієї моделі та інтервал передбачення. Довірчий інтервал розраховується за формулою (3.3) [20, 23]:

$$\hat{Z}_y \pm t_{\alpha/2,n-2} \cdot S_{Z_y} \sqrt{\frac{1}{n} + \frac{(Z_x - \overline{Z_x})^2}{\sum_n^1 (z_{xi} - \overline{Z_x})^2}}\,, \tag{3.3}$$

де $\hat{Z}_y$ – значення, розраховані за рівнянням регресії;

$t_{\alpha/2,n-2}$ – квантіль t-розподілу Стьюдента;

α – рівень статистичної значущості;

n – кількість елементів в вибірці;

$$S_{Z_y} = \sqrt{\frac{1}{n-2} \cdot \sum\nolimits_1^n \left(z_{yi} - \hat{z}_{y_i}\right)^2}\,.$$

Далі будуємо інтервал передбачення для цієї моделі за формулою (3.4) [20, 23] :

$$\hat{Z}_y \pm t_{\alpha/2,n-2} \cdot S_{Z_y} \sqrt{1 + \frac{1}{n} + \frac{(Z_x - \overline{Z_x})^2}{\sum_n^1 (z_{x_i} - \overline{Z_x})^2}}. \quad (3.4)$$

Далі необхідно перейти до вихідних емпіричних даних [20]. Для цього застосуємо формули зворотного перетворення для функції і для верхньої та нижньої границь довірчого інтервалу та інтервалу прогнозування лінійної регресії та отримаємо функцію і верхню та нижню границі довірчого інтервалу та інтервалу прогнозування для нелінійної регресійної моделі.

Нелінійна регресійна модель оцінювання тривалості розробки програмного забезпечення, створеного за методологією Agile має вигляд:

$$\hat{Y} = 10^{b_0+\varepsilon} \cdot X^{b_1} .$$

Отримаємо

$$\hat{Y} = 10^{-1,017+\varepsilon} * X^{0,483}.$$

В табл. 3.1 знаходяться результати виконаних розрахунків: границі довірчих інтервалів емпіричних даних та їх довжини, а також відповідні значення для нормалізованих даних.

Таблиця 3.1 – Границі довірчих інтервалів емпіричних даних та їх довжини, а також відповідні значення для нормалізованих даних

№ п / п	Ліва границя (вихідні дані)	Права границя (вихідні дані)	Довжина (вихідні дані)	Ліва границя (норм)	Права границя (норм)	Довжина (норм)
1	2	3	4	5	6	7
1	0,8722	2,3032	1,4310	0,6867	1,0077	0,3210
2	0,9150	2,3359	1,4208	0,8736	1,2198	0,3461
3	0,9360	2,3519	1,4159	0,9531	1,3082	0,3551
4	0,9378	2,3532	1,4155	0,9595	1,3153	0,3558

Продовження табл. 3.1

1	2	3	4	5	6	7
5	0,9395	2,3546	1,4151	0,9658	1,3223	0,3565
6	0,9535	2,3653	1,4118	1,0150	1,3766	0,3617
7	0,9648	2,3740	1,4091	1,0533	1,4190	0,3656
8	0,9709	2,3786	1,4077	1,0735	1,4411	0,3676
9	0,9709	2,3786	1,4077	1,0735	1,4411	0,3676
10	1,0128	2,4108	1,3980	1,2029	1,5833	0,3804
11	1,0206	2,4169	1,3962	1,2257	1,6083	0,3826
12	1,0467	2,4370	1,3903	1,2988	1,6886	0,3898
13	1,0485	2,4384	1,3899	1,3036	1,6938	0,3903
14	1,1898	2,5484	1,3586	1,6429	2,0707	0,4278
15	1,4227	2,7333	1,3106	2,0789	2,5768	0,4979
16	1,5598	2,8445	1,2847	2,2925	2,8377	0,5452
17	1,5971	2,8750	1,2780	2,3467	2,9055	0,5587
18	1,7463	2,9988	1,2525	2,5501	3,1652	0,6151
19	2,0889	3,2929	1,2040	2,9563	3,7110	0,7546
20	2,6942	3,8534	1,1592	3,5493	4,5681	1,0188
21	2,8384	3,9960	1,1576	3,6757	4,7592	1,0835
22	3,3577	4,5422	1,1845	4,0999	5,4203	1,3204
23	3,9169	5,1903	1,2734	4,5167	6,0971	1,5804
24	4,4770	5,8977	1,4207	4,9034	6,7467	1,8433
25	4,6958	6,1878	1,4920	5,0473	6,9934	1,9461
26	4,9092	6,4771	1,5679	5,1841	7,2303	2,0463
27	5,9435	7,9484	2,0050	5,8014	8,3274	2,5260
28	7,2232	9,8667	2,6435	6,4762	9,5750	3,0987
29	8,1040	11,2203	3,1163	6,8949	10,3724	3,4774
30	8,6334	12,0417	3,4083	7,1317	10,8309	3,6991

Таблиця 3.2 – Інтервали прогнозування для емпіричних величин та їх довжини, а також відповідні значення для нормалізованих даних

№ п / п	Ліва границя (вихідні дані)	Права границя (вихідні дані)	Довжина (вихідні дані)	Ліва границя (норм)	Права границя (норм)	Довжина (норм)
1	2	3	4	5	6	7
1	-1,6623	4,8376	6,4999	0,4498	1,5384	1,0886
2	-1,6234	4,8743	6,4976	0,5623	1,8952	1,3330
3	-1,6043	4,8922	6,4965	0,6096	2,0454	1,4358
4	-1,6027	4,8937	6,4964	0,6134	2,0574	1,4440
5	-1,6011	4,8952	6,4964	0,6171	2,0693	1,4522
6	-1,5884	4,9072	6,4956	0,6463	2,1619	1,5156
7	-1,5781	4,9169	6,4951	0,6690	2,2341	1,5651
8	-1,5726	4,9222	6,4948	0,6809	2,2719	1,5910
9	-1,5726	4,9222	6,4948	0,6809	2,2719	1,5910
10	-1,5345	4,9581	6,4927	0,7573	2,5149	1,7576
11	-1,5274	4,9649	6,4923	0,7707	2,5577	1,7870
12	-1,5036	4,9874	6,4910	0,8138	2,6951	1,8813
13	-1,5021	4,9889	6,4909	0,8166	2,7040	1,8874
14	-1,3731	5,1112	6,4843	1,0168	3,3459	2,3291
15	-1,1592	5,3152	6,4744	1,2779	4,1919	2,9140
16	-1,0325	5,4367	6,4692	1,4084	4,6190	3,2106
17	-0,9979	5,4700	6,4679	1,4419	4,7288	3,2870
18	-0,8589	5,6040	6,4629	1,5684	5,1462	3,5778
19	-0,5359	5,9178	6,4537	1,8266	6,0061	4,1794
20	0,0511	6,4965	6,4455	2,2150	7,3198	5,1048
21	0,1946	6,6398	6,4452	2,2993	7,6081	5,3088
22	0,7249	7,1750	6,4501	2,5855	8,5951	6,0096
23	1,3201	7,7871	6,4670	2,8710	9,5921	6,7212
24	1,9386	8,4362	6,4976	3,1390	10,5390	7,4000
25	2,1850	8,6986	6,5135	3,2394	10,8964	7,6570
26	2,4275	8,9588	6,5314	3,3352	11,2386	7,9034

Продовження табл. 3.2

1	2	3	4	5	6	7
27	3,6211	10,2709	6,6498	3,7709	12,8113	9,0404
28	5,1103	11,9797	6,8694	4,2529	14,5804	10,3275
29	6,1297	13,1945	7,0648	4,5545	15,7024	11,1478
30	6,7383	13,9367	7,1984	4,7259	16,3447	11,6188

З таблиць видно, що довжини більшості довірчих інтервалів та інтервалів прогнозування менші для нормалізованих даних, ніж відповідні довжини для ненормалізованих даних. Також границі інтервалів прогнозування для нормалізованих даних не містять від'ємних значень.

Для ненормалізованих та нормалізованих за методом десяткового логарифму даних були пораховані R^2, MMRE та PRED(0,25). Для ненормалізованих даних: R^2 = 0,7893; MMRE = 0,3616; PRED(0,25) = 0,4667. Для даних, нормалізованих за рахунок десяткового логарифму: R^2 = 0,8012; MMRE = 0,1924; PRED(0,25) = 0,7.

Порівнявши ці результати ми можемо побачити, що по R^2 всі підходи дають допустимий результат, PRED(0,25) гірший у ненормалізованих даних, MMRE також гірший у ненормалізованих даних.

На рисунку 3.1 зображені довірчий інтервал, інтервал передбачення та саме рівняння регресії для ненормалізованих даних.

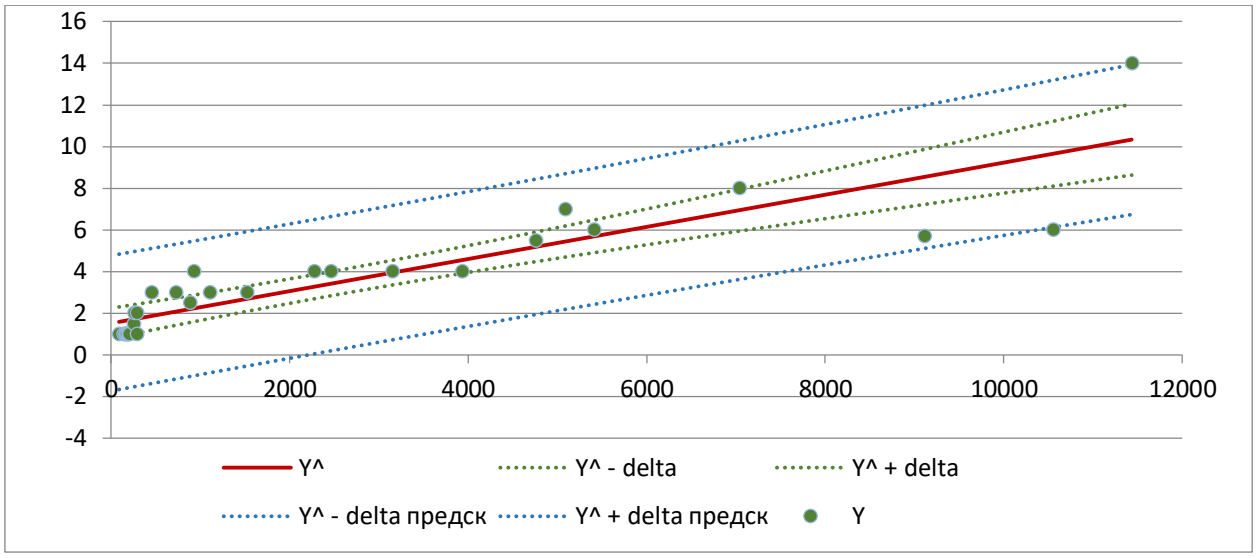

Рисунок 3.1 – Довірчий інтервал, інтервал передбачення та саме рівняння регресії для ненормалізованих даних.

На рисунку 3.2 зображені довірчий інтервал, інтервал передбачення, нелінійне рівняння регресії для даних, нормалізованих за допомогою десяткового логарифму.

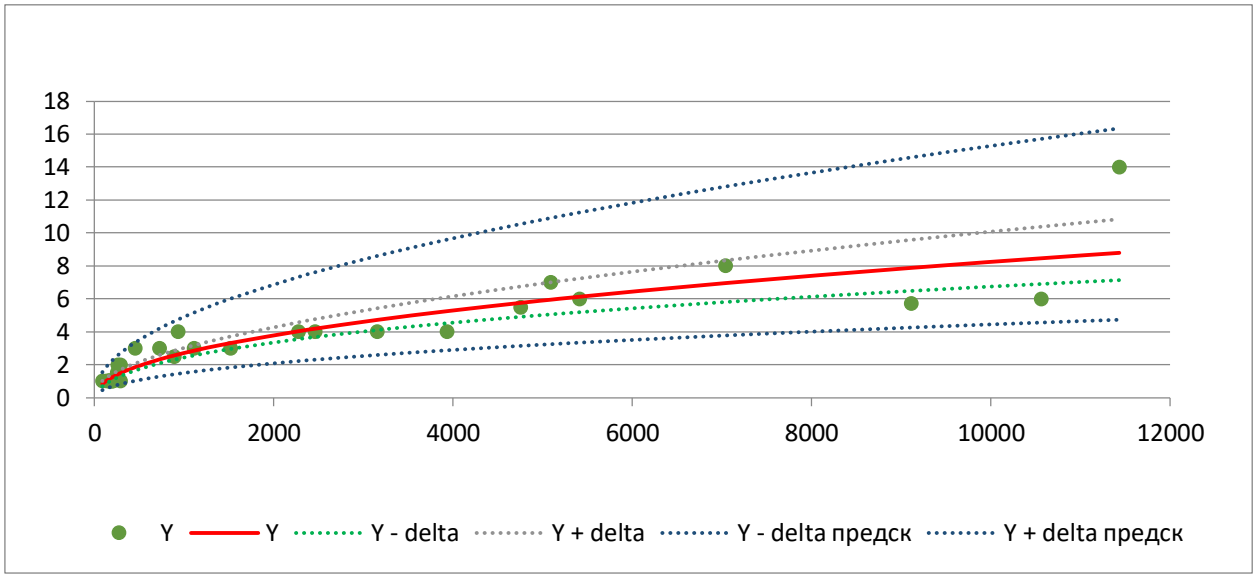

Рисунок 3.2 – Довірчий інтервал, інтервал передбачення, нелінійне рівняння регресії для даних, нормалізованих за допомогою десяткового логарифму

Таким чином, побудована нелінійна регресійна модель має більшу точність оцінювання тривалості розробки програмного забезпечення.

3.2 Постановка задачі на розробку ПЗ для оцінювання тривалості розробки програмного забезпечення, створеного за методологією Agile

Виконавши аналіз існуючих методів оцінювання тривалості розробки програмного забезпечення за методологією Agile, можна сформулювати наступну постановку задачі: розробити проєкт програмного забезпечення для оцінювання тривалості розробки програмних продуктів, створених за методологією Agile. Для розробки ПЗ слід побудувати математичну модель, яка допоможе оцінити тривалість розробки програмного забезпечення за наявності негаусівських емпіричних даних. В якості нормалізуючого перетворення обрано десятковий логарифм.

Розглянемо функції програмного забезпечення:

1) Внесення даних про програмні проєкти;

2) Нормалізація вихідних даних (десятковий логарифм);

3) Побудова для нормалізованих даних лінійної регресії, довірчого інтервалу та інтервалу передбачення;

4) Побудова нелінійної регресії, довірчого інтервалу та інтервалу передбачення для неї через зворотне перетворення;

5) Розрахунок характеристик якості;

6) Оцінювання тривалості розробки програмного забезпечення, її довірчого інтервалу та інтервалу передбачення за допомогою нелінійної регресії.

Емпіричні дані для побудови моделі були обрані із репозитарію International Software Benchmarking Standards Group з релізу 2021 року [19], який отримано згідно з ліцензійною угодою, що визначає порядок аналізу

отриманої інформації. Було відібрано проєкти з методологією розробки Agile (Development Methodologies – Agile) та типом розробки – нові розробки (Development Type – New). За цими критеріями відібрано 39 проєктів з атрибутами трудомісткість (Summary Work Effort) та тривалість розробки (Project Elapsed Time). Емпіричні дані не підпорядковуються нормальному закону розподілу. Це було виявлено шляхом використання критерію χ^2 Пірсона. Після вилучення викидів залишилося 30 даних для побудови моделі.

4 ПРОЄКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ РОЗРОБКИ ПЗ, СТВОРЕНОГО ЗА МЕТОДОЛОГІЄЮ AGILE

Проєкт програмного забезпечення представлений ескізним, технічним та робочим проєктом. При розробці проєкту застосовувався об'єктно-орієнтований підхід, мова моделювання UML.

4.1 Ескізній проєкт програмного забезпечення для оцінювання тривалості розробки ПЗ, створеного за методологією Agile

В ескізному проєкті представлені: діаграма прецедентів, специфікації варіантів використання, діаграми діяльності, проєкт користувальницького інтерфейсу та концептуальна модель даних.

4.1.1 Побудова діаграми прецедентів ПЗ

Для зображення відношень між користувачами системи та самою системою розроблено діаграму прецедентів програмного забезпечення.

Опис суб'єктів взаємодії представлений в таблиці 4.1

Таблиця 4.1 – Опис акторів

Суб'єкти взаємодії	Прецеденти
Користувач	Вся робота з системою виконується одним користувачем, який може ввести дані, нормалізувати дані, побудувати нелінійну регресійну модель, довірчий інтервал та інтервал передбачення, перевірити якість моделі та побудувати графіки, оцінити тривалість розробки нового проєкту

Функції розроблюваного програмного забезпечення представлені у вигляді прецедентів в таблиці 4.2.

Таблиця 4.2 – Виявлені прецеденти

Формулювання	Опис використання
1	2
Введення даних з файлу	Введення емпіричних даних з обраного файлу
Визначення довірчих інтервалів	Розрахунок довірчих інтервалів
Визначення інтервалів передбачення	Розрахунок інтервалів передбачення
Визначення метрик точності моделі	Визначення коефіцієнта детермінації, середньої величини відносної похибки та рівня прогнозування
Нормалізація даних	Нормалізація емпіричних даних за допомогою функції десятковий логарифм
Визначення коефіцієнтів регресії	Знаходження коефіцієнтів регресії методом найменших квадратів
Побудова графіків	Побудова графіків функції, довірчих інтервалів та інтервалів передбачення.
Оцінювання	Оцінювання тривалості розробки нового проєкту

Діаграма прецедентів програмного забезпечення для оцінювання тривалості розробки програмного забезпечення**,** створеного за методологією Agile, наведена на рис. 4.1.

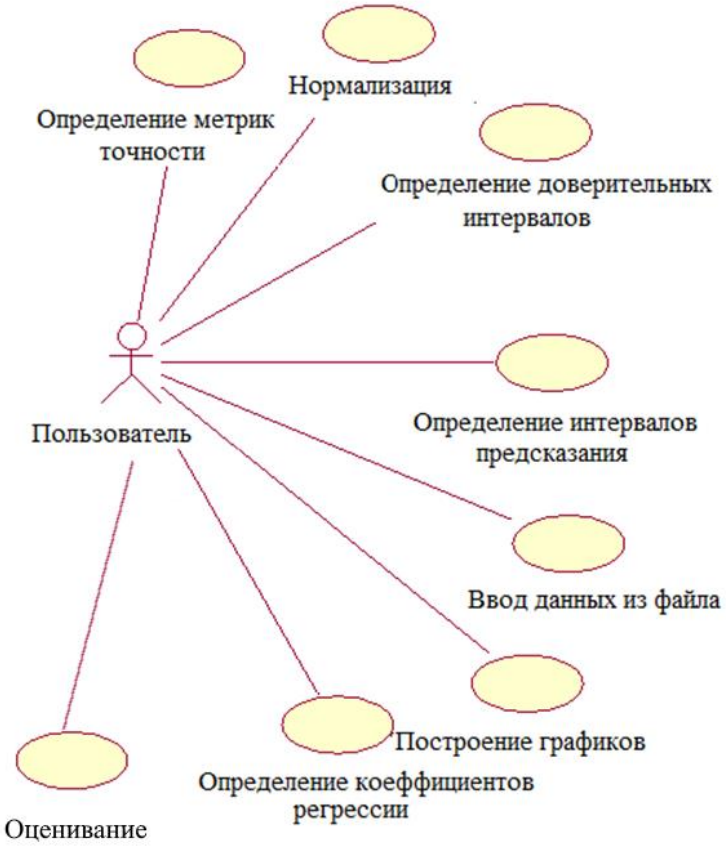

Рисунок 4.1 – Діаграма прецедентів ПЗ для оцінювання тривалості розробки програмного забезпечення**,** створеного за методологією Agile

4.1.2 Специфікації прецедентів програмного забезпечення для оцінювання тривалості розробки ПЗ, створеного за методологією Agile

Для деталізації прецедентів з рисунку 4.1, розробимо їх специфікації.

Прецедент «Визначення метрик точності моделі»

Актор: Користувач.

Короткий опис: Прецедент відповідає за обчислення коефіцієнта детермінації, середньої величини відносної похибки та рівня прогнозування для регресійного рівняння.

Базовий потік подій: Визначаються метрики точності:

1. Обчислюється R^2.
2. Обчислюється MMRE.

3. Обчислюється PRED(x).
4. Виводяться метрики якості.

Альтернативні потоки подій: Не визначені.

Передумова: Введення інформації з файлу, визначені коефіцієнти регресії.

Післяумова: Не визначена.

Прецедент «Нормалізація даних»

Актор: Користувач.

Короткий опис: Прецедент відповідає за нормалізацію емпіричних даних за допомогою функції десятковий логарифм.

Базовий потік подій: Програма логарифмує вихідні дані з файлу.

Альтернативні потоки подій: Не визначені.

Спеціальні вимоги: Не визначені.

Передумова: Наявність файлу з емпіричними даними.

Післяумова: Не визначена.

Прецедент «Визначення довірчих інтервалів»

Актор: Користувач.

Короткий опис: Прецедент визначає інтервали довіри побудованого рівняння регресії.

Базовий потік подій:

1. ПЗ визначає границі інтервалу довіри регресії
2. ПЗ друкує при необхідності інтервали довіри.

Альтернативні потоки подій: Не визначені.

Спеціальні вимоги: Не визначені.

Передумова: Наявність побудованого рівняння регресії.

Післяумова: Не визначена.

Прецедент «Визначення інтервалів прогнозування»

Актор: Користувач.

Короткий опис: Прецедент відповідає за визначення прогнозних інтервалів побудованого рівняння регресії.

Базовий потік подій:

1. ПЗ визначає границі прогнозного інтервалу регресії.

2. ПЗ друкує при необхідності прогнозні інтервали регресії.

Альтернативні потоки подій: Не визначені.

Передумова: Наявність побудованого рівняння регресії.

Післяумова: Не визначена.

Прецедент «Визначення коефіцієнтів регресії»

Актор: Користувач.

Короткий опис: Прецедент відповідає за знаходження коефіцієнтів регресії методом найменших квадратів.

Базовий потік подій: ПЗ визначає коефіцієнти регресії методом найменших квадратів.

Альтернативні потоки подій: Не визначені.

Передумова: Наявність введених та нормалізованих даних.

Післяумова: Не визначена.

Прецедент «Введення даних з файлу»

Актор: Користувач.

Короткий опис: Прецедент відповідає за вибір файлу та введення вихідних даних без дублікатів та викидів.

Базовий потік подій:

1. Вибір файлу користувачем.

2. Введення даних з файлу.

Альтернативні потоки подій:

1. Вибір файлу користувачем.

2. Введення даних з файлу та виведення повідомлення про невідповідність формату.

Передумова: Не визначена.

Післяумова: Не визначена.

Прецедент «Побудова графіків»

Актор: Користувач.

Короткий опис: Прецедент відповідає за побудову графіків довірчих інтервалів та інтервалів передбачення

Базовий потік подій: Програма побудови графіків інтервалів довіри та інтервалів передбачення для вихідних та нормалізованих даних.

Альтернативні потоки подій: Не визначені.

Передумова: Наявність введених і нормалізованих даних, а також розрахованого передбачення

Післяумова: Не визначена.

4.1.3 Діаграма діяльності ПЗ для оцінювання тривалості розробки програмного забезпечення, створеного за методологією Agile

Діаграми діяльності (activity diagrams) відображають послідовність дій, що виконується в процесі реалізації певного варіанта використання або функціонування системи в цілому. Діаграми діяльності є аналогом блок-схеми будь-якого алгоритму.

Для подальшої деталізації варіантів використання розроблялися діаграми діяльності.

На рис. 4.2.представлена діаграма діяльності для варіанту використання «Введення даних з файлу».

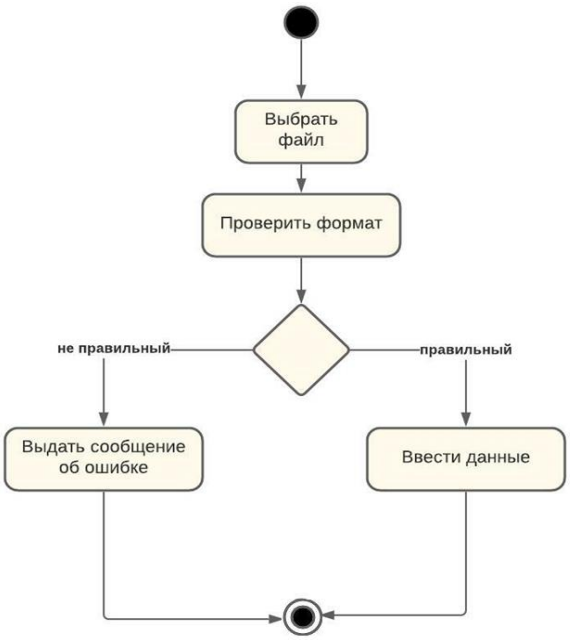

Рисунок 4.2 – Діаграма діяльності для варіанту використання «Введення даних з файлу»

На рис. 4.3 представлена діаграма діяльності для варіанту використання «Визначення коефіцієнтів регресії».

Рисунок 4.3 – Діаграма діяльності для варіанту використання «Визначення коефіцієнтів регресії».

Таким чином, діаграми діяльності мають чималі виразні засоби, що дозволяють будувати різноманітні програмні системи.

4.1.4 Діаграма сутність-зв'язок ПЗ для оцінювання тривалості розробки програмного забезпечення за методологією Agile

Концептуальне проєктування – це створення уявлення (схеми, моделі) даних, що включає визначення сутностей, атрибутів і зв'язків між ними, але не залежить від моделі даних (ієрархічної, мережевої, реляційної і т. д.) і фізичної реалізації.

Концептуальна модель даних програми для оцінювання тривалості розробки програмного забезпечення за методологією Agile у вигляді діаграми сутність-зв'язок представлена на рисунку 4.4.

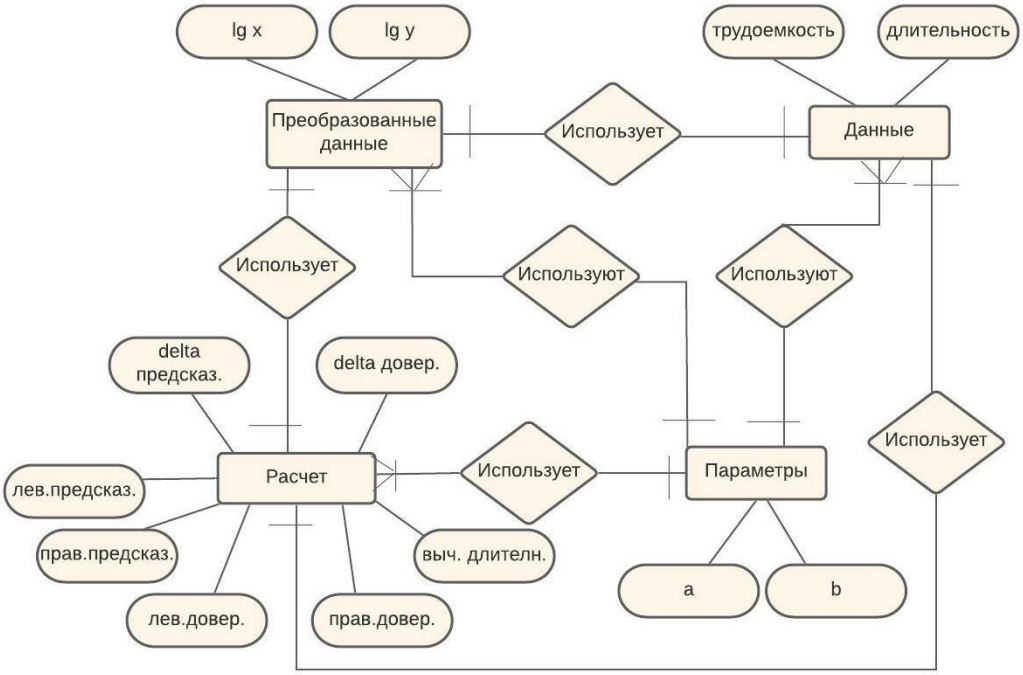

Рисунок 4.4 – Концептуальна модель даних програми для оцінювання тривалості розробки програмного забезпечення за методологією Agile

Як бачимо, на діаграмах представляються сутності предметної галузі, їх властивості та взаємозв'язки між сутностями.

4.1.5 Розробка інтерфейсу користувача ПЗ для оцінювання тривалості розробки програмного забезпечення за методологією Agile

Робота з програмним забезпеченням відбувається за допомогою інтерфейсу користувача. Основуючись на аналізі моделей інтерфейсів, діаграмі варіантів використання, а також на вимогах, які пред'явлені в технічному завданні, був визначений зовнішній інтерфейс.

Ескіз інтерфейсу наведений на рисунку 4.5.

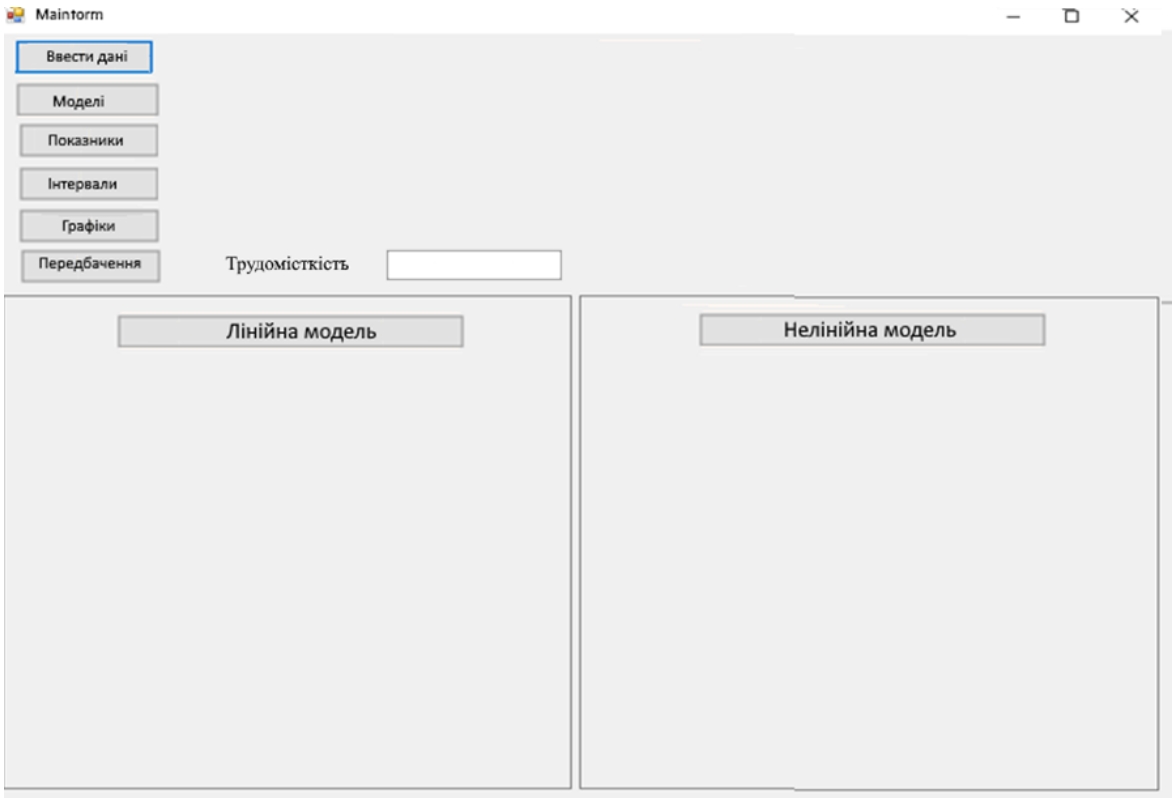

Рисунок 4.5 – Ескіз головного меню програми

Таким чином, в головному вікні є можливість обрати файл з вихідними даними; побудувати лінійну та нелінійну моделі для оцінювання тривалості розробки програмного забезпечення за методологією Agile; розрахувати такі

показники якості, як коефіцієнт детермінації, середня величина відносної похибки та рівень прогнозування; вивести на екран інтервали довіри та передбачення для побудованих моделей; зобразити графіки інтервалів, вихідних даних та розрахованих залежних змінних для двох типів моделей.

4.2 Технічний проєкт ПЗ для оцінювання тривалості розробки програмного забезпечення за методологією Agile

Під час розробки технічного проєкту ПЗ для оцінювання тривалості розробки програмного забезпечення за методологією Agile були досліджені шляхи реалізації необхідних функцій. На етапі технічного проєктування вимоги трансформуються в таку форму, яка відповідає реалізації. Весь проєкт підлягає декомпозиції на такі складові, реалізація яких не становить великої складності і які при об’єднанні в єдиний виріб не створять жодної проблеми.

4.2.1 Діаграма класів ПЗ для оцінювання тривалості розробки програмного забезпечення

Діаграму класів використовують для зображення статичної структури проєктованої системи. Діаграма класів оперує поняттями класу, інтерфейсу, об'єкту, відношення та пакету.

З допомогою діаграми класів можна представляти концептуальний рівень проєктування системи (поняття предметної галузі), рівень специфікацій (операції класу, які видимі ззовні), рівень реалізації (реалізацію класів).

Діаграма класів програми для оцінювання тривалості розробки програмного забезпечення за методологією Agile зображена на рисунку 4.6. На діаграмі представлені наступні класи:

– Клас Program, основним призначенням якого є запуск програми.

– Клас Regres, який реалізує процедуру отримання коефіцієнтів регресійного рівняння.

– Клас MainForm, основною зоною відповідальності якого графічний інтерфейс програми.

– Клас FileRead, який відповідає за зчитування даних з файлу.

– Клас lgPreobr клас призначений для нормалізації вихідних даних даних за допомогою перетворення десяткового логарифму і для подальшого зворотнього перетворення.

– Клас NotLinear клас, який відповідає за основні розрахунки.

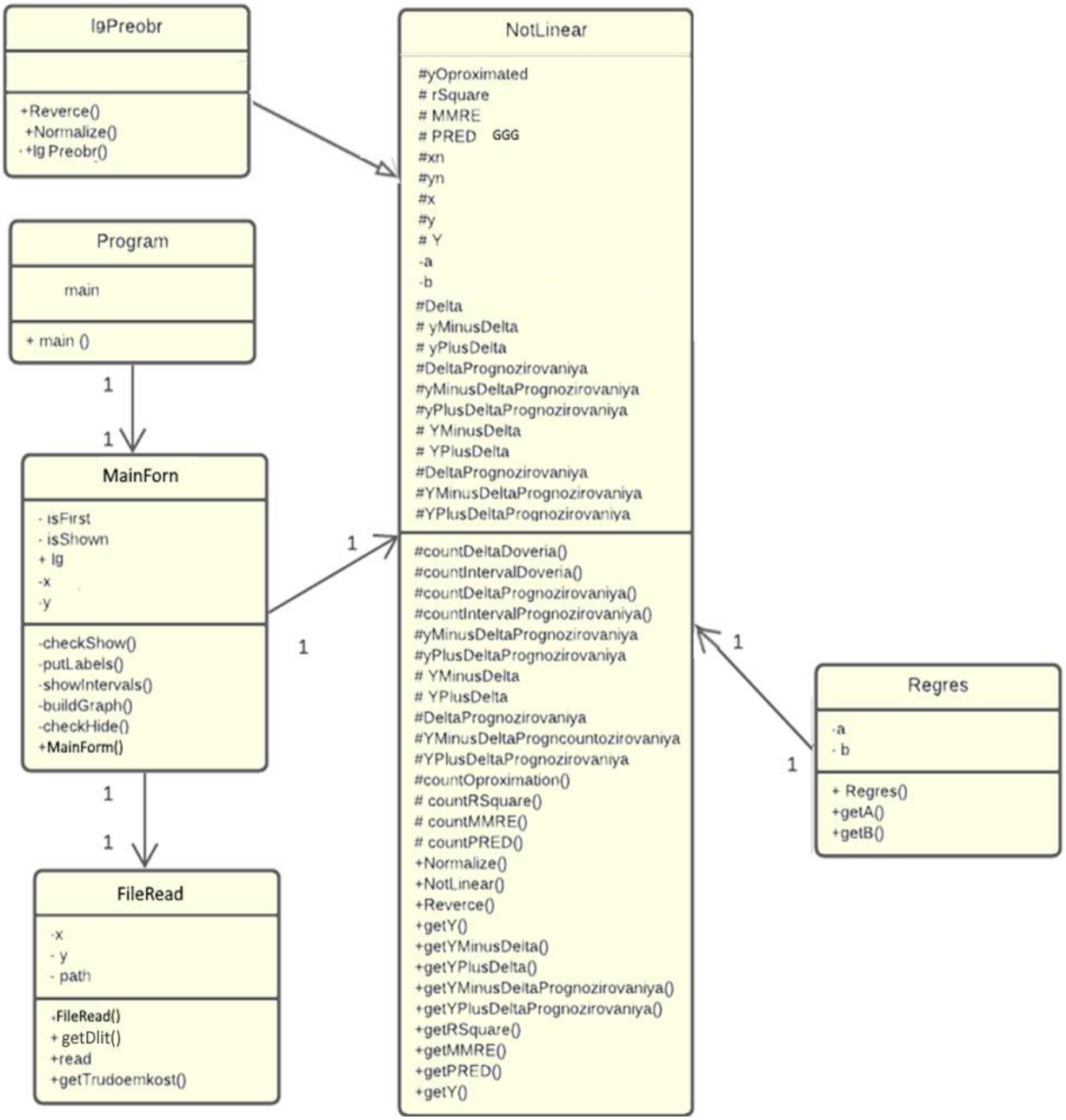

Рисунок 4.6 – Діаграма класів програми для оцінювання тривалості розробки програмного забезпечення за методологією Agile

Таким чином, у роботі розроблена діаграма класів, яка представляє статичну структуру моделі системи.

4.2.2 Динамічне представлення моделі системи для оцінювання тривалості розробки програмного забезпечення

Діаграма класів призначена для статичного представлення системи, що моделюється. Однак елементи системи завжди взаємодіють між собою.

У мові UML це взаємодія елементів у інформаційному аспекті, тобто об'єкти обмінюються деякою інформацією.

Діаграми взаємодій є моделями, що описують поведінку взаємодіючих груп об'єктів.

Існують 2 види діаграм взаємодій:

1) діаграми послідовності дій – sequence diagram;

2) діаграми кооперації (кооперативні діаграми) – collaboration diagram.

Перший тип діаграм відображає взаємодію об'єктів, яка упорядкована за часом, другий – описує поведінку на рівні об’єктів, що обмінюються повідомленнями. Основними елементами діаграм послідовності є: лінії життя, фокус керування, символ знищення, об’єкти, повідомлення. Складовими діаграм кооперацій є: зв’язки, об’єкти, повідомлення. Перелічені діаграми можна будувати як для всієї системи в цілому, так і для окремих варіантів використання.

На рис. 4.7 представлено діаграму послідовності для варіанту використання «Визначення метрик точності» програми для оцінювання тривалості розробки програмного забезпечення за методологією Agile

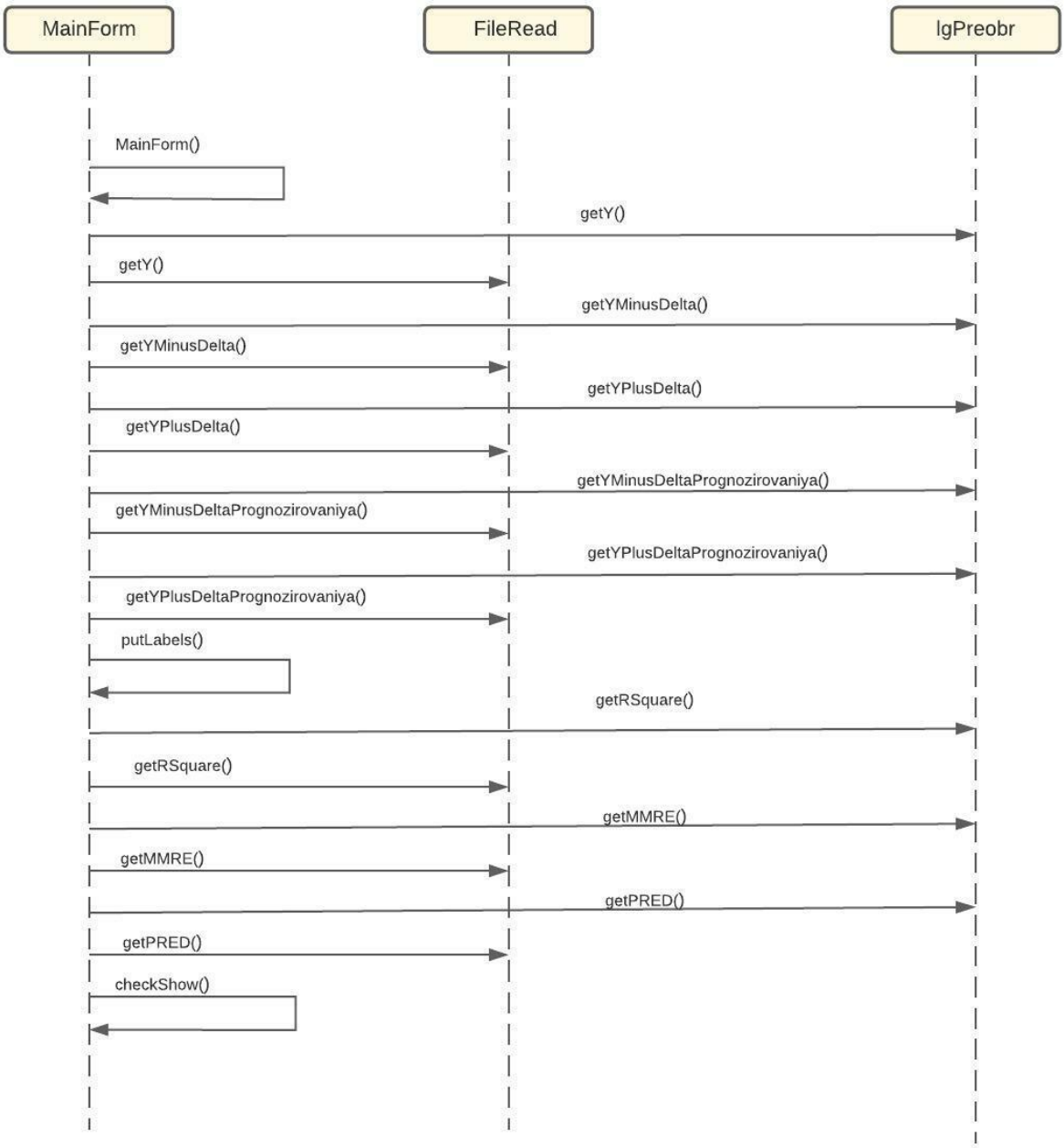

Рисунок 4.7 – Діаграма послідовності для варіанту використання «Визначення метрик точності» програми для оцінювання тривалості розробки програмного забезпечення за методологією Agile

4.3 Робочий проєкт програмного забезпечення для оцінювання тривалості розробки ПЗ, створеного за методологією Agile

При виконанні робочого проєкту були обрані мова програмування та засоби розробки, розроблена діаграма компонентів, проведено тестування та випробування програмного забезпечення.

4.3.1 Вибір засобів розробки програмного забезпечення для оцінювання тривалості розробки ПЗ

В якості мови програмування була обрана об'єктно-орієнтована мова C #. Синтаксис C # схожий на C. Мова по синтаксису близька до C++ і Java.

C # підтримує інкапсуляцію, спадкування, поліморфізм, перевантаження оператора, статичну типізацію.

Розглянемо переваги C #:

– Підтримка продуктів Microsoft.

– Надає безкоштовно деякі інструменти невеликим компаніям та окремим розробникам, як наприклад: Visual Studio, Windows Server, Parallels Desktop для Mac Pro тощо.

– Типи даних мають фіксований розмір. Це підвищує «переносимість» мови та спрощує програмування.

– Має «збірщика сміття» Це означає, що в більшості випадків він займеться звільненням пам'яті. Вищезазначений загальномовний CLR сам запустить збірник сміття та очистить пам’ять.

– Має велику кількість спеціальних конструкцій, які призначені для розуміння та написання коду. Вони не мають значення при компіляції.

– Низький поріг входження. Синтаксис C# має багато подібності з іншими мовами програмування, що полегшує перехід програмістів на C#. Мова C# часто вважається найбільш зрозумілою і придатною для початківців.

– Надає можливість писати додатки для операційних систем iOS, Android, MacOS і Linux з використанням Xamarin.

– Мова затребувана на ринку праці.

Як і всі мови програмування, C # має свої недоліки:

– Орієнтування на платформу Windows;

– Мова безкоштовна лише для невеликих фірм, індивідуальних програмістів, стартапів та студентів. Ліцензія для великих компаній досить дорога.

C# – потужна та універсальна, яка дозволяє розв'язувати різноманітні задачі, мова досить потужна та універсальна. На C# часто розробляють:

– веб-додатки,

– прикладні програми,

– ігри,

– мобільні програми для Android або iOS,

– програми під Windows.

4.3.2 Розробка діаграми компонентів програмного забезпечення для оцінювання тривалості розробки ПЗ

При розробці проєкту програмного забезпечення необхідно розробити моделі логічного та фізичного рівнів та узгодити їх між собою. Мова UML містить діаграми реалізації, які відносяться до фізичних та включають діаграму компонентів і діаграму розгортання.

Діаграма компонентів визначає особливості фізичного представлення системи. З допомогою діаграми можна представити архітектуру розроблюваної системи, визначити залежності між компонентами програми. В якості компонентів можуть бути бібліотеки, вихідні тексти, веб-сторінки, виконувані файли, файли довідки. Діаграма компонентів може містити компоненти, інтерфейси та зв'язки (асоціації, узагальнення, агрегації і залежності) між ними.

На рисунку 4.8 представлено діаграму компонентів ПЗ для оцінювання тривалості розробки програмного забезпечення за методологією Agile.

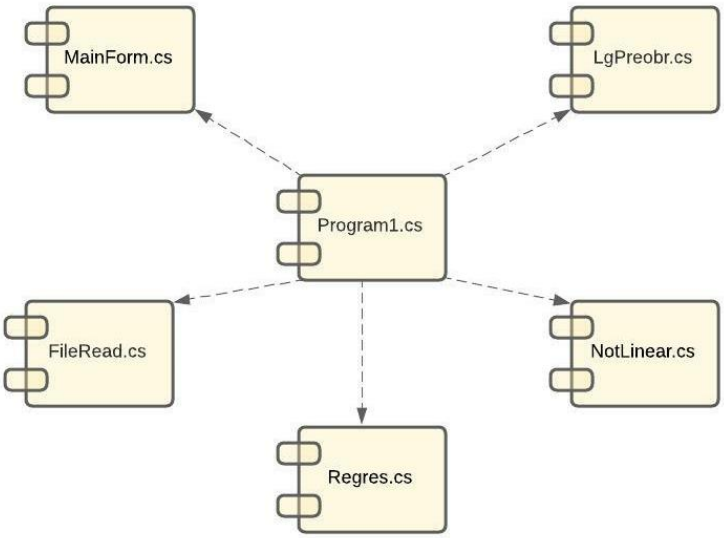

Рисунок 4.8 – Діаграма компонентів ПЗ для оцінювання тривалості розробки програмного забезпечення за методологією Agile

Таким чином з допомогою діаграми компонент можливо отримати візуалізацію загальної організації структури вихідного коду.

4.3.3 Кодування, тестування та випробування ПЗ для оцінювання тривалості розробки програмного забезпечення за методологією Agile

В якості мови програмування для реалізації нелінійної моделі оцінювання тривалості розробки програмного забезпечення за методологією Agile обрано C#. Текст програми наведено в додатку Г.

Мета тестування – оцінити відповідність програмного забезпечення вимогам, які були сформульовані при постановці завдання, виявити помилки проєктування. Якщо достатній рівень відповідності програмного забезпечення був досягнутий, відсутні грубі помилки, то програмне забезпечення приймається до експлуатації.

Тестування будемо здійснювати з використанням Use Case. Таке тестування дає змогу виявити додаткові помилки у додатку, які складно знайти при тестуванні індивідуальних модулів, частин програми окремо одна від одної.

Тест 1 – Тестування можливості вивести на екран інтервали передбачення та довірчі інтервали для лінійної та нелінійної моделей оцінювання тривалості розробки програмного забезпечення за методологією Agile:

– Ввели дані для оцінювання тривалості розробки програмного забезпечення за методологією Agile.

– Натиснули на кнопку «Показати інтервали».

– Довірчі інтервали та інтервали передбачення відобразилися на екрані.

– Замість кнопки «Показати інтервали» з’явилася кнопка «Сховати інтервали».

Результат відповідає очікуваному.

Тест 2 – Тестування можливості визначення таких метрик точності, як коефіцієнт детермінації, середня величина відносної похибки, рівень прогнозування,

– Натиснули на кнопку «Вибрати файл».

– Обрали файл з вихідними даними.

– На екрані відобразилися метрики точності регресійних моделей, що відповідає очікуваному.

Аналогічно виконали тестування інших функцій.

Випробування програмного забезпечення було проведено згідно програми та методики випробувань, що наведена в додатку В, та здійснено з використанням методів чорного ящика, а саме розбиття на класи еквівалентності та аналізу граничних умов. Розглянемо випробування функції оцінювання тривалості розробки програмного забезпечення, її довірчого інтервалу та інтервалу передбачення:

– Ввели дані для оцінювання тривалості розробки програмного забезпечення за методологією Agile.

– Ввели значення трудомісткості для прогнозування.

– Натиснули на кнопку «Передбачення» (рис.4.9).

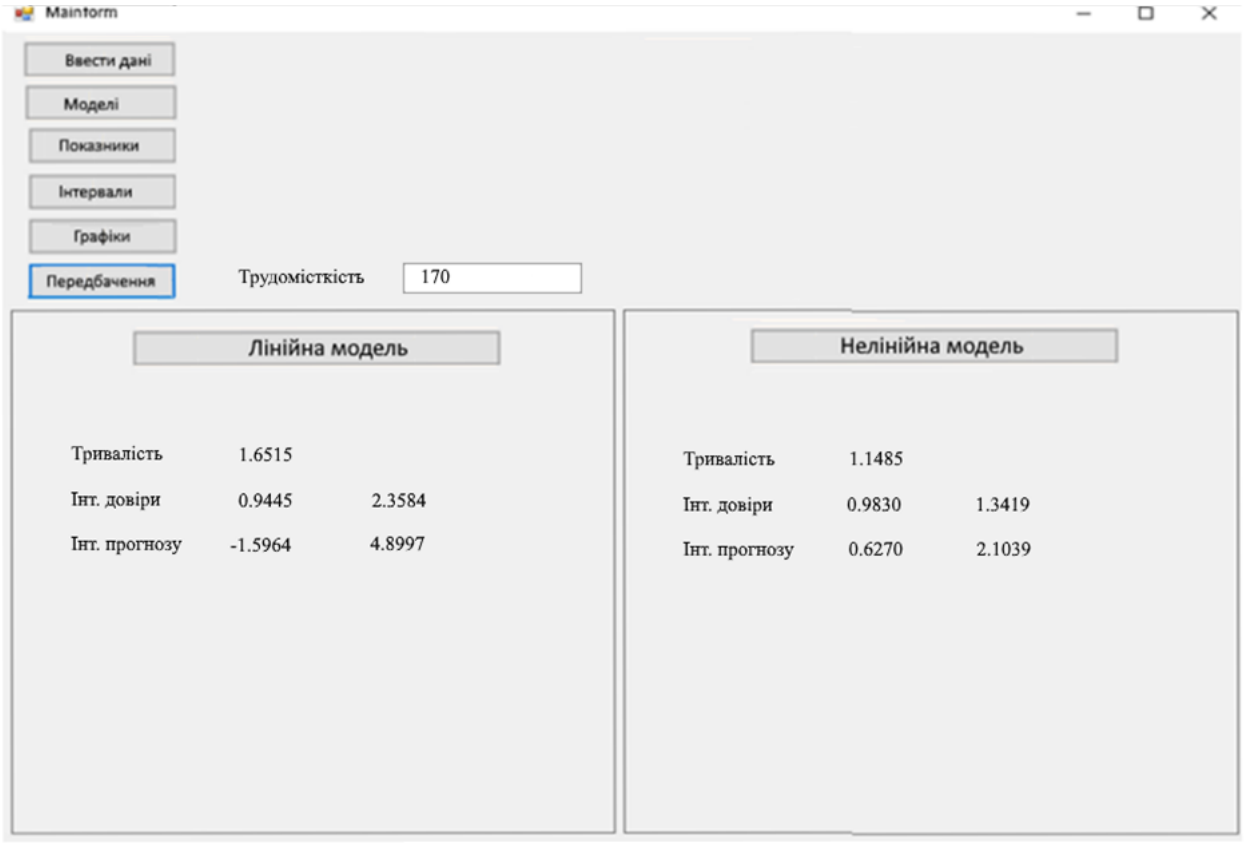

Рисунок 4.9 – Вікно з порахованим прогнозом

Таким чином, отримали прогнозне значення тривалості розробки, інтервал довіри та інтервал прогнозу для лінійної та нелінійної моделей.

Результати випробування показали, що розроблене програмне забезпечення повністю відповідає вимогам, що визначені в технічному завданні.

5 РЕЗУЛЬТАТИ РОЗРОБКИ ПЗ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗА МЕТОДОЛОГІЄЮ AGILE

Мета, яка ставилася на початку виконання кваліфікаційної роботи, була повністю досягнута, тобто підвищена достовірність оцінювання тривалості розробки програмного забезпечення, створеного за методологією Agile, та розроблене програмне забезпечення для її реалізації.

Для досягнення поставленої вирішили наступні завдання:

– проаналізували існуючі моделі оцінювання тривалості розробки ПЗ, порівняли їх;

– обґрунтували необхідність розробки нелінійної регресійної моделі для оцінювання тривалості розробки ПЗ за методологією Agile;

– обрали в якості нормалізуючого перетворення для емпіричних даних, отриманих із попередньо завершених проєктів, десятковий логарифм;

– нормалізували отримані емпіричні дані;

– перевірили нормалізовані дані на викиди;

– побудували модель лінійної регресії, інтервал довіри та прогнозний інтервал для даних з нормалізацією;

– перейшли від лінійної регресійної моделі до нелінійної та побудували модель нелінійної регресії, довірчий інтервал та інтервал прогнозування для вихідних даних;

– побудувати модель лінійної регресії, інтервал довіри та прогнозний інтервал без нормалізації даних в припущенні про нормальність вихідних даних;

– виконали порівняння лінійної моделі з нелінійною;

– виконали оцінювання тривалості розробки нового проєкту;

– розробили ПЗ для оцінювання тривалості розробки програмного забезпечення, створеного за методологією Agile за розробленою методикою.

Розроблене програмне забезпечення відповідає вимогам до функціональних характеристик, вимогам до організації вхідних-вихідних даних вимогам до складу та параметрів технічних засобів технічного завдання.

Також в роботі були розроблені наступні додатки:

– Технічне завдання (додаток А);
– Опис програми (додаток Б);
– Програма та методика випробувань (додаток В);
– Текст програми (додаток Г);
– Інструкція користувача (додаток Д).

6 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВІД РОЗРОБКИ ТА ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

6.1 Вступ

Дана робота присвячена розробці програми для оцінювання тривалості розробки програмного забезпечення за методологією Agile. Програмне забезпечення можна використати в галузі управління проєктами розробки програмного забезпечення.

Створення програмного забезпечення вимагає одноразових витрат на її розробку, придбання необхідних технічних засобів і поточних витрат на функціонування. Економія від функціонування програмного забезпечення визначається з врахуванням витрат на її експлуатацію. Відношення цієї економії до витрат на створення програмного забезпечення характеризує економічну ефективність капітальних вкладень. Економічні показники визначаються по діючим на момент розрахунку оптовим цінам, тарифам і ставкам заробітної плати.

6.2 Розрахунок витрат на створення й експлуатацію програмного забезпечення

Витрати на розробку програми для оцінювання тривалості розробки програмного забезпечення за методологією Agile складаються з витрат: на оплату праці розробника, на амортизацію комп‘ютера , на якому відбувається розробка, на експлуатацію цього комп‘ютера, на засоби розробки, та витрат на виробничі запаси.

Розробка виконується програмістом із місячним окладом 10 000 грн. Додаткова заробітна плата складає 20% від основної. Таким чином, основна і

додаткова заробітна плата програміста складає 12 000 грн. Вартість ноутбука «Dell Inspiron 3582» складає 8 000 грн. (Екран 15.6" TN+film (1366x768) HD, матовий / Intel Pentium Silver N5000 (1.1 – 2.7 ГГц) / RAM 4 ГБ / HDD 1 ТБ / Intel UHD Graphics 605 / DVD+/-RW / Wi-Fi / Bluetooth / вебкамера / Windows 10 Home 64bit / 2.28 кг / чорний). При вартості кіловат-години електроенергії у 1,68 грн. розраховується вартість програми для оцінювання тривалості розробки програмного забезпечення за методологією Agile. Витрати на допоміжні матеріали наведені у табл. 6.1.

Таблиця 6.1 – Витрати на допоміжні матеріали

Вид допоміжних матеріалів	Вартість
Офісний папір	100,00
Заправлення картриджа до принтеру	100,00
Флеш-накопичувач даних	250,00
Разом допоміжні матеріали	450,00

Вартість програмного забезпечення визначимо за формулою:

$$Впз = (Вопт+Всз+ЗВВ+Ве)*Т+Вдм, \quad (6.1)$$

де Впз - вартість програмного забезпечення, грн.; Вопт – витрати на оплату праці (основна та додаткова), грн.; Всз – відрахування на соціальні заходи або єдиний соціальний внесок (36,76% від основної і додаткової заробітної плати), грн.; ЗВВ – загально-виробничі витрати (10% від основної заробітної плати), грн.; Зе – витрати на електроенергію, грн.; Т – тривалість розробки, міс.; Вдм – витрати на допоміжні матеріали, грн.

Витрати на електроенергію за умови споживання 0,5 кВт, тривалості роботи за місяць, дорівнює 22*8=176 годин, що у підсумку складає:

$$Ве =176*1,68*0,5=151,06 \text{ грн.}$$

Витрати на розробку програмного забезпечення наведені у табл. 6.2.

Таблиця 6.2 – Витрати на розробку програмного забезпечення

Стаття витрат	Сума витрат, грн.
Витрати на оплату праці	12000,00
Відрахування на соціальні заходи	4411,20
Загальновиробничі витрати	1000,00
Витрати на допоміжні матеріали	450,00
Витрати на електроенергію	151,06

Відповідно до формули (6.1) вартість програмного забезпечення складає:

Впз=(12000+4411,2+1000+151,06)*2+450=17562,26*2+450=35 574,52 грн.

Амортизаційні відрахування на основні засоби (ноутбук «Dell Inspiron 3582») складають 25% від первісної вартості в рік:

$$Вамор = 8000*0,25 = 2\ 000,00 \text{ грн}$$

У масштабах підприємства річні витрати на матеріали визначаються у розмірі 5% вартості основних засобів і складають:

$$Вм = 8000*0,05 = 400,00 \text{ грн.}$$

Річне споживання електроенергії ноутбуком у годинах визначається в такій спосіб:

$$Чн = 264,5*Тз, \quad (6.2)$$

де Тз – середнє місячне завантаження устаткування (близько 4 годин); 264,5 - середня кількість робочих днів на рік.

Отже річне споживання електроенергії ноутбуком складає:

$$Чн = 264,5*4 = 1058 \text{ годин.}$$

Витрати на споживання електроенергії при 1058 годин роботи ноутбука за рік становить:

$$Ве = 1058*1,68*0,5 = 908,08 \text{ грн.}$$

Експлуатаційні витрати для ноутбука за рік складають:

$$Взр = 2000+400+908,08 = 3\ 308,08 \text{ грн.}$$

Отже за перший рік витрати на створення й експлуатацію програми складають:

Все = 35 574,52+3 308,08 =35 574,52грн.

6.3 Економічна ефективність розробки і впровадження

Основним показником економічної ефективності функціонування програмного забезпечення є зниження трудомісткості розрахункових робіт щодо оцінювання тривалості розробки програмного забезпечення за методологією Agile. До числа основних факторів, що визначають приріст прибутку в зв‘язку з впровадженням програмного забезпечення, відносяться:

– підвищення продуктивності праці;

– вивільнення робочого часу.

Визначимо економічну ефективність, ґрунтуючись на тому, що впровадження програмного забезпечення вивільняє 0,5 працівника (за експертною оцінкою фахівців підприємства).

Оплата праці 0,5 працівника за рік складає:

10000*12*0,5=60 000,00 грн.

Річний економічний ефект визначається за формулою:

$$Е_{рік} = \Delta С_п - Е_н * К, \quad (6.3)$$

де $\Delta С_п$ – вивільнені кошти після впровадження програмного забезпечення = 60 000,00-3 308,08 =56 691,92

$Е_н$ – нормативний коефіцієнт економічної ефективності (дорівнює коефіцієнту амортизації 0,25%);

К – капітальні витрати на впровадження програмного забезпечення 35 574,52 грн.

Річний економічний ефект становить:

$Е_н$ = 56 691,92 - 35 574,52*0,25 = 47 798,29 грн.

Строк окупності капітальних вкладень визначається за наступною формулою:

$$T = \frac{K}{\Delta C_n}, \tag{6.4}$$

Строк окупності становить:

T = 35 574,52/56 691,92 = 0,63 року

Отже строк окупності програми оцінювання тривалості розробки програмного забезпечення за методологією Agile складає приблизно 7,5 місяців.

6.4 Висновки за розділом 6

За отриманим значенням річного економічного ефекту і строку окупності можна зробити висновок, що розробка програми для оцінювання тривалості розробки програмного забезпечення за методологією Agile є економічно вигідною та обґрунтованою.

7 ОХОРОНА ПРАЦІ

7.1 Вступ

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

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

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

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

Охорона праці – один з центральних інститутів трудового права. Він має виключно практичне значення. Недодержання вимог охорони праці створює небезпеку для здоров'я і життя працівників. У свою чергу і ті, кого законодавець називає власником або уповноваженим ним органом, несуть

сувору, у тому числі й кримінальну відповідальність за порушення правил охорони праці [26].

7.2 Аналіз шкідливих та небезпечних факторів у офісі комп'ютерної фірми

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

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

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

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

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

За вимогами санітарних норм освітлення повинно бути достатньо яскравим, рівномірним, не засліплювати очі та не створювати відблисків на

робочій поверхні. За спектральним складом освітлення має наближатися до сонячного світла. Гігієнічними нормами вимагається максимально використовувати природне освітлення, оскільки денне світло краще сприймається органами зору. Штучне освітлення у виробничих та адміністративно-громадських приміщеннях має здійснюватись системою загального рівномірного освітлення, допускають застосування системи комбінованого освітлення. Значення освітленості на поверхні робочого столу в зоні розміщення документів має становити 300-500лк. Як джерела світла для штучного освітлення мають застосовувати переважно люмінесцентні лампи типу ЛБ [27].

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

Гігієнічними нормативами (СН 3222-85, ГОСТ 12.1.003-85, ГР 2411-81) встановлені допустимі рівні звуку різних частот для робітників певних професій. У таблиці 7.1 наведені нормативи для людей, що працюють з ЕОМ.

Виходячи з даних норм, при проведенні комплексу заходів зі зменшення шуму усувається не будь-яка віброакустична активність обладнання, а тільки та, яка перевищує встановлений граничний рівень та шкідливо впливає на організм людини. Усунення промислового шуму в першу чергу відбувається шляхом вдосконалення технологічних процесів та обладнання. При цьому в першу чергу усуваються найбільш сильні джерела шуму. Також необхідно знешумлювати усе обладнання, яке цього потребує, оскільки знешумлення окремих одиниць при наявності великої кількості обладнання, що продовжує випромінювати шум, недоцільно.

Таблиця 7.1 – Допустимі рівні шуму

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

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

терморегуляцію, зменшення тепловиділення організмом. Фізіологічно оптимальною є відносна вологість в межах 40..60%.

Сукупність описаних показників стану навколишнього середовища називають мікрокліматом. Для приміщень з ЕОМ встановлені норми мікроклімату приведені у таблиці 7.2.

Таблиця 7.2 – Норми мікроклімату для приміщень з ЕОМ

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

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

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

Людина відчуває дратівну дію змінного струму промислової частоти силою 0,6 -1,5 мА і постійного струму 5-7 мА. Ці струми не представляють серйозної небезпеки для організму людини, тому що при такій силі струму можливе самостійне звільнення людини від контакту зі струмоведучими частинами. Для перемінного струму промислової частоти сила невідпускаючого струму знаходиться в межах 6-20 мА. Постійний струм не

викликає невідпускаючого ефекту, але приводить до сильних болючих відчуттів, сила такого струму 15-80 мА і більше [26].

7.3 Розрахунок системи пожежогасіння у офісі комп'ютерної фірми

Таблиця груп приміщень наведена в таблиці 7.3.

Таблиця 7.3 – Групи приміщень

Група	Приміщення	Пожежне навантаження
1	Приміщення книгосховищ, ЕОМ, магазинів, будівель управлінь, готелів, лікарень	до 200 Мдж/м^2
2	Приміщення з використанням рідин, які легко спалахують (ЛСР) і горять (ГР); деревообробні, швейні та інші приміщення; підприємства з обслуговування автомобілів	200...2000 Мдж/м^2
3	Приміщення гумотехнічного виробництва	200...2000 Мдж/м^2
4	Приміщення для виробництва, обробки, переробки різних матеріалів з використанням ЛСР і ГР	>2000 Мдж/м^2
5	Склади негорючих матеріалів в упаковці, що горить	>2000 Мдж/м^2
6	Склади твердих матеріалів, що горять	>2000 Мдж/м^2
7	Склади лаків, фарб, ЛСР, ГР, пластмас та ін.	>2000 Мдж/м^2

Група приміщення: 1

Тип зрошувача: вода

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

1. Знайдемо групу приміщення (табл. 7.3) згідно з пожежним навантаженням, які забезпечуються автоматичними установками пожежогасіння (ДВН В.2.5-13-98).

За таблицею 7.3 приміщення офісу комп'ютерної фірми належить до 1 групи.

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

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

3. Знаходимо площу приміщення:

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

4. Знаходимо необхідну кількість зрошувачів:

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

5. Розміщуємо зрошувачі на плані приміщення.

6. Знаходимо необхідну інтенсивність води в трубопроводі:

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

7. Знаходимо інтенсивність води крізь один спринклер:

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

8. Знаходимо необхідний об’єм води:

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

7.4 Розробка заходів щодо зменшення впливу шкідливих та небезпечних факторів

Сьогодні фахівці в області ергономіки вже зрозуміли, що не можна знайти ідеальне положення, у якому можна перебувати і працювати протягом усього робочого дня. Для більшості людей комфортабельним робочим місцем повинне бути таке, котре можна пристосувати не менш чим для двох позицій, при цьому положення крісла, монітора повинні щораз відповідати

виконуваній роботі і звичкам. Багато хто вважають, що для роботи на комп'ютері більше підходять вертикальне і злегка похиле положення. Можливо, щоб крісло було злегка нахилене вперед.

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

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

Дисплей. Положення тіла звичайно відповідає напрямку погляду; дисплеї, розташовані занадто низько чи під неправильним кутом, є основними причинами сутулості. Відстань від дисплея до очей може варіюватися в залежності від характеру виконуваної роботи – 40-70 см. При роботі з текстом відстань від екрана до очей повинна лише небагато перевищувати відстань між книгою й очима і складати 40-45 см. Необхідно давати відпочинок очам і час від часу їх просто закривати. Не можна допускати, щоб очі увесь час були сфокусовані на одній відстані, працюючи з текстом, рекомендується установлювати великий шрифт. Дуже важливо, щоб екран монітора не мерехтів. Частота регенерації зображення повинна бути не менш 72 Гц, тому що при більш низькій частоті помітне мерехтіння, хоча люди з особливо чуттєвим зором можуть помітити його і при більш високій частоті. Їм доведеться підібрати графічну плату і монітор з частотою регенерації 85 Гц і вище.

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

дисплея повинний бути перпендикулярний шибці – цим виключаються відблиски на екрані.

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

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

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

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

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

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

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

- Сильно зжати пальці, а потім розпрямити їх.

Виконання приведених рекомендацій дозволяє скоротити вплив шкідливих виробничих факторів на організм людини [26].

8 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

8.1 Вступ

Навколишнє (природне) середовище – це сукупність об’єктів та умов природи, в яких проходить діяльність будь-якого суб’єкта, що відрізняються від інших складових частин навколишнього середовища властивостями самопідтримки та саморегуляції без впливу людини.

«Охорона навколишнього середовища» (ОНС) – це система законодавчих актів, нормативних документів, інструкцій, стандартів, ДСТ, що регламентують чи відображають вимоги до охорони природи та навколишнього середовища, при будівництві, реконструкції та експлуатації промислових об’єктів, будинків і споруджень, у тому числі житло-цивільного невиробничого призначення.

«Охорона навколишнього середовища» полягає в здійснені комплексних технічних рішень по:

- охороні атмосферного повітря від забруднення;
- охороні поверхневих і підземних вод від забруднення, виснаження;
- відновленню (рекультивація) земельних ділянок;
- використання родючого шару ґрунту, охороні надр і тваринного світу.

Наука і техніка початку третього тисячоліття розвивається в темпах геометричної прогресії, не є виключенням і промисловість як одна із самих (якщо не самої) масштабних сфер діяльності людини. Подібного роду тенденція поширилась по усьому світі і вже захопила, що розвиваються, у минулому слаборозвинені країни. У зв’язку з не бездоганністю технологічних процесів на даному етапі неминуче негативний вплив промисловості на навколишнє середовище, промислових відходів як компонента даного впливу. Щорічно в усьому світі й у нашій країні мільярди тонн твердих, рідких, газоподібних відходів надходить і біосферу, завдаючи тим самим непоправної

втрати як живій, так і неживій природі.

Зростаючі темпи зміни середовища проживання приводить до порушення взаємозв'язку між ними і людиною, зниженню адаптаційних можливостей організму. Середовище проживання може містити такі речовини, з якими організм в ході еволюції не зштовхувався і тому не має відповідних аналізаторних систем, що сигналізують про їхню залежність. У зв'язку з цим оцінити стан здоров'я людини, зрозуміти характер патології у відриві від аналізу змін, що відбуваються у навколишньому середовищі неможливо.

Велике значення тому має організація інформаційної системи «здоров'я населення – навколишнє середовище», дані для якої можуть збиратися через державну статистичну звітність. Задача державної інформаційної системи полягає у зборі даних про забруднення навколишнього середовища, стан здоров'я населення.

8.2 Аналіз існуючої проблеми необхідності охорони навколишнього середовища

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

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

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

Завданням законодавства про охорону навколишнього природного середовища є регулювання відносин у галузі охорони, використання і відтворення природних ресурсів, забезпечення екологічної безпеки, запобігання і ліквідації негативного впливу господарської та іншої діяльності на навколишнє природне середовище, збереження природних ресурсів, генетичного фонду живої природи, ландшафтів та інших природних комплексів, унікальних територій та природних об'єктів, пов'язаних з історико-культурною спадщиною.

Основними принципами охорони навколишнього природного середовища є:

а) пріоритетність вимог екологічної безпеки, обов'язковість додержання екологічних стандартів, нормативів та лімітів використання природних ресурсів при здійсненні господарської, управлінської та іншої діяльності;

б) гарантування екологічно безпечного середовища для життя і здоров'я людей;

в) запобіжний характер заходів щодо охорони навколишнього природного середовища;

г) екологізація матеріального виробництва на основі комплексності рішень у питаннях охорони навколишнього природного середовища, використання та відтворення відновлюваних природних ресурсів, широкого впровадження новітніх технологій;

д) збереження просторової та видової різноманітності і цілісності природних об'єктів і комплексів;

е) науково обгрунтоване узгодження екологічних, економічних та соціальних інтересів суспільства на основі поєднання міждисциплінарних знань екологічних, соціальних, природничих і

технічних наук та прогнозування стану навколишнього природного середовища;

є) обов'язковість надання висновків державної екологічної експертизи;

ж) гласність і демократизм при прийнятті рішень, реалізація яких впливає на стан навколишнього природного середовища, формування у населення екологічного світогляду;

з) науково обгрунтоване нормування впливу господарської та іншої діяльності на навколишнє природне середовище;

и) безоплатність загального та платність спеціального використання природних ресурсів для господарської діяльності;

і) компенсація шкоди, заподіяної порушенням законодавства про охорону навколишнього природного середовища;

ї) вирішення питань охорони навколишнього природного середовища та використання природних ресурсів з урахуванням ступеня антропогенної зміненості територій, сукупної дії факторів, що негативно впливають на екологічну обстановку;

й) поєднання заходів стимулювання і відповідальності у справі охорони навколишнього природного середовища;

к) вирішення проблем охорони навколишнього природного середовища на основі широкого міждержавного співробітництва;

л) встановлення екологічного податку, збору за спеціальне використання води, збору за спеціальне використання лісових ресурсів, плати за користування надрами відповідно до Податкового кодексу України.

8.3 Забруднення навколишнього середовища співробітниками офісу комп'ютерної фірми

В процесі діяльності офісу комп'ютерної фірми утворюються різні тверді побутові відходи. До них можна віднести папір, картонні коробки, різні пакети та т.і., в цілому офісні відходи.

Проблема твердих побутових відходів характерна не тільки для цього закладу, з нею стикаються майже всі установи та заклади, які займаються подібною справою. Ця проблема є не тільки проблемою відокремлених підприємств, установ та закладів, але й всього міста та держави в цілому.

Проблема твердих побутових відходів (ТПВ) в Україні конкурує з показниками й аспектами цієї проблеми у світовому масштабі.

При сильній зволоженості та високій температурі повітря починається процес анаеробного розкладання ТПВ за рахунок розвитку анаеробних бактерій.

Несанкціоноване, стихійне складування ТПВ без урахування вимог і прийомів екологічної біотехнології викликає виділення шкідливих хімічних (сірководень, індол, скатол і т.п.) і біохімічних компонентів, які поширюючись забруднюють ґрунтовий шар, попадають у ґрунтові води, а потім у відкриті водойми. Особливо всі ці несанкціоновані смітники ТПВ, що містять харчові відходи небезпечні в спекотну та суху пору року при t>+25°С, коли дуже підсилюється розвиток всіх видів мікро- і макрофлори, мікро- і макрофауни, й що цілком природно, йде інтенсивна ферментація всіх харчових відходів і відходів природних полімерних матеріалів. При такій температурі t>+25°С створюються умови для інтенсивного розвитку й поширення найнебезпечніших інфекційних захворювань – холери й чуми. От чому й небезпечні стихійні смітники ТПВ не тільки для природного середовища, але й для людини.

У даний момент найпоширеніший спосіб знищення ТПВ – це полігони. Однак, цей простий спосіб супроводжують наступні проблеми:

- надмірно швидке переповнення існуючих полігонів через великий обсяг і малу щільність розташовуваних відходів. Без попереднього ущільнення середня щільність ТПВ становить 200-220 кг/м3, що досягає всього лише 450-500 кг/м3 після ущільнення з використанням сміттєвозів;

- негативні фактори для навколишнього середовища: зараження підземних вод сміттєвими продуктами, виділення неприємного запаху, розкид відходів вітром, мимовільне загоряння полігонів, безконтрольне утворення метану й неестетичний вид є тільки частиною проблем, що турбують екологів і визивають серйозні заперечення з боку місцевої влади;

- відсутність площ, придатних для розміщення полігонів на зручній відстані від великих міст. Розширення міст витісняє полігони на усе більш далеку відстань. Даний фактор у сполученні з ростом цін на землю збільшує вартість транспортування ТПВ;

- неможливість усунення полігонів. Незважаючи на використання найсучасніших технологій, наше суспільство завжди буде мати потребу в їхньому використанні для знищення не перероблених фракцій: золи, металобрухту, будівельного сміття.

8.4 Розробка заходів щодо зменшення забруднення

Збір відходів часто є найбільш дорогим компонентом усього процесу утилізації. Тому правильна організація збору відходів може заощадити значну кількість грошових витрат. Існуюча в Україні система збору ТПВ повинна залишатися стандартизованою з погляду економічності. У той же час додаткове планування необхідно для того, щоб вирішити нові проблеми (наприклад, відходи комерційних кіосків, на збір яких часто не вистачає ресурсів).

У густонаселених територіях нерідко доводиться транспортувати відходи на великі відстані. Рішенням у цьому випадку може бути станція

тимчасового зберігання відходів, від якої сміття може вивозитися великими по вантажопідйомності машинами або по залізниці. При цьому треба відзначити, що станції проміжного зберігання являють собою об'єкти підвищеної екологічної небезпеки й можуть при неправильному розташуванні й експлуатації викликати не менше дорікань місцевих жителів і громадських організацій, ніж смітники.

Вторинна переробка

Досить багато компонентів ТПВ можуть бути перероблені в корисні продукти. Скло звичайно перероблюють шляхом здрібнювання й переплавлення (бажано, щоб вихідне скло було одного кольору). Скляний бій низької якості після здрібнювання використовується як наповнювач для будівельних матеріалів (наприклад, т.зв. «глассфальт»). У багатьох містах існують підприємства по відмиванню й повторному використанню скляного посуду. Така ж, безумовно, позитивна практика існує, наприклад, у Данії.

Сталеві й алюмінієві банки переплавляються з метою одержання відповідного металу. При цьому виплавка алюмінію з баночок для прохолодних напоїв вимагає тільки 5% від енергії, необхідної для виготовлення тієї ж кількості алюмінію з руди, і є одним з найбільш вигідних видів «ресайклінга».

Паперові відходи різного типу вже багато десятків років застосовують поряд зі звичайною целюлозою для виготовлення пульпи – сировини для паперу. Зі змішаних або низькоякісних паперових відходів можна виготовляти туалетний або обгортковий папір і картон. На жаль, в Україні тільки в невеликих масштабах присутня технологія виробництва високоякісного паперу з високоякісних відходів (обрізків друкарень, використаного паперу для ксероксів і лазерних принтерів і т.п.). Паперові відходи можуть також використовуватися в будівництві для виробництва теплоізоляційних матеріалів і в сільському господарстві – замість соломи на фермах.

Пластик. Переробка пластику досить дорогий та складний процес. З деяких видів пластику можна одержувати високоякісний пластик тих же

властивостей, інші види пластику (наприклад ПВХ) після переробки можуть бути використані тільки як будівельні матеріали.

Компостування

Компостування – це технологія переробки відходів, заснована на їхньому природному біорозкладанні. Найбільш широко компостування застосовується для переробки відходів органічного, насамперед рослинного походження, таких як листи, вітки й скошена трава. Існують технології компостування харчових відходів та ТПВ.

У Україні компостування за допомогою компостних ям часто застосовується населенням в індивідуальних будинках або на садових ділянках. У той же час процес компостування може бути централізований і проводитися на спеціальних площадках. Існує кілька технологій компостування, що розрізняються за вартістю й складністю. Більше прості й дешеві технології вимагають більше місця й процес компостування займає більше часу.

Кінцевим продуктом компостування є компост, що може знайти різні застосування в міському й сільському господарстві.

Компостування, застосовуване в Україні на механізованих сміттєпереробних заводах, наприклад, у Львові, представляє собою процес зброджування в біореакторах усього обсягу ТПВ, а не тільки його органічної складової. Хоча характеристики кінцевого продукту можуть бути значно поліпшені шляхом добування з відходів металу, пластику й т.п., все ж таки він представляє собою досить небезпечний продукт і знаходить дуже обмежене застосування (на Заході такий «компост» застосовують тільки для покриття смітників).

Таким чином, проблема побутових відходів є досить важливою для нашого суспільства. Існують розроблені методи для високотехнологічної утилізації сміття, але для того щоб вони були ефективними, треба, щоб кожна людина перейнялася відповідальністю за екологічний стан навколишнього середовища.

ВИСНОВКИ

Кваліфікаційна робота присвячена удосконаленню регресійної моделі для оцінювання тривалості розробки програмного забезпечення за методологією Agile та створенню програми для її реалізації Під час виконання роботи були поставленні задачі, які було виконано у повному обсязі:

– проаналізували існуючі моделі оцінювання тривалості розробки ПЗ, порівняли їх;

– обґрунтували необхідність розробки нелінійної регресійної моделі для оцінювання тривалості розробки ПЗ за методологією Agile;

– обрали в якості нормалізуючого перетворення для емпіричних даних, отриманих із попередньо завершених проєктів, десятковий логарифм;

– нормалізували отримані емпіричні дані;

– перевірили нормалізовані дані на викиди;

– побудували модель лінійної регресії, інтервал довіри та прогнозний інтервал для даних з нормалізацією;

– перейшли від лінійної регресійної моделі до нелінійної та побудували модель нелінійної регресії, довірчий інтервал та інтервал прогнозування для вихідних даних;

– побудували модель лінійної регресії, інтервал довіри та прогнозний інтервал без нормалізації даних в припущенні про нормальність вихідних даних;

– виконали порівняння лінійної моделі з нелінійною;

– оцінили тривалість розробки нового проєкту за методологією Agile;

– розробили ПЗ для оцінювання тривалості розробки програмного забезпечення, створеного за методологією Agile за розробленою методикою.

ПЗ для оцінювання тривалості розробки програмного забезпечення за методологією Agile, яке розроблено в рамках кваліфікаційної роботи, дозволить автоматизувати та скоротити час відповідних розрахунків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Методика оценки длительности ИТ-проектов для анализа их инвестиционной привлекательности / Л.Л. Куликова, В.Ю. Швакин // Программные продукты и системы // Международный научно-практический журнал. URL: https://cyberleninka.ru/article/n/metodika-otsenki-dlitelnosti-it-proektov-dlya-analiza-ih-investitsionnoy-privlekatelnosti (дата звернення: 28.11.2021).

2. Оценка стоимости разработки программного продукта: обзор / Е.В. Ваганова, А.А. Земцов, С.Л. Миньков // Проблемы учета и финансов. – Томск: ТГУ, 2016. – №1 (21). – С.58-62.

3. Модели, методы и средства оценки стоимости программного обеспечения / Н.А. Сидоров, Д.В. Баценко, Ю.Н. Василенко, Ю.В. Щебетин // Проблеми програмування. – 2006. – N 2-3. – С. 290-298. URL: http://dspace.nbuv.gov.ua/bitstream/handle/123456789/1541/36-Sidorov.pdf (дата звернення: 28.11.2021).

4. Parkinson S.N. Parkinson's Law and Other Studies in Administration. URL: http://sas2.elte.hu/tg/ptorv/Parkinson-s-Law.pdf (дата звернення: 28.11.2021).

5. Boehm, B.W. Software engineering economics [Текст] / B.W. Boehm. – Prentice-Hall, 1981. – 320 p.

6. Shepperd M., C. Schofield Estimating software project effort using analogy. – IEEE Trans Software Eng. – 1997. – P. 736

7. Порівняння ймовірнісних моделей тривалості робіт в програмних проектах / С.Б. Приходько, А.В. Пухалевич // Восточно-Европейский журнал передовых технологий // научный журнал. – Харьков: Технологический центр, 2012. – №1/13 (55). – С.39-41.

8. Розробка нелінійних регресійних моделей тривалості програмних проектів на основі перетворення Джонсона / С.Б. Приходько, А.В. Пухалевич // Зб. наук. праць НУК. – Миколаїв: НУК, 2014. – № 3. – С.76-80.

9. Confidence interval estimation of PC software project duration regression based on Johnson transformation. Radio-electronic and computer systems. S B Prykhodko, A V Pukhalevich. 2014. 104-107.

10. Вибір нормалізуючого перетворення для інтервального оцінювання часу виконання проектів підготовки боксерів-жінок до змагань / С.Б. Приходько, Н.В. Князь // Зб. наук. праць НУК. – Миколаїв: НУК, 2014. – № 5 (455). – С.87-91.

11. Latanska L.O. Preprocessing of empirical data for building a nonlinear regression model for estimating the duration of agile methodology development it-projects / L.O. Latanska, L.M. Makarova, V.O. Kairov // «Free and Open Source Software»: Матеріали XIII-ої Міжнародної науково-практичної конференції (16-18 листопада 2021 р.). – Харків: Харківський національний університет будівництва та архітектури, 2021. – с.9-10.

12. Agile-манифест разработки программного обеспечения. URL: https://agilemanifesto.org/iso/ru/manifesto.html (дата звернення: 30.11.2021).

13. Анализ Agile. Мифы и действительность. URL: https://habr.com/ru/post/436178/ (дата звернення: 30.11.2021).

14. Johnson, Kim. Software cost estimation – Metrics and models [Текст] / Johnson Kim. – University of Calgary, 2001. – 115 p.

15. Boehm, B.W. The COCOMO 2.0 Software Cost Estimation Model [Текст] / B.W. Boehm. – American Programmer, 2000. – 586 p.

16. Ивченко Г.И. Ивченко Григорий Иванович, Медведев Юрий Иванович Математическая статистика: Учебник [Текст] / Г.И Ивченко, Ю.И. Медведев. – М.: Книжный дом «ЛИБРОКОМ», 2014. – 352 с.

17. Магнус, Я.Р. Эконометрика. Начальный курс: Учеб. – 9-е изд., испр [Текст] / Я.Р. Магнус, П.К. Катышев, А.А. Пересецкий. – М.: Издательский дом «Дело» РАНХиГС, 2021. – 504 с.

18. Кожевніков, І.Г. Емпіричні моделі оцінки вартості розробки програмного забезпечення [Текст] / І.Г. Кожевніков // Економічний простір. – 2014. – №83. – С.195-208.

19. International Software Benchmarking Standards Group. URL: https://www.isbsg.org/ (дата звернення: 30.11.2021).

20. Нелінійна регресія. вибір і порівняння нелінійних регресійних моделей. URL: http://dn.khnu.km.ua/dn/ k_default.aspx?M=k1314&T=02&lng=1&st=0 (дата звернення: 30.11.2021).

21. Регресійний аналіз. URL: https://pidruchniki.com/17280924/ekonomika/regresiyniy_analiz (дата звернення: 30.11.2021).

22. Chatterjee, Samprit. Handbook of Regression Analysis [Text] / Samprit Chatterjee, Jeffrey S. Somonoff. Wiley, 2012. – 240 p.

23. Приходько С.Б. Методичні вказівки та завдання до виконання лабораторних робіт з дисципліни «Обробка експериментальних даних на комп'ютері» / С.Б. Приходько Л.М. Макарова, К.С. Приходько. – Миколаїв: НУК, 2018. – 76с.

24. Регрессионный анализ. URL: HTTP://WWW.MACHINELEARNING.RU/WIKI/INDEX.PHP?TITLE=%D0%A0%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%81%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7 (дата звернення: 25.11.2021).

25. Охрана труда при работе с комп'ютером. URL: https://www.sop.com.ua/article/ru/242-qqq-16-m9-05-09-2016-ohrana-truda-pri-rabote-s-kompyuterom (дата звернення: 30.11.2021).

26. Как снизить шум в офисе: практические рекомендации. URL**:** https://skomplekt.com/kak-snizit-shum-v-ofise/ (дата звернення: 25.11.2021).

27. Охорона праці. Освітлення виробничих приміщень. URL: http://www.ztec.com.ua/ztec/e-lib/%D0%9E%D1%85%D0%BE%D1%80%D0%BE%D0%BD%D0%B0%20%D0%BF%D1%80%D0%B0%D1%86%D1%96/%D0%A2%D0%B5%D0%BC%D0%B0%209%20%D0%9E%D1%81%D0%B2%D1%96%D1%82%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F.pdf (дата звернення: 25.11.2021).

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ПЗ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗА МЕТОДОЛОГІЄЮ AGILE

Вступ

Назва роботи – «Нелінійна регресійна модель для оцінювання тривалості проєктів зі створення програмного забезпечення за методологією Agile та розробка програми для її реалізації». Програмне забезпечення являється програмою, яка буде надавати змогу користувачу провести оцінювання тривалості розробки програмного забезпечення за методологією Agile на основі даних про трудомісткість розробки.

1. Підстави для розробки

Документом, на підставі якого ведеться розробка програмного продукту, є завдання на кваліфікаційну роботу «Нелінійна регресійна модель для оцінювання тривалості проєктів зі створення програмного забезпечення за методологією Agile та розробка програми для її реалізації».

2. Призначення розробки

2.1. Функціональне призначення

Даний програмний продукт дозволить автоматизувати та скоротити час відповідних розрахунків.

2.2. Експлуатаційне призначення

Програмний продукт буде використовуватись для оцінювання тривалості розробки програмного забезпечення за методологією Agile і побудови інтервала довіри та інтервала передбачення на основі десяткового логарифму.

3. Вимоги до програми

3.1. Вимоги до функціональних характеристик

3.1.1. Вимоги до складу функцій, що виконуються

Програма повинна забезпечувати можливість виконання наведених нижче функцій:

1) Внесення даних про програмні проєкти;

2) Нормалізація вихідних даних (десятковий логарифм);

3) Побудова для нормалізованих даних лінійної регресії, довірчого інтервалу та інтервалу передбачення;

4) Побудова нелінійної регресії, довірчого інтервалу та інтервалу передбачення для неї через зворотне перетворення;

5) Розрахунок характеристик якості;

6) Оцінювання тривалості розробки програмного забезпечення, її довірчого інтервалу та інтервалу передбачення за допомогою нелінійної регресії.

3.1.2 Вимоги до організації вхідних та вихідних даних

Вхідні дані мають бути організовані у текстових файлах. Внесення вхідних даних виконується шляхом вибору шляху до відповідного текстового файлу у спеціальному вікні.

Вихідні дані (довірчі інтервали, інтервали передбачення та метрики якості) повинні виводитися в відповідних місцях на створеній програмою формі.

3.2 Вимоги до надійності

3.2.1 Вимоги до забезпечення надійності функціонування програми

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

3.2.2 Відмова виконання програмного продукту через некоректні дії користувача

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

3.4 Умови експлуатації

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

3.5 Вимоги до інформаційної та програмної сумісності

Системні програмні засоби (не вільно поширювані), використовувані програмою, повинні бути представлені ліцензійною версією ОС Windows.

3.6 Вимоги до захисту інформації та програм

Вимоги до захисту інформації та програм відсутні.

3.7 Вимоги до маркування та упаковки

3.7.1 Вимоги до маркування

Вимоги до маркування відсутні.

3.7.2 Вимоги до упаковки

Вимоги до упаковки відсутні.

3.8 Вимоги до транспортування та зберігання

Вимоги до транспортування програми відсутні.

4 Вимоги до програмної документації

Склад програмної документації:

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

5 Техніко-економічні показники

Економічна ефективність впровадження програмного забезпечення розрахована у розділі 6. Згідно з отриманими результатами впровадження даного ПЗ є доцільним, а термін його окупності становить приблизно 7,5 місяці.

6 Стадії та етапи розробки програмного забезпечення

Стадії та етапи розробки програми для оцінювання тривалості розробки програмного забезпечення за методологією Agile зазначені в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки програмного забезпечення

Стадії розробки	Етапи робіт	Вміст робіт	Контрольні точки
1	2	3	4
Технічне завдання	Обумовлення необхідності розробки	Постановка задачі. Збір початкових матеріалів.	З 07.09.2021 до 13.09.2021
	Розробка та затвердження технічного завдання	Визначення вимог до ПЗ. Визначення вимог до ПЗ. Обговорення та затвердження технічного завдання	З 14.09.2021 до 20.09.2021
Ескізний проєкт	Розробка ескізного проєкту	Попередня розробка структури вхідних та вихідних даних, уточнення методів рішення завдань, загальний опис алгоритму	З 21.09.2021 до 11.10.2021
	Затвердження ескізного проєкту	Розробка пояснювальної записки. Узгодження і затвердження ескізного проєкту	З 12.10.2021 до 14.10.2021

Продовження таблиці А.1

1	2	3	4
Технічний проєкт	Розробка технічного проєкту	Уточнення структури вхідних та вихідних даних, розробка алгоритму рішення задачі, розробка структури ПЗ	З 15.10.2021 до 01.11.2021
	Затвердження технічного проєкту	Розробка пояснювальної записки. Узгодження і затвердження технічного проєкту	З 02.11.2021 до 08.11.2021
Робочий проєкт	Розробка ПЗ	Програмування та налагодження програми	З 09.11.2021 до 19.11.2021
	Розробка програмної документації	Розробка програмної документації відповідно до вимог ГОСТ 19.101-77	З 20.11.2021 до 01.12.2021
	Випробування програми	Розробка, узгодження та затвердження програми і методики випробувань, коригування програми і програмної документації за результатами випробувань	З 02.12.2021 до 04.12.2021

7 Порядок контролю та приймання

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

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

ДОДАТОК Б

ОПИС ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗА МЕТОДОЛОГІЄЮ AGILE

1 Загальні відомості

1.1 Позначення і найменування програми

Найменування: ПЗ оцінювання тривалості розробки програмного забезпечення за методологією Agile «T_Agile».

Позначення: «T_Agile».

1.2 Програмне забезпечення, необхідне для функціонування програми

Програмне забезпечення «T_Agile» є крос-платформним і основною необхідною вимогою для функціонування виробу є наявність .NET Framework (4.5 і вище версії).

1.3 Мови програмування

ПЗ «T_Agile» реалізовано на мові високого рівня C#. Модулі, реалізовані на інших мовах програмування, не використовуються.

2 Функціональне призначення

2.1 Класи вирішуваних завдань і призначення програми

ПЗ «T_Agile» повинно забезпечувати можливість виконання нижче перерахованих функцій:

1) Внесення даних про програмні проєкти;

2) Нормалізація вихідних даних (десятковий логарифм);

3) Побудова для нормалізованих даних лінійної регресії, довірчого інтервалу та інтервалу передбачення;

4) Побудова нелінійної регресії, довірчого інтервалу та інтервалу передбачення для неї через зворотне перетворення;

5) Розрахунок характеристик якості;

6) Оцінювання тривалості розробки програмного забезпечення, її довірчого інтервалу та інтервалу передбачення за допомогою нелінійної регресії.

2.2 Функціональні обмеження

Файли повинні бути формату EXE.

3 Використовувані технічні засоби

ПЗ «T_Agile» призначене для використання на персональних комп'ютерах, що працюють під управлінням наступних операційних систем: ОС MS Windows XP/7/8/8.1/10

Для ПЗ «T_Agile» пред'являються такі апаратні вимоги до ПЕОМ:

- оперативна пам'ять 512 МБ;
- дисковий простір - не менше 512 МБ вільного місця на диску;
- роздільна здатність екрану - 1280 x 1024 пікселів;
- клавіатура 101/102-х клавішна рус / лат;

Швидкість роботи ПЗ «T_Agile» на конкретному комп'ютері залежить також від характеристик окремих його комплектуючих (процесора, оперативної пам'яті і ін.).

4 Виклик і завантаження

Запуск ПЗ «T_Agile» в графічному режимі здійснюється двома способами:

1) за допомогою файлу запуску: T_Agile.exe;

2) з використанням командного рядку: T_Agile.exe.

5 Вхідні і вихідні дані

Вхідні дані програми(тривалість та трудомісткість) повинні знаходитись в окремому текстовому файлі у два стовбці. Вихідні дані програми мають відображатися на екрані користувача.

ДОДАТОК В

ПРОГРАМА І МЕТОДИКА ВИПРОБУВАНЬ ПЗ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗА МЕТОДОЛОГІЄЮ AGILE

1 Об’єкт випробувань

Об’єктом випробувань є ПЗ для оцінювання тривалості розробки програмного забезпечення за методологією Agile

2 Мета випробувань

Перевірка функціональних можливостей програмного забезпечення на відповідність вимогам технічного завдання.

3 Вимоги до програми

В ході тестування програмного продукту необхідно перевірити, що розроблене ПЗ реалізує наступні функції:

1) Внесення даних про програмні проєкти;

2) Нормалізація вихідних даних (десятковий логарифм);

3) Побудова для нормалізованих даних лінійної регресії, довірчого інтервалу та інтервалу передбачення;

4) Побудова нелінійної регресії, довірчого інтервалу та інтервалу передбачення для неї через зворотне перетворення;

5) Розрахунок характеристик якості;

6) Оцінювання тривалості розробки програмного забезпечення, її довірчого інтервалу та інтервалу передбачення за допомогою нелінійної регресії.

4 Вимоги до програмної документації

Склад програмної документації, що надається на випробування:

– технічне завдання;

– текст програми;

– опис програми;

– програма і методика випробувань;

– інструкція користувача.

5 Засоби і порядок випробувань

Випробування проводилось на платформі Windows 7 32bit.

Порядок випробувань:

– встановити і запустити програму;

– перевірити можливості додавання даних до програми;

– перевірити можливість продивитися довірчі інтервали та інтервали передбачення;

– перевірити можливість сховати інтервали;

– перевірити можливість розрахувати метрики якості;

– перевірити неможливість ввести файл з даними неправильного формату.

6 Методика випробувань

Тест 1: перевірити можливості додавання даних до програми

Сценарій:

– користувач натискає кнопку «Вибрати файл»;

– відкривається вікно в якому користувач може вибрати шлях до файлу;

– вибір файлу;

– ведення даних з файлу.

Очікуваний результат: після натискання кнопки «Вибрати файл» файл додається.

Тест 2: перевірити можливість продивитися довірчі інтервали та інтервали передбачення

Сценарій:

– Користувач вводить дані.
– Користувач натискає кнопку «Інтервали».
– Інтервали відображаються у таблиці.
– Кнопка «Інтервали» змінює текст на «Сховати інтервали».

Очікуваний результат: після натискання кнопки «Інтервали» на головній формі додається таблиця, яка містить ці інтервали.

Тест 3: перевірити можливість сховати інтервали

Сценарій:

– Користувач ввів дані та натиснув кнопку «Інтервали».
– Користувач натискає кнопку «Сховати інтервали».
– Таблиця з інтервалами зникає із основної форми.
– Кнопка «Сховати інтервали» змінює текст на «Інтервали».

Очікуваний результат: таблиця з інтервалами зникає а кнопка «Сховати інтервали» змінює текст на «Інтервали».

Тест 4: перевірити можливість розрахувати метрики якості

Сценарій:

– Користувач вводить дані.
– Користувач натискає кнопку «Показники».
– Метрики відображаються у формі.

Очікуваний результат: метрики відображаються у формі.

Тест 5: перевірити неможливість ввести файл з даними неправильного формату

Сценарій:

– Користувач вводить дані неправильного формату.
– Програма видає повідомлення про неправильний формат даних.

Очікуваний результат: програма видає повідомлення про неправильний формат даних.

Тест 6: перевірити можливість оцінювання тривалості розробки програмного забезпечення, її довірчого інтервалу та інтервалу передбачення Сценарій:

- Користувач вводить дані.
- Користувач вводить значення трудомісткості для прогнозування.
- Користувач натискає кнопку «Передбачення»

Очікуваний результат: програма розраховує тривалість розробки програмного забезпечення, її довірчий інтервал та інтервал передбачення.

ДОДАТОК Г

ТЕКСТ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗА МЕТОДОЛОГІЄЮ AGILE

lgPreobr.cs

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WindowsFormsApp1
{
    abstract class lgPreobr
    {
        public const double t = 2.3685;
        private double S;
        private double ZxAverage;
        private double sum;
        protected int length;
        protected double[] x;
        protected double[] y;
        protected double[] Yoproximated;
        protected double[] xn;
        protected double[] yn;
        protected double[] delta;
```

```
protected double[] yMinusDelta;
protected double[] yPlusDelta;
protected double[] deltaPrognozirovaniya;
protected double[] yMinusDeltaPrognozirovaniya;
protected double[] yPlusDeltaPrognozirovaniya;
protected double[] Y;
protected double[] YMinusDelta;
protected double[] YPlusDelta;
protected double[] YMinusDeltaPrognozirovaniya;
protected double[] YPlusDeltaPrognozirovaniya;
private double a;
private double b;
protected double rSquare;
protected double MMRE;
protected double PRED;

public abstract void Normalize();
public abstract void Reverse();
protected void countIntervalDoveria()
{
   for (int i = 0; i < length; i++)
   {
      yMinusDelta[i] = Yoproximated[i] - delta[i];
      yPlusDelta[i] = Yoproximated[i] + delta[i];
   }
}

protected void countS()
{
   double sum = 0;
```

```
    for(int i = 0; i < length; i++)
    {
                        sum += (yn[i] - Yoproximated[i]) * (yn[i] -
                Yoproximated[i]);
    }

    S = Math.Sqrt(sum/(length-2));
}

public LgPreobr(double[] x, double[] y, bool check)
{
    this.x = x;
    this.y = y;
    this.length = x.Length;
    this.check = check;

    Yoproximated = new double[length];
    xn = new double[length];
    yn = new double[length];
    delta = new double[length];
    yMinusDelta = new double[length];
    yPlusDelta = new double[length];
    deltaPrognozirovaniya = new double[length];
    yMinusDeltaPrognozirovaniya = new double[length];
    yPlusDeltaPrognozirovaniya = new double[length];
    Y = new double[length];
    YMinusDelta = new double[length];
    YPlusDelta = new double[length];
    YMinusDeltaPrognozirovaniya = new double[length];
```

```
    YPlusDeltaPrognozirovaniya = new double[length];

    this.Normalize();
    this.countOproximation();
    this.countS();
    this.countDeltaDoveria();
    this.countIntervalDoveria();
    this.countDeltaPrognozirovaniya();
    this.countIntervalPrognozirovaniya();
    this.Reverse();
    this.countMMRE();
    this.countPRED();
    this.countRSquare();
}

protected void countOproximation()
{
    Regres regres = new Regres(xn, yn);
    a = regres.getA();
    b = regres.getB();
    for (int i = 0; i < length; i++)
    {
        Yoproximated[i] = a * xn[i] + b;
    }
}
protected void countDeltaPrognozirovaniya()
{
    for (int i = 0; i < length; i++)
    {
```

```
                    deltaPrognozirovaniya[i] = t * S * Math.Sqrt( 1 +
                (double)1 / (double)length + (xn[i] - ZxAverage) * (xn[i] -
                ZxAverage) / sum);
    }
}
protected void countIntervalPrognozirovaniya()
{
    for (int i = 0; i < length; i++)
    {
                    yMinusDeltaPrognozirovaniya[i] = Yoproximated[i]
                - deltaPrognozirovaniya[i];
                    yPlusDeltaPrognozirovaniya[i] = Yoproximated[i] +
                deltaPrognozirovaniya[i];
    }
}
protected void countDeltaDoveria()
{
    double sumZx = 0;

    for(int i = 0; i < length; i++)
    {
        sumZx += xn[i];
    }

    ZxAverage = sumZx / length;

    //count sum (Zx - Zxcp)^2
    sum = 0;
    for (int i = 0; i < length; i++)
    {
```

```
        sum += (xn[i] - ZxAverage) * (xn[i] - ZxAverage);
    }

    for (int i = 0; i < length; i++)
    {
                        delta[i] = t * S * Math.Sqrt( (double)1 /
                (double)length + xn[i] - ZxAverage) * (xn[i] - ZxAverage)
                / sum );
    }

}
protected void countRSquare()
{
    double sum = 0;
    for(int i = 0; i < length; i++)
    {
        sum += y[i];
    }
    double yAverage = sum / length;

    double sum1 = 0;
    for (int i = 0; i < length; i++)
    {
        sum1 += (y[i] - yAverage) * (y[i] - yAverage);
    }

    double sum2 = 0;
    for (int i = 0; i < length; i++)
    {
        sum2 += (y[i] - Y[i]) * (y[i] - Y[i]);
```

```
    }

    rSquare = 1 - sum2 / sum1;
}

protected void countMMRE()
{
    double sum = 0;
    for (int i = 0; i < length; i++)
    {
        sum += Math.Abs(Y[i] - y[i]) / Y[i];
    }
    MMRE = sum / length;
}

protected void countPRED()
{
    double sum = 0;
    for (int i = 0; i < length; i++)
    {
        Debug.WriteLine(Math.Abs(y[i] - Y[i]) / y[i]);
        if(Math.Abs(y[i] - Y[i])/y[i] <= 0.25)
        {
            sum += 1;
        }
    }
    PRED = sum / length;
}

public double[] getY()
```

```
{
   return Y;
}

public double[] getYMinusDelta()
{
   return YMinusDelta;
}

public double[] getYPlusDelta()
{
   return YPlusDelta;
}

public double[] getYMinusDeltaPrognozirovaniya()
{
   return YMinusDeltaPrognozirovaniya;
}

public double[] getYPlusDeltaPrognozirovaniya()
{
   return YPlusDeltaPrognozirovaniya;
}

public double getRSquare()
{
   return rSquare;
}

public double getMMRE()
```

```
        {
            return MMRE;
        }

        public double getPRED()
        {
            return PRED;
        }
    }
}
```

ДОДАТОК Д

ІНСТРУКЦІЯ КОРИСТУВАЧА ПЗ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗА МЕТОДОЛОГІЄЮ AGILE

1. Для того, щоб запустити програму, потрібно запустити файл «T_Agile».exe». При запуску програми з’являється головна форма програми (рис. Д.1).

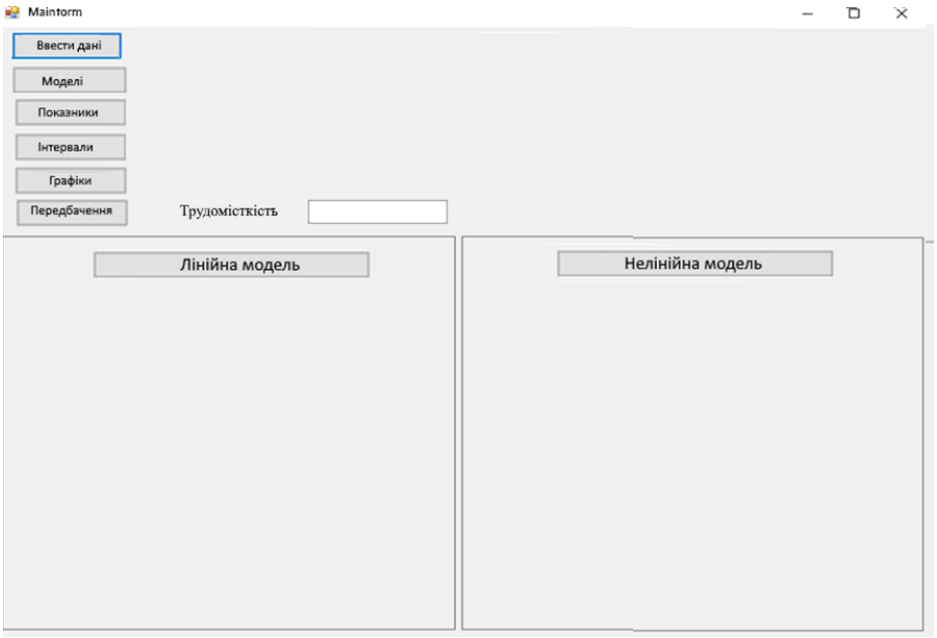

Рисунок Д.1 – Головна форма

2. Для введення даних необхідно натиснути на кнопку «Ввести дані» (рис. Д.2).

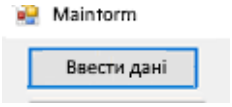

Рисунок Д.2 – Додавання даних

3. Потім необхідно у вікні, що відкриється, знайти файл з даними та обрати його.

4. Після цього необхідно натиснути на кнопку «Моделі» для побудови лінійної та нелінійної моделей(рис. Д.4).

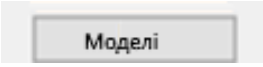

Рисунок Д.3 – Побудова моделей

5. Після побудови моделей можна розраховувати показники, обчислювати довірчі інтервали та інтервали передбачення, будувати графіки та виконувати передбачення.

Наприклад, при виборі пункту «Показники», отримаємо вікно з розрахованими показниками (рис. Д.4).

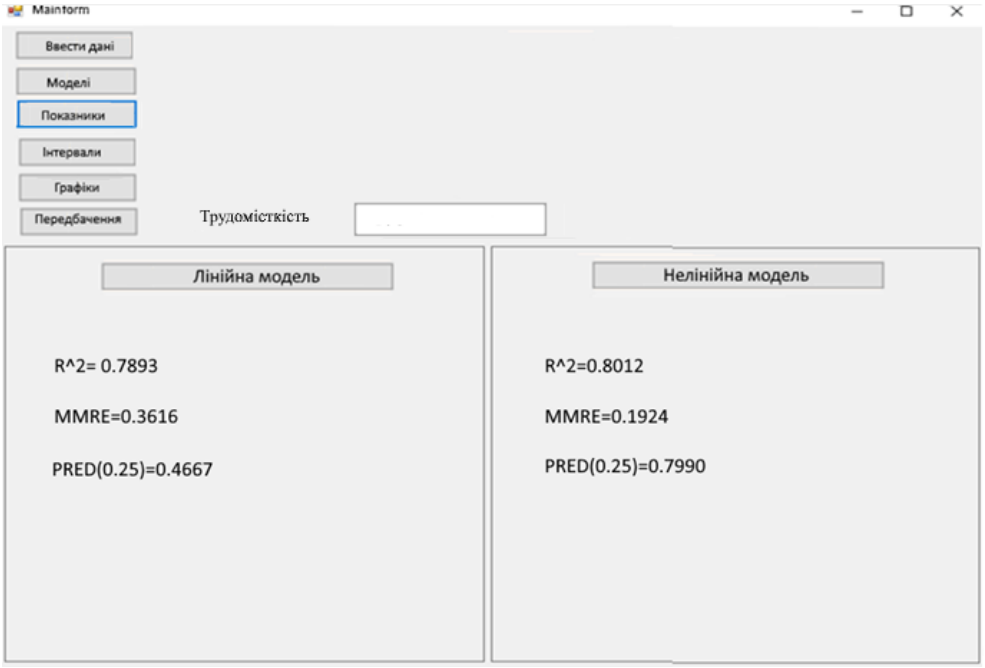

Рисунок Д.4 – Метрики якості

6. При неправильному форматі введених даних програма показує повідомлення про помилку (рис. Д.5)

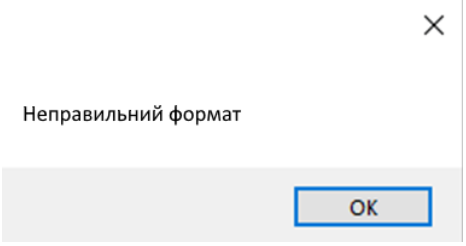

Рисунок Д.5 – Повідомлення про помилку