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

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено

В.о. завідувача кафедри

_____ О.В. Коваль
(підпис) (ініціали, прізвище)

“___”_____2019р.

ДИПЛОМНА РОБОТА
на здобуття ступеня бакалавра

з напряму підготовки 6.050103 «Програмна інженерія»

на тему Автоматизована система розрахунку вартості програмного забезпечення

Виконав: студент 4 курсу, групи ТВз-51

Кисельов Юрій Юрійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник к.е.н., доцент Сегеда І.В. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент к.т.н., с.н.с. кафедри АТЕП Крижанівський К.С. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

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

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший , бакалаврський

Напрям підготовки 6.050103 «Програмна інженерія»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ О.В. Коваль
(підпис)
”___”_____2019р.

ЗАВДАННЯ
на дипломну роботу студенту
Кисельову Юрію Юрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Автоматизована система розрахунку вартості програмного забезпечення

керівник роботи к.е.н., доцент Сегеда Ірина Василівна
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від «22» травня 2019р. №1324

2. Строк подання студентом роботи «10» червня 2019р.
3. Вихідні дані до роботи: мова програмування — C#, середовище розробки — Visual Studio Community 2015
4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити) розглянути та проаналізувати методи і моделі оцінки вартості програмного забезпечення, обґрунтувати вибір моделі оцінки вартості програмного забезпечення та алгоритм розробки програмного додатку, розробити програмне забезпечення, розробити інтерфейс користувача, зробити висновки за результатами роботи
5. Перелік ілюстративного матеріалу
1. Методи оцінки вартості ПЗ. 2. Конструктивна модель витрат (COCOMO). 3. Формули розрахунку базового рівня COCOMO. 4. Формули розрахунку розширеного рівня COCOMO. 5. Засоби розробки. 6. Структура проекту. 7. Діаграма класів. 8. Структура БД. 9. Стартовий інтерфейс додатку. 10. Інтерфейс для обчислення базової моделі COCOMO. 11. Інтерфейс для

обчислення розширеної моделі COCOMO. 12. Інтерфейс для обчислення EDM та PAM моделі COCOMO II. 13. Результати роботи програми.
6. Дата видачі завдання «1» грудня 2018 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі	12.02.2019 – 18.03.2019	
2	Розробка архітектури та загальної структури системи	27.03.2019 – 03.04.2019	
3.	Розробка структур окремих підсистем	09.04.2019 – 16.04.2019	
4.	Програмна реалізація системи	17.04.2019 – 12.05.2019	
5.	Оформлення пояснювальної записки	03.05.2019 – 06.06.2019	
6.	Захист програмного продукту	14.05.2019	
7.	Передзахист	29.05.2019	
8.	Захист	18.06.2019	

Студент _____ Кисельов Ю.Ю.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Сегеда І.В.
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Метою роботи є дослідження існуючих методів та моделей оцінки вартості програмного забезпечення, практичне оволодіння прийомами та навичками створення оцінки програмних систем та проектів, розробка автоматизованої системи розрахунку вартості програмного забезпечення, з використанням засобів оцінки вартості програмного продукту.

Записка містить 65 сторінок, 13 рисунків, 13 таблиць, 4 додатків і 12 бібліографічних найменувань за «Переліком посилань».

Ключові слова: методи оцінки, COCOMO, оцінка вартості програмного забезпечення, C#.

ABSTRACT

The aim of the work is a research of existing methods and models of the software cost estimation, practical acquirement of the techniques and skills to create an assessment of software systems and projects, development of the automated systems for calculating software costs, using tools for evaluating the cost of a software product.

The note contains 65 pages, 13 figures 13 tables, 4 attachments and 12 links.

Keywords: C #, methods of estimation, COCOMO, estimation of cost of the software.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

COCOMO (Constructive Cost Model) — конструктивна модель вартості.

SLOC (Source Lines of Code) — кількість рядків коду.

KSLOC (Kilo of Source Line of Code) — обсяг програмного продукту в тисячах рядків вихідного тексту.

BASIC COCOMO — модель базового рівня.

INTERMEDIATE COCOMO — модель проміжного рівня.

COST DRIVERS (CDk) — атрибути факторів витрат.

EARLY DESIGN — попередня оцінка на початковій фазі проекту.

POST ARCHITECTURE — детальна оцінка після опрацювання архітектури проекту.

ВСТУП

В сучасних економічних умовах для успішної конкуренції компаній особливо актуальна розробка і впровадження нових технологій. Важливим елементом прийняття інвестиційних рішень щодо впровадження технологічних проектів є оцінка їх ефективності. Розрахунок витрат на створення програмного забезпечення (ПЗ) є найважливішим етапом його розробки. Тільки при наявності коректної і обґрунтованої оцінки можлива досконала організація управління проектом, вчасне отримання результатів та раціональний розподіл коштів.

При розробці нового програмного продукту необхідно правильно розрахувати вартість розробки, яка безпосередньо залежить від термінів його виконання. Розвиток моделей, засобів і методів оцінки вартості програмного забезпечення досягло рівня практичного застосування. Однак не всі вони використовуються при розробці ПЗ через відсутність в Україні інформації, засобів і фахівців. Існують труднощі в оцінці витрат на розробку програмного забезпечення, так як необхідно враховувати безліч різноманітних факторів: складність і обсяг розв'язуваної задачі, вартість і доступність необхідних ресурсів, необхідний рівень абстракції при розробці та ін.

Недооцінка вартості, часу та інших ресурсів, необхідних для створення програмного продукту, тягне за собою недостатню чисельність проектної команди, надмірно стислі терміни розробки і, як результат, втрату довіри до розробників у разі порушення графіка. З іншого боку, якщо для проекту виділено більше ресурсів, ніж реально необхідно, причому без належного контролю за їх використанням, то такий програмний продукт виявиться більш дорогим, ніж повинен був бути при грамотній оцінці, що призведе до затримки початку розробки наступного проекту.

Проектування програмного забезпечення сучасних інформаційно-керуючих систем —трудомістка і тривала робота, що потребує участі висококваліфікованих фахівців. Обсяг вихідного коду програмного забезпечення інформаційно технічних систем зростає у часі експоненційно та становить мільйони рядків програмного коду.

Подібне збільшення обсягів ПЗ супроводжується нелінійним збільшенням складності, що, у свою чергу, впливає на собівартість програмного забезпечення. Після завершення процесу розробки ПЗ настає етап поступової його модифікації, що дозволяє враховувати нові інформаційні вимоги. В силу специфіки методу створення програм процес супроводження є досить дорогим, тривалим і важко передбачуваним.

Необхідність врахування всіх вищезазначених складових підтверджує складність та багатофакторність визначення вартості створеного власними силами програмного забезпечення.

Мета виконання даної роботи полягає у розробці автоматизованої системи розрахунку вартості програмного забезпечення з використанням засобів оцінки вартості програмного продукту, проведенні комплексного аналізу існуючих методів та моделей оцінки вартості програмних продуктів, щодо встановлення їх характеристик, переваг та недоліків; у здійсненні вибору алгоритмічних моделей, на базі яких розроблено додаток.

При розробці програмного додатку використано Visual Studio Community 2015 (мова імплементації — C#), Microsoft SQL Server, API Windows Forms, Entity Framework 6.1.3.

У першому розділі записки сформульовано мету роботи, у другому розділі детально проаналізовані вибрані алгоритмічні моделі COCOMO та COCOMO II; третій розділ містить опис засобів розробки; у четвертому розділі здійснено опис програмної реалізації розробленого додатку, п’ятий розділ містить опис методики роботи користувача.

1. ЗАДАЧА РОЗРОБКИ АВТОМАТИЗОВАНОЇ СИСТЕМИ РОЗРАХУНКУ ВАРТОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Система має оцінювати трудоємність і час розробки програмного продукту за методикою COCOMO і COCOMO II, а також розраховувати вартість розробки програмного забезпечення.

В розробленому додатку розрахунки проводяться:

— за базовою моделлю COCOMO, яка застосовується на етапі створення специфікації;

— за розширеною моделлю COCOMO, яка застосовується після визначення вимог до програмного забезпечення;

— за моделлю раннього етапу розробки Early Design Model COCOMO II;

— за пост-архітектурною моделлю Post Architecture Model COCOMO II.

Система має забезпечити:

— вибір типу проекту (Basic COCOMO);

— вибір факторів витрат (Intermediate COCOMO);

— вибір чинників масштабу (EDM та PAM COCOMO II);

— вибір множників трудомісткості (EDM (7 шт.) та PAM COCOMO II (17 шт.)).

База даних повинна зберігати всі необхідні коефіцієнти, для розрахунку трудоємності та часу розробки програмного проекту.

Програма повинна мати прикладний інтерфейс для внесення користувачем вхідних даних та отримання результатів обчислень.

2. РОЗГЛЯД І АНАЛІЗ АЛГОРИТМІВ РІЗНИХ РІВНІВ КОНСТРУКТИВНОЇ МОДЕЛЕЙ Б.БОЕМА ТА ОЦІНКА ВАРТОСТІ ПРОГРАМНОО ПРОДУКТУ

Існує цілий ряд алгоритмічних моделей для прогнозування витрат і собівартості [1, 3, 4, 6] та створення графіка розробки програмного забезпечення. Однією з самих цікавих є модель COCOMO. Ця модель має добре розроблену технічну документацію та є загальнодоступною. Вона пройшла довгий шлях розвитку з часу першої появи у 1981 році, остання версія моделі опублікована у 1995 році.

2.1 Алгоритмічна модель розрахунку вартості програмного забезпечення COCOMO

Модель COCOMO [1, 2, 7, 8, 13] заснована на статистичному аналізі фактичних даних по виконанню 63 проектів в компанії TRW Aerospace. Аналізувалися проекти обсягом від 2 до 100 тисяч рядків коду, на мовах програмування від асемблерів до високорівневої мови PL/1, засновані на каскадній моделі життєвого циклу розробки ПЗ.

Модель складається з ієрархії трьох послідовно деталізованих та уточнюючих рівнів (режимів). На кожному рівні всі проекти розбиваються на три групи за рівнем складності:

1) розповсюджений або органічний тип (organic projects);

2) напівнезалежний або напіврозподілений тип (semidetached projects);

3) вбудований тип (embedded projects).

В моделі COCOMO використовуються три режими (рівні), за допомогою яких класифікується складність системи, а також середовище розробки.

Розповсюджений або органічний (organic) режим. Розповсюджений режим звичайно класифікується як платіжна відомість, опис або наукове обчислення. Інші характеристики режиму: невелика команда по розробці проекту, необхідні невеликі нововведення, є несуворі обмеження і кінцевий термін, а середовище розробки є стабільним.

Напівнезалежний або напіврозподілений (semidetach) режим. Напівнезалежний режим типізується прикладними системами, наприклад, компіляторами, системами баз даних або редакторами. Інші характеристики: невелика команда по розробці проекту середнього розміру, необхідні деякі інновації, помірні обмеження і кінцевий термін, а середовище розробки дещо нестабільне.

Вбудований (embedded) режим. Вбудований режим характеризується режимами реального часу, наприклад, системами контролю повітряного руху або воєнними системами. Інші характеристики: велика команда розробників проекту, великий об’єм необхідних інновацій, жорсткі обмеження і терміни здачі. Середовище розробки в цьому випадку складається з багатьох складних інтерфейсів, включаючи ті з них, які поставляються замовникам разом з апаратним забезпеченням.

Тип тієї чи іншої групи можна розглядати як один з параметрів моделі COCOMO.

Модель COCOMO поділяється на рівні: базовий (basic), проміжний (intermediate), деталізований (advanced).

2.1.1 Базовий рівень (Basic COCOMO)

Модель базового рівня моделі (Basic COCOMO) [1, 7] — двохпараметрична. Як параметри виступають тип проекту і обсяг програми (число рядків коду).

Рівняння цієї моделі мають вигляд (2.1), (2.2), (2.3), (2.4):

$$PM = a_i \times (SIZE)^{b_i}, \tag{2.1}$$

$$TM = c_i \times (PM)^{d_i}, \tag{2.2}$$

$$SS = PM / TM, \tag{2.3}$$

$$P = SIZE / PM, \tag{2.4}$$

де PM (People × Month) — трудомісткість (люд. × міс.);

TM (Time at Month) — час розробки в календарних місяцях;

SIZE — обсяг програмного продукту в тисячах рядків вихідного тексту (Kilo of Source Line of Code — KSLOC);

SS — середня чисельність персоналу;

P — продуктивність.

Коефіцієнти a_i, b_i, c_i, d_i вибираються з таблиці 2.1.

Таблиця 2.1 Значення коефіцієнтів базового рівня моделі COCOMO залежно від типу проекту [7]

Тип проекту	a_i	b_i	c_i	d_i
Розповсюджений (Органічний)	2,4	1,05	2,5	0,38
Напівнезалежний (Напіврозділений)	3,0	1,12	2,5	0,35
Вбудований	3,6	1,20	2,5	0,32

Модель цього рівня підходить для ранньої швидкої приблизної оцінки витрат, але точність її дуже низька.

2.1.2 Проміжний рівень (Intermediate COCOMO)

На цьому рівні базова модель уточнена за рахунок введення додаткових 15 «Атрибутів вартості» (або факторів витрат) Cost Drivers (CDk) [7], які згруповані за чотирма категоріями:

1) Характеристики продукту (Product Attributes):

— *RELY* — Необхідна надійність ПЗ (Required Software Reliability);

— *DATA* — Розмір БД додатку (Size of Application Database);

— *CPLX* — Складність продукту (Complexity of the Product);

2) Характеристики апаратного забезпечення (Hardware Attributes):

— TIME — Обмеження швидкодії при виконанні програми (Run-Time Performance Constraints);

— STOR — Обмеження пам’яті (Memory Constraints);

— VIRT (PVOL) — Нестійкість оточення віртуальної машини (Volatility of the Virtual Machine Environment);

— *TURN (STIME)* — Необхідний час відновлення (Required Turnabout Time);

3) Характеристики персоналу (Personnel Attributes):

— *ACAP (ASAP)* — Аналітичні здібності (Analyst Capability);

— *AEXP* — Досвід розробки (Applications Experience);

— *PCAP (PERS)* — Здібності до розробки ПЗ (Software Engineer Capability);

— *VEXP (PEXP)* — Досвід використання віртуальних машин (Virtual Machine Experience);

— *LEXP (LTEX)* — Досвід розробки на мовах програмування (Programming Language Experience);

4) Характеристики проекту (Project Attributes):

— *MODP (FCIL)* — Застосування методів розробки ПЗ (Application of Software Engineering Methods);

— *TOOL* — Використання інструментарію розробки ПО (Use of Software Tools);

— *SCED* — Вимоги дотримання графіка розробки (Required Development Schedule).

Значення кожного атрибута вибирається з таблиці 2.2 відповідно до його ступеня значущості (рейтингу) в конкретному проекті.

Таблиця 2.2 Значення атрибутів вартості залежно від їх рівня [7]

Атрибути вартості, CD_k	**Рейтинг**					
	Дуже низький	**Низький**	**Середній**	**Високий**	**Дуже високий**	**Критичний**
Характеристики продукту						
1. Необхідна надійність ПЗ	0,75	0,88	1,00	1,15	1,40	n/a
2. Розмір БД додатка	n/a	0,94	1,00	1,08	1,16	n/a
3. Складність продукту	0,70	0,85	1,00	1,15	1,30	1,65
Характеристики апаратного забезпечення						
4. Обмеження швидкодії при виконанні програми	n/a	n/a	1,00	1,11	1,30	1,66

Продовження таблиці 2.2

5. Обмеження пам'яті	n/a	n/a	1,00	1,06	1,21	1,56
6. Нестійкість оточення віртуальної машини	n/a	0,87	1,00	1,15	1,30	n/a
7. Не2обхідний час відновлення	n/a	0,87	1,00	1,07	1,15	n/a
Характеристики персоналу						
8. Аналітичні здібності	1,46	1,19	1,00	0,86	0,71	n/a
9. Досвід розробки	1,29	1,13	1,00	0,91	0,82	n/a
10. Здібності до розробки ПЗ	1,42	1,17	1,00	0,86	0,70	n/a
11. Досвід використання віртуальних машин	1,21	1,10	1,00	0,90	n/a	n/a
12. Досвід розробки на мовах програмування	1,14	1,07	1,00	0,95	n/a	n/a
Характеристики проекту						
13. Застосування методів розробки ПЗ	1,24	1,10	1,00	0,91	0,82	n/a
14. Використання інструментарію розробки	1,24	1,10	1,00	0,91	0,83	n/a
15. Вимоги дотримання графіка розробки	1,23	1,08	1,00	1,04	1,10	n/a

Примітка: n/a (not available) - дані відсутні, тобто відповідний рівень не оцінюється

Формула проміжного рівня моделі має вигляд (2.5)

$$PM = EAF \times a_i \times (SIZE)^{b_i}, \tag{2.5}$$

де PM — трудомісткість (люд. × міс.);

$SIZE$ — обсяг програмного продукту в тисячах рядків вихідного тексту (Kilo of Source Line of Code — KSLOC).

EAF (Effort Adjustment Factor) .(2.6) — добуток обраних атрибутів вартості з таблиці 2.2.:

$$EAF = \prod_{k=1}^{15} CD_k, \tag{2.6}$$

Коефіцієнти моделі вибираються з таблиці 2.3.

Таблиця 2.3 Значення коефіцієнтів проміжного рівня моделі COCOMO залежно від типу проекту [7]

Тип проекту, *i*	a_i	b_i
1. Розповсюджений	3,2	1,05
2. Напівнезалежний	3,0	1,12
3. Вбудований	2,8	1,20

Час розробки розраховується за тією ж формулою, що і для базової моделі.

2.2 Алгоритмічна модель розрахунку вартості програмного забезпечення COCOMO II

У 1997 методика була вдосконалена і отримала назву COCOMO II [1, 7, 14, 15].

Розрізняють дві стадії оцінки проекту:

1) попередня оцінка на початковій фазі (Early Design)
2) детальна оцінка після опрацювання архітектури (Post Architecture).

Для розрахунку трудомісткості необхідно спочатку оцінити фактори (чинники) маштабу (Scale Drivers) та множники трудомісткості (Cost Drivers або Effort Multipliers). Фактори масштабу застосовуються на двох стадіях оцінки проекту. Множники трудомісткості відрізняються для різних стадій оцінки проекту. На різних стадіях відрізняється їх кількість і значення. Для стадії Early Design необхідно оцінити сім множників трудомісткості, а для стадії Post Architecture — сімнадцять.

Формула оцінки трудомісткості проекту в люд. × міс. має вигляд (2.7)

$$PM = EAF \times A \times (SIZE)^E, \qquad (2.7)$$

де $E = B + 0{,}01 \times \sum_{j=1}^{5} SF_j$,

$B = 0{,}91$, $A = 2{,}94$ для попередньої оцінки;

$A = 2{,}45$ для детальної оцінки;

SF_j — фктори масштабу (Scale Factors) з таблиць 2.4-2.5

SIZE — обсяг програмного продукту в тисячах рядків вихідного тексту (KSLOC — Kilo of Source Line of Code);

EM_j — множники трудомісткості (Effort Multipliers). n=7 — для попередньої оцінки таблиця 2.6, n=17 — для детальної оцінки таблиця 2.7;

EAF — (Effort Adjustment Factor) — добуток обраних множників трудоємкості (2.8):

$$EAF = \prod_{k=1}^{n} EM_k , \tag{2.8}$$

В методиці використовуються п’ять факторів масштабу, які визначаються наступними характеристиками проекту:

PREC — прецедентність, наявність досвіду аналогічних розробок,

FLEX — гнучкість процесу розробки,

RESL — архітектура і дозвіл ризиків,

TEAM — спрацьованість команди,

PMAT — зрілість процесів.

Всі фактори масштабу мають певну оцінку: Very Low — дуже низька оцінка фактора, Low — низька оцінка, Nominal — середня оцінка, High — висока оцінка, Very High — дуже висока оцінка, Extra High — критично висока оцінка.

Детальний опис факторів масштабу наведено в таблиці 2.4.

Таблиця 2.4 Опис рівнів значущості чинників масштабу [7]

SFj	Опис	Рівень значущості чинника					
		Дуже низький	Низький	Середній	Високий	Дуже високий	Критичний
1. PREC. Precedentedness	Прецедентність, наявність досвіду аналогічних розробок	досвід у продукті і платформі відсутній	продукт і платформа не дуже знайомі	деякий досвід в продукті і платформі присутній	продукт і платформа в основному відомі	продукт і платформа великою мірою знайомі	продукт і платформа повністю знайомі

Продовження таблиці 2.4

2. FLEX. Development Flexibility	Гнучкість процесу розробки	процес строго детермінований	допускаються деякі компроміси	значна жорсткість процесу	відносна жорсткість процесу	незначна жорсткість процесу	визначені тільки загальні цілі
3. RESL. Architecture / Risk Resolution	Архітектура і дозвіл ризиків	ризики відомі / / проаналізовані на 20%	ризики відомі / / проаналізовані на 40%	ризики відомі / / проаналізовані на 60%	ризики відомі / / проаналізовані на 75%	ризики відомі / / проаналізовані на 90%	ризики дозволені на 100%
4. TEAM. Team Cohesion	спрацьованість команди	формальна взаємодія	важка взаємодія до деякої міри	частіше колективна робота	у основному колективна робота	висока міра взаємодії	повна довіра, взаємозаміна і взаємодопомога
5. PMAT. Process Maturity	зрілість процесів	CMM Рівень 1 (нижче середнього)	CMM Рівень 1 (вище середнього)	CMM Рівень 2	CMM Рівень 3	CMM Рівень 4	CMM Level 5

Ці фактори застосовуються на обох стадіях оцінки проекту.

Числові значення фактору масштабу в залежності від оцінки його рівня, наведені в таблиці 2.5.

Таблиця 2.5 Значення чинника масштабу залежно від оцінки його рівня [7]

Чинник масштабу, *SFj*	**Оцінка рівня чинника (фактора)**					
	Very Low	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
1. PREC	6,20	4,96	3,72	2,48	1,24	0,00
2. FLEX	5,07	4,05	3,04	2,03	1,01	0,00
3. RESL	7,07	5,65	4,24	2,83	1,41	0,00
4. TEAM	5,48	4,38	3,29	2,19	1,10	0,00
5. PMAT	7,80	6,24	4,68	3,12	1,56	0,00

Кількість і значення множників трудомісткості відрізняються для різних стадій оцінки проекту.

2.2.1 Стадія попередньої оцінки трудомісткості програмного проекту (Early Design)

Для цієї оцінки [7] необхідно оцінити для проекту рівень семи множників трудомісткості :

— **параметри персоналу:**

1. PERS (Personnel Capability) — кваліфікація персоналу (Extra Low — аналітики і програмісти мають нижчу кваліфікацію, плинність більше 45%; Extra High — аналітики і програмісти мають вищу кваліфікацію, плинність менше 4%);

2. PREX (Personnel Experience) — досвід персоналу (Extra Low — новий додаток, інструменти і платформа; Extra High — додаток, інструменти і платформа добре відомі);

— **параметри продукту:**

3. RCPX (Product Reliability and Complexity) — складність і надійність продукту (Extra Low — продукт простий, спеціальних вимог по надійності немає, БД маленька, документація не потрібна; Extra High — продукт дуже складний, вимоги по надійності жорсткі, БД надвелика, документація потрібно в повному обсязі);

4. RUSE (Developed for Reusability) — розробка для повторного використання (Low — не вимагається; Extra High — передбачається повторне використання в інших продуктах);

— **параметри платформи:**

5. PDIF (Platform Difficulty) — складність платформи розробки (Extra Low

— спеціальні обмеження по пам'яті і швидкодії відсутні, платформа стабільна; Extra High — жорсткі обмеження по пам'яті і швидкодії, платформа нестабільна);

— **параметри проекта:**

6. FCIL (Facilities) — обладнання (Extra Low — інструменти найпростіші, комунікації ускладнені; Extra High — інтегровані засоби підтримки життєвого циклу, інтерактивні мультимедіа комунікації);

7. SCED (Required Development Schedule) — необхідний виконання графіка робіт (Very Low — 75% від номінальної тривалості; Very High — 160% від

номінальної тривалості).

Значення множників трудомісткості в залежності від їх рівня наведені в таблиці 2.6.

Таблиця 2.6 Значення множників трудомісткості залежно від оцінки їх рівня (Early Design) [7]

№	Множник трудомісткості, EMi	Оцінка рівня множника трудомісткості						
		Extra Low	Very Low	Low	Nominal	High	Very High	Extra High
1	PERS	2,12	1,62	1,26	1,00	0,83	0,63	0,50
2	PREX	1,59	1,33	1,22	1,00	0,87	0,74	0,62
3	RCPX	0,49	0,60	0,83	1,00	1,33	1,91	2,72
4	RUSE	n/a	n/a	0,95	1,00	1,07	1,15	1,24
5	PDIF	n/a	n/a	0,87	1,00	1,29	1,81	2,61
6	FCIL	1,43	1,30	1,10	1,00	0,87	0,73	0,62
7	SCED	n/a	1,43	1,14	1,00	1,00	n/a	n/a

Примітка: n/a (not available) - дані відсутні, тобто відповідний рівень не оцінюється

2.2.2 Стадія детальної оцінки після опрацювання архітектури (Post Architecture)

Для цієї оцінки [7] необхідно оцінити для проекту рівень сімнадцяти множників трудомісткості EM_j :

— **параметри персоналу:**

1) Analyst Capability (ACAP) — можливості аналітика;

2) Applications Experience (AEXP) — досвід розробки додатків;

3) Programmer Capability (PCAP) — можливості програміста;

4) Personnel Continuity (PCON) — тривалість роботи персоналу;

5) Platform Experience (PEXP) — досвід роботи з платформою;

6) Language and Tool Experience (LTEX) — досвід використання мови програмування і інструментальних засобів.

— **параметри продукту:**

7) Required Software Reliability (RELY) — необхідна надійність програми;

8) Database Size (DATA) — розмір бази даних;

9) Software Product Complexity (CPLX) — складність програми;

10) Required Reusability (RUSE) — необхідна можливість багаторазового

використання;

11) Documentation Match to Life-Cycle Needs (DOCU) — відповідність документації потребам життєвого циклу.

— **параметри платформи:**

12) Execution Time Constraint (TIME) — обмеження часу виконання;

13) Main Storage Constraint (STOR) — обмеження пам'яті;

14) Platform Volatility (PVOL) — змінність платформи.

— **параметри проекту:**

15) Use of Software Tools (TOOL) — використання інструментальних програмних засобів;

16) Multisite Development (SITE) — багатоабонентська (віддалена) розробка;

17) Required Development Schedule (SCED) — необхідний виконання графіка робіт.

Значення множників трудомісткості в залежності від їх рівня наведені в таблиці 2.7

Таблиця 2.7 Значення множників трудомісткості залежно від оцінки їх рівня (Post Architecture) [7]

№	Effort Multiplier, EMJ		Very Low	Low	Nominal	High	Very High	Extra High
	Personnel Factors							
1	ACAP	Analyst Capability	1,42	1,29	1,00	0,85	0,71	n/a
2	AEXP	Applications Experience	1,22	1,10	1,00	0,88	0,81	n/a
3	PCAP	Programmer Capability	1,34	1,15	1,00	0,88	0,76	n/a
4	PCON	Personnel Continuity	1,29	1,12	1,00	0,90	0,81	n/a
5	PEXP	Platform Experience	1,19	1,09	1,00	0,91	0,85	n/a
6	LTEX	Language and Tool Experience	1,20	1,09	1,00	0,91	0,84	n/a
	Product Factors							
7	RELY	Required Software Reliability	0,84	0,92	1,00	1,10	1,26	n/a
8	DATA	Database Size	n/a	0,23	1,00	1,14	1,28	n/a
9	CPLX	Software Product Complexity	0,73	0,87	1,00	1,17	1,34	1,74
10	RUSE	Required Reusability	n/a	0,95	1,00	1,07	1,15	1,24
11	DOCU	Documentation Match to Life -	0,81	0,91	1,00	1,11	1,23	n/a
		Cycle Needs						

Продовження таблиці 2.7

	Platform Factors							
12	TIME	Execution Time Constraint	n/a	n/a	1,00	1,11	1,29	1,63
13	STOR	Main Storage Constraint	n/a	n/a	1,00	1,05	1,17	1,46
14	PVOL	Platform Volatility	n/a	0,87	1,00	1,15	1,30	n/a
	Project Factors							
15	TOOL	Use of Software Tools	1,17	1,09	1,00	0,90	0,78	n/a
17	SITE	Multisite Development	1,22	1,09	1,00	0,93	0,86	0,80
16	SCED	Required Development Schedule	1,43	1,14	1,00	1,00	1,00	n/a

Примітка: n/a (not available) - дані відсутні, тобто відповідний рівень не оцінюється

Тривалість проекту або час розробки проекту TM в методиці COCOMO II для обох рівнів розраховується за формулою (2.9):

$$TM = SCED \times C \times (PM_{NS})^{B+0,2\times(E-B)}, \quad (2.9)$$

де $C = 3{,}67; D = 0.28;$ PM_{NS} — розрахована трудомісткість проекту без урахування множника SCED, що визначає стиснення розкладу.

2.3 Оцінка вартості програмного продукту

Зазвичай, для оцінки вартості проекту по розробці програмного забезпечення використовують три параметра [6, 11]:

— вартість апаратних засобів і програмного забезпечення;

— витрати на відрядження і навчання;

— витрати на персонал.

В більшості випадків найбільш суттєвими є витрати на персонал. Загальна сума витрат на персонал складається з декількох статей витрат:

— витрати на утримання офісу;

— витрати на утримання допоміжного персоналу;

— витрати на утримання комп’ютерної мережі і засобів зв’язку;

— витрати на соціальне забезпечення;

Накладні витрати прирівнюються до подвійної зарплатні програміста, в залежності від розміру компанії і витрат на її утримання.

В даній роботі вартість розробки програмного забезпечення [9] вираховується за формулою (2.10)

$$ВАРТІСТЬ\_РОЗРОБКИ = ТРУДОМІСТКІСТЬ * ПИТОМА\_ВАРТІСТЬ, \quad (2.10)$$

Де $ТРУДОМІСТКІСТЬ$ розраховується за методиками COCOMO і COCOMO II,

$ПИТОМА\_ВАРТІСТЬ$ — питома вартість витрат на людину в місяць.

3. ЗАСОБИ РОЗРОБКИ

При створенні програмного продукту дуже важливим є вибір засобів реалізації. Проаналізувавши існуючі технології, мовою розробки я вибрав C#, яка забезпечує якість і надійність ПЗ, а також швидкість розробки.

Програмний додаток Автоматизована системи розрахунку вартості програмного забезпечення був створений в Visual Studio Community 2015 — інтегрованому середовищі розробки програмного забезпечення від фірми Microsoft. Була використана мова програмування C#. Графічний інтерфейс користувача був розроблений за допомогою API Windows Forms. Для збереження даних була використана система управління базами даних (СУБД) Microsoft SQL Server. Для доступу до бази даних та автоматичного створення сутностей та управлінням ними був застосований Entity Framework 6.1.3.

3.1 Середовище розробки Visual Studio Community 2015

Середовище Visual Studio [17] дозволяє розробляти додатки, використовуючи різні мови програмування: Visual C#, Visual Basic, Visual F#, Visual C++, Python і т.д. Також існує можливість розробляти додатки не тільки під Windows, а і під інші популярні платформи: Android, iOS.

Інтегроване середовище розробки Visual Studio - це оригінальне середовище запуску, яке дозволяє редагувати, налагоджувати і створювати код, а потім публікувати додатки. Інтегроване середовище розробки (IDE) — це багатофункціональна програма, яку можна використовувати для різних аспектів розробки програмного забезпечення. Крім стандартного редактора і відладчика, які існують в більшості середовищ IDE, Visual Studio включає в себе компілятори, засоби виконання коду, графічні конструктори і багато інших функцій для спрощення процесу розробки програмного забезпечення.

3.2 Мова програмування C#

На сьогоднішній момент мова програмування C# [16, 18, 19] одна з найпотужніших, що швидко розвиваються і затребуваних мов в IT-галузі.

Мову C # і пов'язану з нею середу .NET Framework можна без перебільшення назвати найзначнішою з пропонованих в даний час технологій для розробників. Середовище .NET було створено для того, щоб в ньому можна було розробляти практично будь-який додаток для запуску в Windows, а C # є мовою програмування, яка була спеціально створена для використання в .NET Framework. Наприклад, із застосуванням C # і .NET Framework можна створювати динамічні веб-сторінки, додатки Windows Presentation Foundation, веб-служби XML, компоненти для розподілених додатків, компоненти для доступу до баз даних, класичні настільні додатки Windows і навіть клієнтські програми нового інтелектуального типу, що володіють можливостями для роботи в оперативному і автономному режимах.

C # є об'єктно-орієнтованим і в цьому плані багато перейняв у Java і C ++. Наприклад, C # підтримує поліморфізм, успадкування, перевантаження операторів, статичну типізацію. Об'єктно-орієнтований підхід дозволяє вирішити завдання з побудови великих, але в той же час гнучких, масштабованих і розширюваних додатків.

C# — мова, призначена не тільки для створення програмного забезпечення, здатного працювати в Інтернеті. Вона надає засоби для кодування практично будь-якого типу програмного забезпечення або компонентів для платформи Windows. Мова C# і середовище .NET привели до революційних змін в написанні програм і зробили програмування додатків для Windows набагато простішим, ніж будь-коли.

C# — характеризується двома наступними перевагами:

— C# спроектована і розроблена спеціально для застосування з Microsoft .NET Framework (розвиненою платформою розробки, розгортання і виконання розподілених додатків).

— C# — мова, заснована на сучасній об'єктно-орієнтованої методології

проектування, при розробці якого фахівці з Microsoft спиралися на досвід створення подібних мов, побудованих відповідно до запропонованих близько 20 років тому об'єктно-орієнтованими принципами.

3.3 Entity Framework 6.1.3

Платформа Entity Framework [20] являє собою спеціальну об'єктно-орієнтовану технологію на базі фреймворка .NET для роботи з даними.

Entity Framework дає можливість працювати з даними на більш високому рівні абстракції, створювати і супроводжувати додатки, орієнтовані на роботу з даними, одночасно з цим скорочуючи обсяг коду, в порівнянні з традиційними програмами. Так як Entity Framework входить до складу .NET Framework, Entity Framework додатки можуть виконуватися на будь-якому комп'ютері, на якому встановлена платформа .NET Framework.

Центральної концепцією Entity Framework є поняття сутності або entity. Сутність представляє набір даних, асоційованих з певним об'єктом. Тому дана технологія передбачає роботу не з таблицями, а з об'єктами і їх наборами.

Сутності можуть бути пов'язані асоціативним зв'язком один-до-багатьох, один-до-одного і багато-до-багатьох, подібно до того, як в реальній базі даних відбувається зв'язок через зовнішні ключі.

Відмінною рисою Entity Framework є використання запитів LINQ для вибірки даних з БД. За допомогою LINQ є можливість не тільки отримувати певні рядки, що зберігають об'єкти, з бази даних, а й отримувати об'єкти, пов'язані різними асоціативними зв'язками.

Іншим ключовим поняттям є Entity Data Model. Ця модель зіставляє класи сутностей з реальними таблицями в БД. Entity Data Model складається з трьох рівнів: концептуального, рівень сховища і рівень зіставлення (маппінга).

На концептуальному рівні відбувається визначення класів сутностей, які використовуються в додатку.

Рівень сховища визначає таблиці, стовпці, відносини між таблицями і типи

даних, з якими порівнюється використовувана база даних.

Рівень зіставлення (маппінга) служить посередником між попередніми двома, визначаючи зіставлення між властивостями класу суті і стовпцями таблиць.

Таким чином, можливо через класи, визначені у додатку, взаємодіяти з таблицями з бази даних.

Entity Framework передбачає три можливі способи взаємодії з базою даних:

— Database first: Entity Framework створює набір класів, які відображають модель конкретної бази даних;

— Model first: спочатку розробник створює модель бази даних, по якій потім Entity Framework створює реальну базу даних на сервері;

— Code first: розробник створює клас моделі даних, які будуть зберігатися в БД а потім Entity Framework за цією моделлю генерує базу даних і її таблиці.

3.4 Microsoft SQL Server

Microsoft SQL Server — це система управління реляційними базами даних, або RDBMS, яка підтримує широкий спектр обробки транзакцій, бізнес-аналітики та аналітичних додатків у корпоративних IT-середовищах. Це одна з трьох провідних технологій баз даних, разом з Oracle Database та IBM DB2.

Основним компонентом Microsoft SQL Server є SQL Server Database Engine, який керує зберіганням, обробкою та безпекою даних. Вона включає в себе реляційний движок, який обробляє команди і запити, і механізм зберігання, який управляє файлами баз даних, таблицями, сторінками, індексами, буферами даних і транзакціями. Зберігаються процедури, тригери, перегляди та інші об'єкти бази даних також створюються і виконуються в Database Engine.

Як і інше програмне забезпечення реляційних СУБД, Microsoft SQL Server побудований на основі стандартизованої мови програмування SQL, яку IT-спеціалісти використовують для управління базами даних і запиту даних, які вони містять. SQL Server прив'язаний до Transact-SQL (T-SQL), реалізації SQL від Microsoft, що додає набір розширень власного програмування до стандартної мови.

Одна з найважливіших функції захисту SQL Server включає прозоре шифрування даних, яке шифрує файли даних у базах даних, а також дрібнозернистий аудит, який збирає детальну інформацію про використання бази даних для звітування про відповідність нормам. Корпорація Майкрософт також підтримує протокол безпеки транспортного рівня для забезпечення зв'язку між клієнтами SQL Server і серверами баз даних.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Додаток розроблений за схемою дворівневої архітектури «Клієнт-Сервер», яка в подальшому може бути масштабована без додаткової зміни програмного коду для колективної роботи користувачів. Для реалізації цієї схеми виконуваний файл та допоміжні бібліотеки копіюються на файловий сервер, а база даних (БД) встановлюється на окремий сервер (рисунок 4.1).

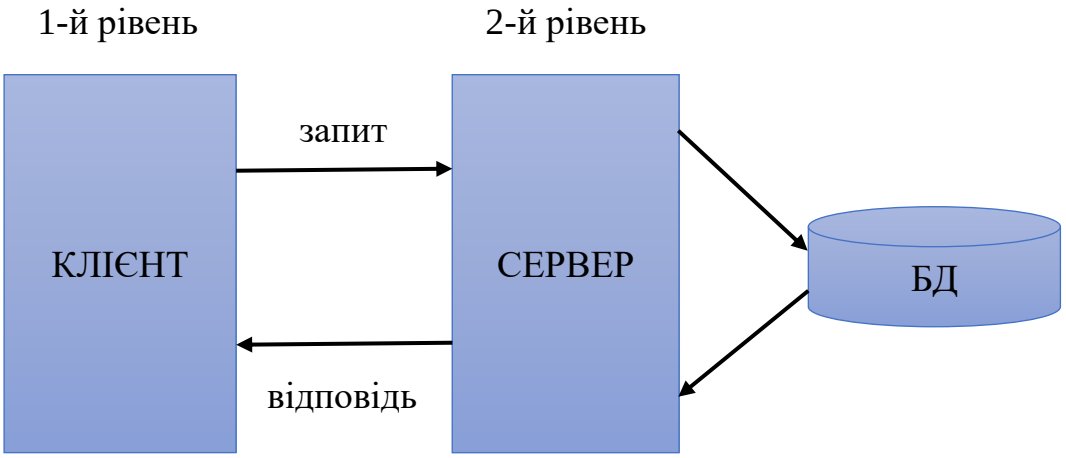

Рисунок 4.1 — Схема дворівневої архітектури

Перевагами даної схеми є:

— можливість розподілу функцій обчислювальної системи між декількома незалежними комп’ютерами;

— збереження всіх даних на окремому захищеному сервері баз даних;

— можливість одночасної роботи групи користувачів;

— гарантія цілісності даних.

4.1 Структура проекту

Структура проекту складається з трьох директорій:

— Business, в якій зберігаються файли для обробки даних проекту в залежності від обраної моделі ;

— Data, в якій зберігається база даних, опис даних та контекст бази даних;

— Presentation, в якій зберігається опис параметрів графічного інтерфейсу користувача.

4.2 Діаграма класів

Діаграма класів проекту (рисунок 4.2) відображає класи і відносини між ними.

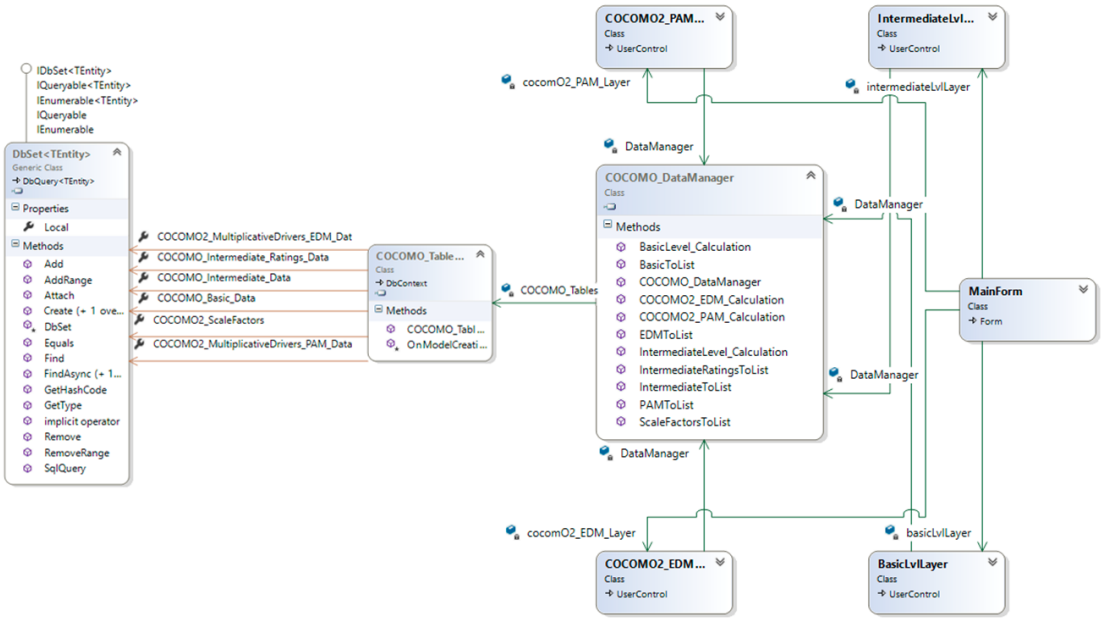

Рисунок 4.2 — Діаграма класів проекту

Діаграма класів використовується на етапі проектування, для відображення структури класів, що формують архітектуру системи.

4.3 Структура бази даних

База даних складається з шести таблиць (рисунок 4.3):

— COCOMO_Basic;

— COCOMO_Intermediate;

— COCOMO_Intermediate_Ratings;

— COCOMO2_MultiplicativeDrivers_EDM;

— COCOMO2_MultiplicativeDrivers_PAM;

— COCOMO2_ScaleFactors.

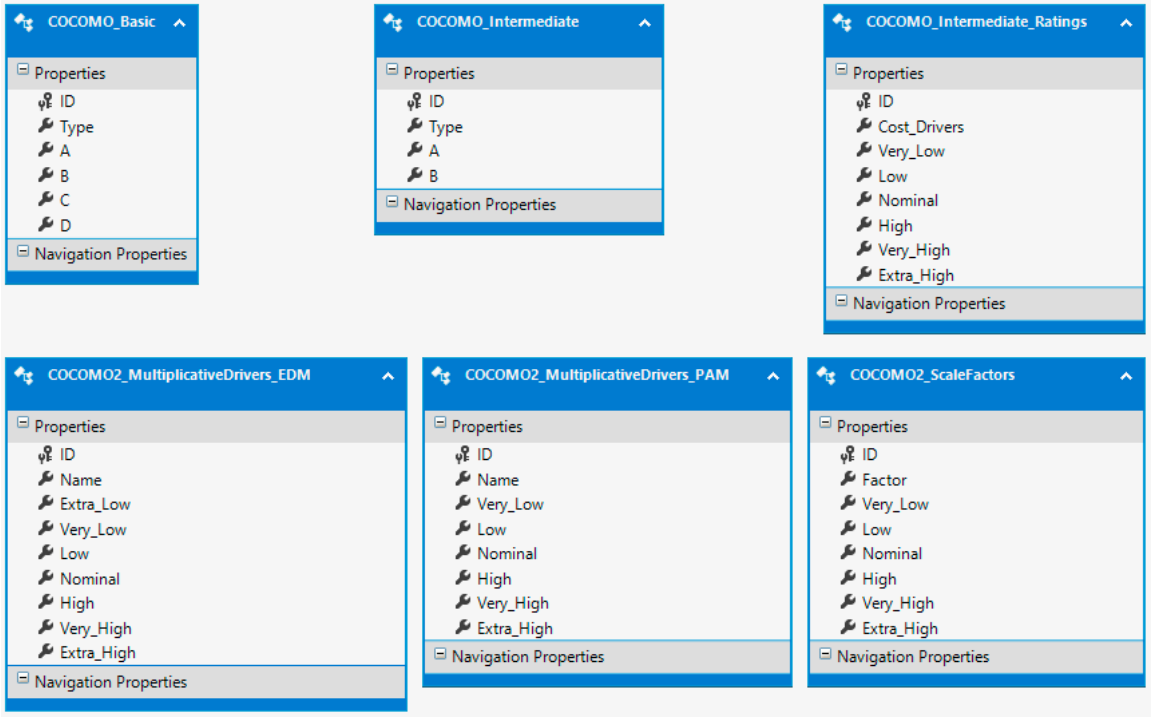

Рисунок 4.3 — Структура БД

Таблиця COCOMO_Basic призначена для зберігання значень коефіцієнтів базового рівня моделі COCOMO в залежності від типу проекту, її структура описана в таблиці 4.1.

Таблиця 4.1 Структура таблиці COCOMO_Basic

Назва	**Тип поля**	**Опис**
ID	int	Унікальний ідентифікатор
Type	nvarchar	Тип проекту
A	float	Коефіцієнти базового рівня моделі COCOMO
B	float	
C	float	
D	float	

Таблиця COCOMO_Intermediate призначена для зберігання значень коефіцієнтів проміжного рівня моделі COCOMO, її структура описана в таблиці 4.2.

Таблиця 4.2 Структура таблиці COCOMO_ Intermediate

Назва	**Тип поля**	**Опис**
ID	int	Унікальний ідентифікатор
Type	nvarchar	Тип проекту
A	float	Коефіцієнти проміжного рівня моделі COCOMO
B	float	

В таблиці COCOMO_Intermediate_Ratings зберігаються значення атрибутів вартості, її структура описана в таблиці 4.3.

Таблиця 4.3 Структура таблиці COCOMO_Intermediate_Ratings

Назва	**Тип поля**	**Опис**
ID	int	Унікальний ідентифікатор
Cost Drivers	nvarchar	Атрибути вартості
Very Low	float	Значення коефіцієнтів відповідно до рейтингу
Low	float	
Nominal	float	
High	float	
Very High	float	
Extra High	float	

Таблиця COCOMO2_ScaleFactors призначена для зберігання значень коефіцієнтів EDM та PAM моделі COCOMO II, її структура описана в таблиці 4.4.

Таблиця 4.4 Структура таблиці COCOMO2_ScaleFactors

Назва	**Тип поля**	**Опис**
ID	int	Унікальний ідентифікатор
Factor	nvarchar	Фактор масштабу
Very Low	float	Значення коефіцієнтів відповідно до рейтингу
Low	float	
Nominal	float	
High	float	
Very High	float	
Extra High	float	

Таблиця COCOMO2_MultiplicativeDrivers_EDM призначена для зберігання значень множників трудомісткості COCOMO II EDM, її структура описана в таблиці 4.5.

Таблиця 4.5 Структура таблиці COCOMO2_MultiplicativeDrivers_EDM

Назва	**Тип поля**	**Опис**
ID	int	Унікальний ідентифікатор
Name	nvarchar	Множник трудомісткості
Extra Low	float	Значення коефіцієнтів відповідно до рейтингу
Very Low	float	
Low	float	
Nominal	float	
High	float	
Very High	float	
Extra High	float	

Таблиця COCOMO2_MultiplicativeDrivers_PAM призначена для зберігання значень множників трудомісткості COCOMO II PAM, її структура описана

в таблиці 4.6.

Таблиця 4.6 Структура таблиці COCOMO2_MultiplicativeDrivers_EDM

Назва	**Тип поля**	**Опис**
ID	int	Унікальний ідентифікатор
Name	nvarchar	Множник трудомісткості
Very Low	float	Значення коефіцієнтів відповідно до рейтингу
Low	float	
Nominal	float	
High	float	
Very High	float	
Extra High	float	

5. МЕТОДИКА РОБОТИ КОРИСТУВАЧА

Кількість рядків оцінено з використанням методу функціональних точок.

Введення даних здійснюється користувачем в залежності від обраної моделі розрахунку вартості.

Розрахунок вартості здійснюється на основі введених та вибраних користувачем даних.

Результати розрахунків виводяться у вікно форми, в залежності від обраної моделі розрахунку.

Перед початком роботи виконується запуск розробленого додатку. Після запуску відображається форма з вкладками (рисунок 5.1).

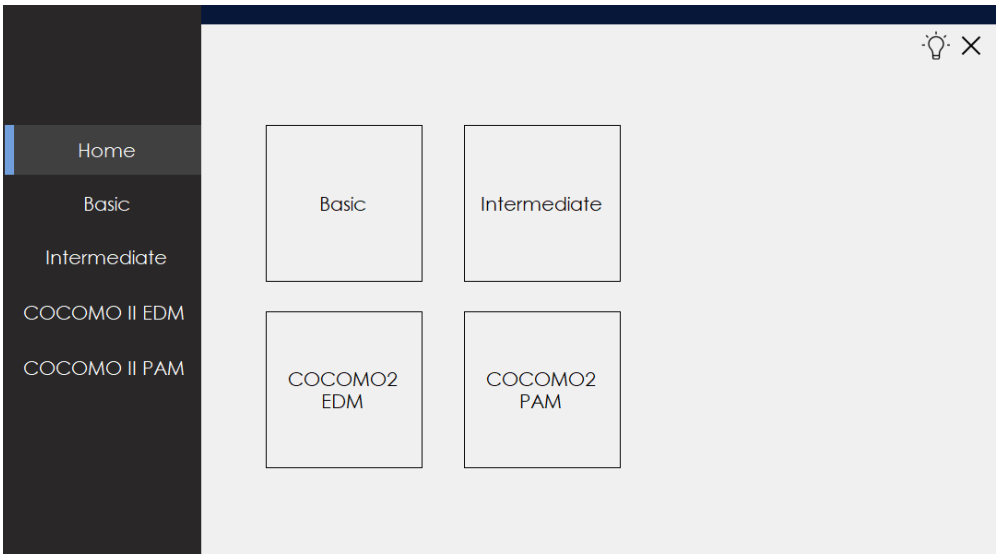

Рисунок 5.1 — Вікно форми COCOMO Home

На вкладці COCOMO Basic здійснюються розрахунки для базового рівня моделі COCOMO. Для цього з випадаючого списку поля **Тип проекту** необхідно вибрати тип проекту (поширений, вбудований або напівнезалежний) і в полі введення **Обсяг програмного продукту** задати обсяг програмного коду в тисячах рядків (рисунок 5.2).

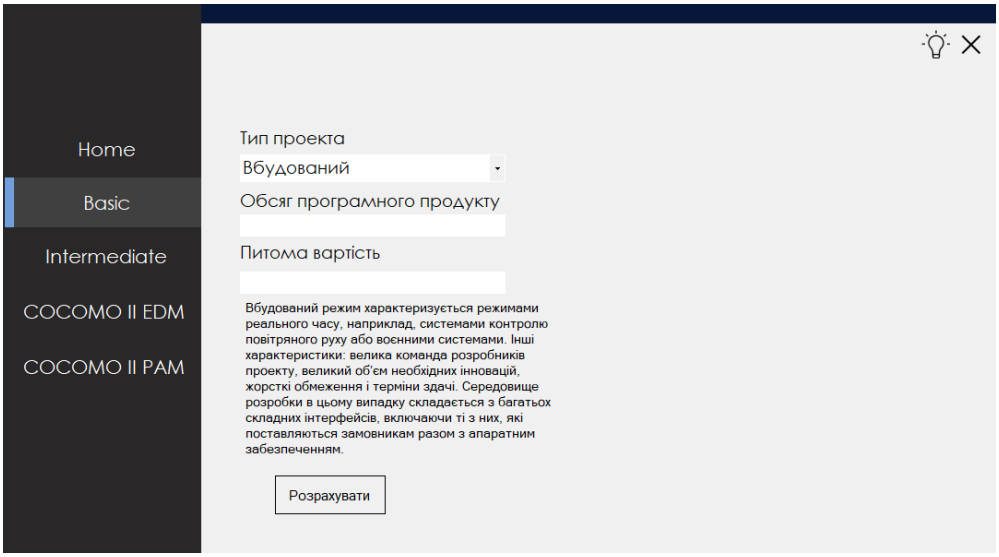

Рисунок 5.2 — Вікно форми COCOMO Basic

Після натискання кнопки **Розрахувати** програма видає трудомісткість проекту в людино * місяцях, час розробки в місяцях і вартість розробки в умовних грошових одиницях (рисунок 5.3).

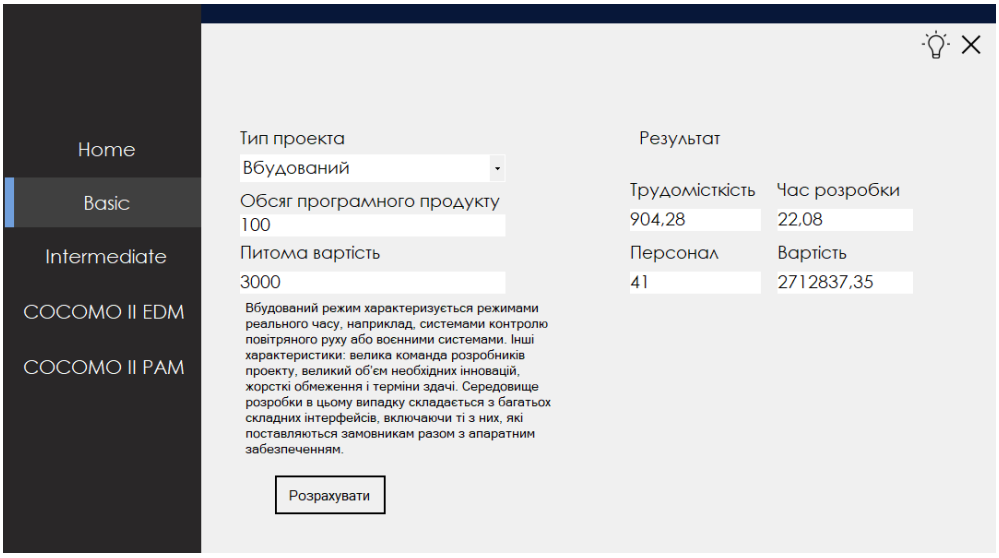

Рисунок 5.3 — Вікно форми COCOMO Basic з результатами розрахунку

На вкладці COCOMO Intermediate проводиться розрахунок для проміжного рівня моделі COCOMO. Крім типу проекту і його розміру в цьому випадку необхідно вибрати за допомогою перемикача одну з 4 категорій витрат.

При виборі категорії **Характеристики продукту** (рисунок 5.4) з випадаючих списків вибираємо рейтинг (дуже низький, низький, середній, високий, дуже високий, критичний) для факторів витрат: Необхідна надійність ПЗ; Розмір БД додатку; Складність продукту.

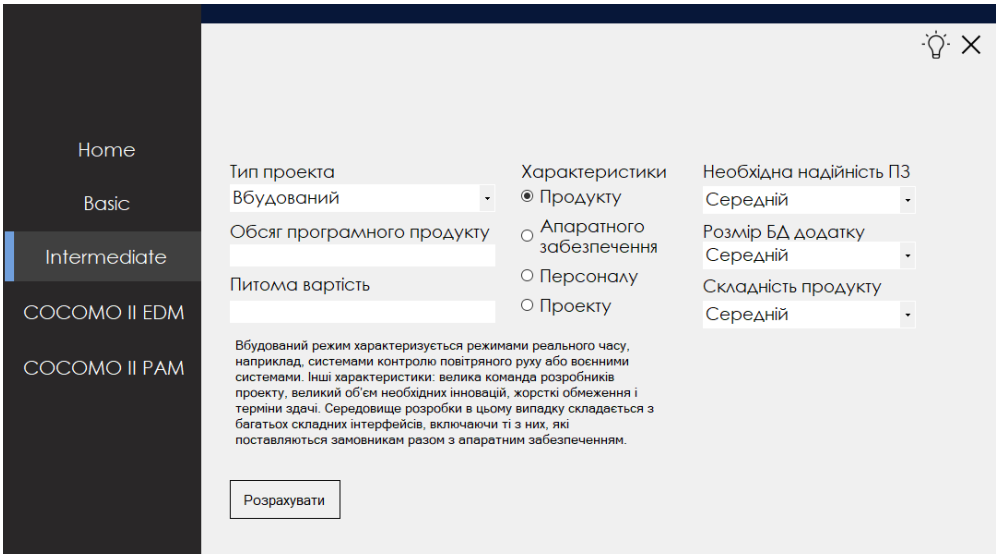

Рисунок 5.4 — Вікно форми COCOMO Intermediate при виборі категорії Характеристики продукту

У категорії **Характеристики апаратного забезпечення** (рисунок 5.5) вибираємо рейтинг для факторів витрат Обмеження швидкодії при виконанні програми; Обмеження пам'яті; Нестійкість оточення віртуальної машини; Необхідний час відновлення;

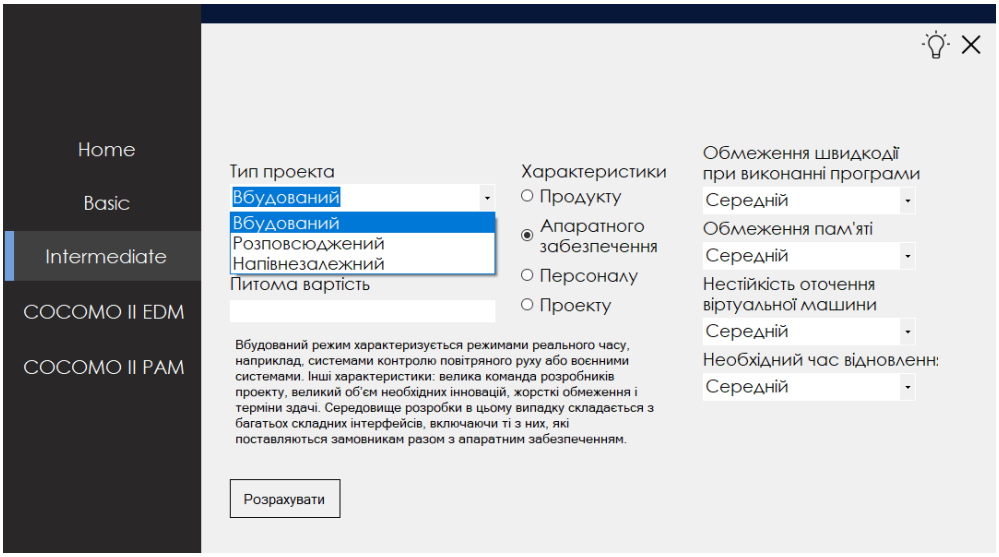

Рисунок 5.5 — Вікно форми COCOMO Intermediate при виборі категорії Характеристики апаратного забезпечення

При перемиканні в категорію **Характеристики персоналу** (рисунок 5.6) вибираємо рейтинг для факторів витрат Аналітичні здібності; Здібності до розробки ПЗ; Досвід розробки; Досвід використання віртуальних машин; Досвід розробки на мовах програмування.

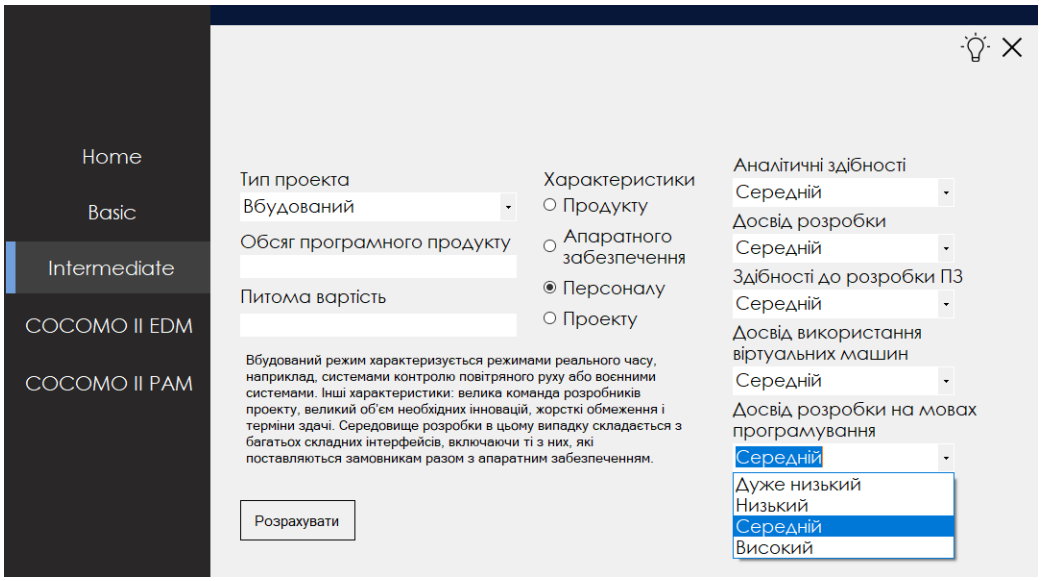

Рисунок 5.6 — Вікно форми COCOMO Intermediate при виборі категорії Характеристики персоналу

У категорії **Характеристики проекту** (рисунок 5.7) вибираємо рейтинг для факторів витрат Використання інструментарію розробки ПО; Застосування методів розробки; Вимоги дотримання графіка розробки.

Після натискання кнопки **Розрахувати** програма видає трудомісткість проекту в людино * місяцях, час розробки в місяцях і вартість розробки в умовних грошових одиницях (рисунок 5.7).

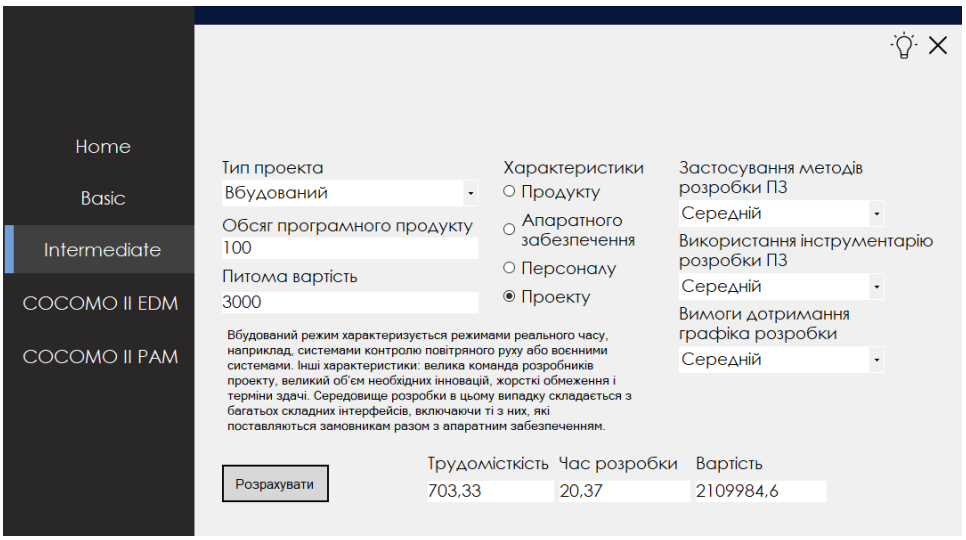

Рисунок 5.7 — Вікно форми COCOMO Intermediate при виборі категорії Характеристики проекту з результатами розрахунку

На вкладці COCOMO II EDM (рисунок 5.8) виконується попередня оцінка проекту на початковій фазі по удосконаленій моделі оцінки вартості програмного забезпечення

З випадаючих списків необхідно вибрати рівні значущості чинників масштабу опис яких наведено в таблиці 2.4. та множники трудомісткості які наведені в підрозділі 2.2.1. (рисунок 5.8).

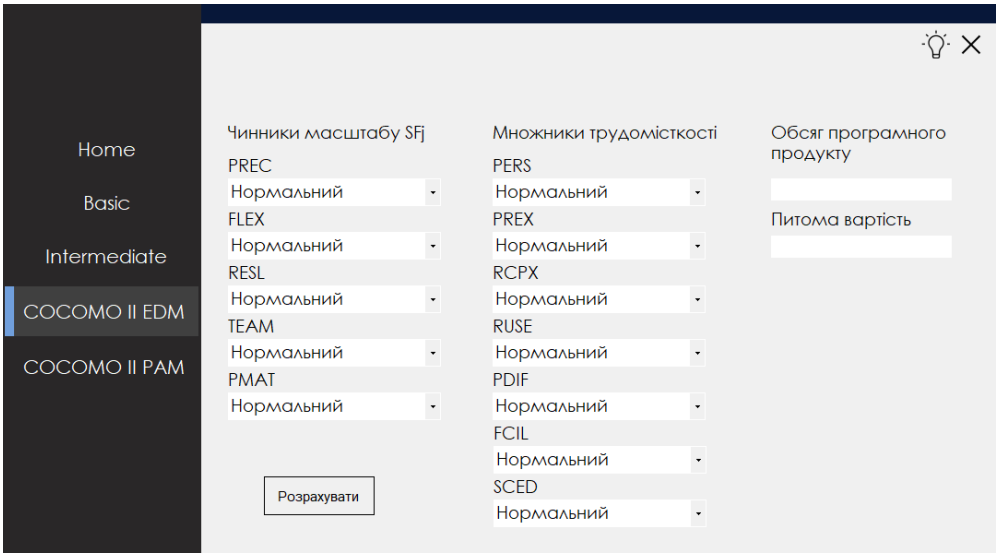

Рисунок 5.8 — Вікно форми COCOMO II EDM

Після натискання на кнопку Розрахувати, буде здійснений розрахунок, програма видає трудомісткість проекту в людино * місяцях, час розробки в місяцях і вартість розробки в умовних грошових одиницях.

На вкладці COCOMO II PAM (рисунок 5.9) здійснюється розрахунок для стадії детальної оцінки після опрацювання архітектури.

З випадаючих списків необхідно вибрати рівні значущості чинників масштабу опис яких наведено в таблиці 2.4. та множники трудомісткості розподілені за категоріями опис яких наведено в підрозділі 2.2.2, в полі введення **Обсяг програмного продукту** задати обсяг програмного коду в тисячах рядків, в полі введення **Питома вартість** в умовних грошових одиницях.

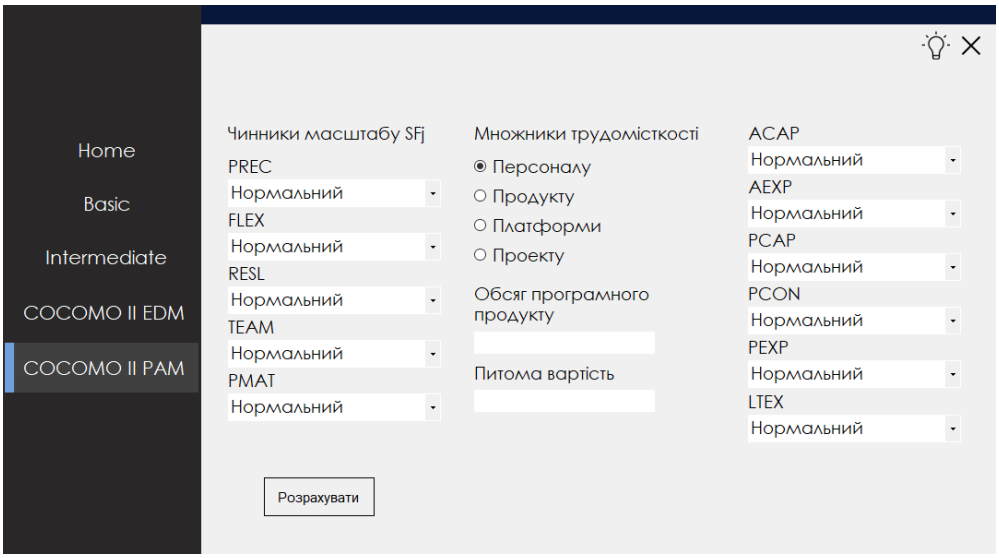

Рисунок 5.9 — Вікно форми COCOMO II PAM

Після натискання на кнопку Розрахувати, буде здійснений розрахунок, програма видає трудомісткість проекту в людино * місяцях, час розробки в місяцях і вартість розробки в умовних грошових одиницях (рисунок 5.10).

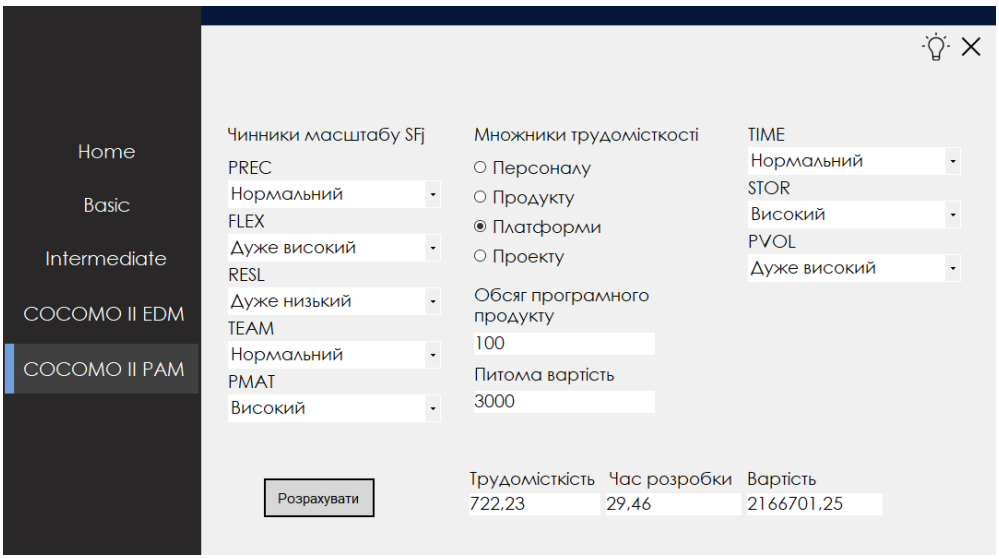

Рисунок 5.10 — Вікно форми COCOMO II PAM з результатами розрахунку

ВИСНОВКИ

Під час розробки дипломного проекту були досліджені і проаналізовані проблеми, які виникають під час оцінки вартості програмного забезпечення. Розглянуті методи та моделі оцінки вартості програмного продукту. Для кожної розглянутої моделі були наведені переваги і недоліки.

Розроблений додаток з оцінки вартості програмного забезпечення з використанням конструктивних моделей вартості (COCOMO і COCOMO II), що найповніше описують процес розробки програмного забезпечення та мають найбільшу кількість необхідних характеристик (факторів).

Розроблена автоматизована система на підставі вищевказаних формул виконує розрахунок вартості програмного забезпечення. В програмі вираховуються трудомісткість, час розробки та вартість ПЗ.

Програмна система реалізована у вигляді десктопного додатку.

Розроблений додаток може бути корисним менеджерам та керівникам проектів для здійснення контролю за розробкою програмних продуктів, так як створення нових систем має супроводжуватись оцінкою вартості їх розробки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Боєм Б. У. Инженерное проектирование программного обеспечения / Б. У. Боєм. — М.: Радио и связь, 1985. — 512 с.
2. Liming W. Cost Estimating Methods [Електронний ресурс] / W. Liming // University of Calgary — Режим доступу до ресурсу: https://www.computing.dcu.ie/~renaat/ca421/LWu1.html.
3. Глазова М. А. Системы оценки стоимости проектов по разработке программного обеспечения [Електронний ресурс] / М. А. Глазова — Режим доступу до ресурсу: http://www.appliedinformatics.ru/general/upload/articles/informatica_15_-12-renamed.pdf.
4. Рябокінь Ю. М. Оцінка вартості програмного забезпечення [Електронний ресурс] / Ю. М. Рябокінь // ВІСНИК КНУТД №1 (82) Серія «Технічні науки» — с. 117-124. — 2015. — Режим доступу до ресурсу: https://er.knutd.edu.ua/bitstream/123456789/149/1/V82_P117-124.pdf.
5. Алиев Х. Р. Алиев Комбинированная модель оценки трудоемкости разработки программного обеспечения [Електронний ресурс] / Х. Р. Алиев // Научно-технические ведомости СПбГПУ с. 201-218. – 2010. – Режим доступу до ресурсу: https://cyberleninka.ru/article/v/kombinirovannaya-model-otsenki-trudoemkosti-razrabotki-programmnogo-obespe-cheniya.
6. Соммервилл И. Инженерия программного обеспечения / Иан Соммервилл. – М.: Издательский дом Вильямс, 2002. – 624 с. – (6-е издание).
7. Хрущ Л. З. Економіка програмного забезпечення / Л. З. Хрущ. – Івано-Франківськ: ЛІК, 2018. – 103 с.
8. МОДЕЛИ, МЕТОДЫ И СРЕДСТВА ОЦЕНКИ СТОИМОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ [Електронний ресурс] / Н. А.Сидоров, Д. В. Баценко, Ю. Н. Василенко, Ю. В. Щебетин – Режим доступу до ресурсу: http://dspace.nbuv.gov.ua/bitstream/handle/123456789/1541/36-Sidorov.pdf.

9. Гагарина Л. Г. Технология разработки программного обеспечения / Л. Г. Гагарина, Е. В. Кокорева, Б. Д. Виснадул. – Москва: ИД ФОРУМ, ИНФРА-М, 2008. – 400 с.
10. Брукс Ф. Мифический человеко-месяц или как создаются программные системы [Електронний ресурс] / Фредерик Брукс – Режим доступу до ресурсу: https://nsu.ru/xmlui/bitstream/handle/nsu/8870/Frederick_Brooks.pdf.
11. Лавріщева К. М. ПРОГРАМНА ІНЖЕНЕРІЯ / К. М. Лавріщева. – Київ, 2008. – 319 с.
12. Ваганова Е. В. Оценка стоимости разработки программного продукта: Обзор [Електронний ресурс] / Е. В. Ваганова, А. А. Земцов, С. Л. Миньков // Проблемі учета и финансов № 1(21). – 2016. – Режим доступу до ресурсу: https://ocenschiki-i-eksperty.ru/files/comment-files/31ad450582d3e041e6b024aeca3e67c1d5a3d88a68a137474aeb583a77dbc0e6/otsenka-stoimosti-razrabotki-programmnogo-produkta-obzor.pdf.
13. Битковский Д. И. Применение модели COCOMO в экономике программной инженерии [Електронний ресурс] / Д. И. Битковский, А. В. Моторко // Экономика и бизнес: теория и практика. – 2017. – Режим доступу до ресурсу: http://economyandbusiness.ru/primenenie-modeli-sosomo-v-ekonomike-programmnoj-inzhenerii.
14. Садовский И. Д. Применение модели COCOMO II для оценки разработки Программного обеспечения в Windows проектах [Електронний ресурс] / И. Д. Садовский // Экономика и бизнес: теория и практика. – 2016. – Режим доступу до ресурсу: http://economyandbusiness.ru/primenenie-modeli-cocomo-ii-dlya-otsenki-razrabotki-programmnogo-obespecheniya-v-windows-proektah.
15. Вендров А. М. Проектирование программного обеспечения экономических информационных систем / А. М. Вендров. – М.: Финансы и статистика, 2005. – 544 с. – (2-е).
16. Visual C# 2008: базовый курс / [К. Уотсон, К. Нейгел, Я. Педерсен та ін.]. – М.: И.Д. Вильямс, 2009. – 1216 с.
17. Сведения о проектах и решениях [Електронний ресурс] – Режим доступу до

ресурсу: https://docs.microsoft.com/ru-ru/visualstudio/get-started/tutorial-projects-solutions?view=vs-2019.

18. Шарп Д. Microsoft Visual C#. Подробное руководство / Дж. Шарп. – СПб: Питер, 2017. – 848 с. – (8-е издание).
19. C# / [Х. Дейтел, П. Дейтел, Д. Листфилд та ін.]. – СПб: БХВ-Петербург, 2006. – 1056 с.
20. Руководство по Entity Framework [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/sharp/entityframework/.

ДОДАТОК 1

Автоматизована система розрахунку вартості програмного забезпечення

СПЕЦИФІКАЦІЯ

УКР.КПІм.ІгоряСікорського_ТЕФ_АПЕПС_ДА3229_19Б

Аркушів 2

Київ 2019

Позначення	Найменування	Примітки
Документація		
УКР.КПІм.ІгоряСікорського_ТЕФ_АПЕПС_ДА 3229_19Б 81-1	Записка.docx	Текстова частина дипломної роботи
Компоненти		
УКР.КПІм.ІгоряСікорського_ТЕФ_АПЕПС_ДА 3229_19Б 12-1	COCOMO_App.exe	Основний компонент додатку розрахунку вартості

ДОДАТОК 2

Автоматизована система розрахунку вартості програмного забезпечення

ТЕКСТ ПРОГРАМНОГО МОДУЛЮ

УКР.КПІм.ІгоряСікорського_ТЕФ_АПЕПС_ДА3229_19Б

Аркушів 8

Київ 2019

Тест програми розрахунку вартості програмного забезпечення

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using COCOMO_Data;
using System.Data.Entity;

namespace COCOMO_Business
{
   public class COCOMO_DataManager
   {
      private COCOMO_TablesEntities COCOMO_Tables = new COCOMO_TablesEntities();

      #region -- ToList()
      public List<COCOMO_Basic> BasicToList()
      {
         return COCOMO_Tables.COCOMO_Basic_Data.ToList();
      }
      public List<COCOMO_Intermediate> IntermediateToList()
      {
         return COCOMO_Tables.COCOMO_Intermediate_Data.ToList();
      }
      public List<COCOMO_Intermediate_Ratings> IntermediateRatingsToList()
      {
         return COCOMO_Tables.COCOMO_Intermediate_Ratings_Data.ToList();
      }
      public List<COCOMO2_ScaleFactors> ScaleFactorsToList()
      {
         return COCOMO_Tables.COCOMO2_ScaleFactors.ToList();
      }
      public List<COCOMO2_MultiplicativeDrivers_EDM> EDMToList()
      {
         return COCOMO_Tables.COCOMO2_MultiplicativeDrivers_EDM_Data.ToList();
      }
      public List<COCOMO2_MultiplicativeDrivers_PAM> PAMToList()
      {
         COCOMO_Tables.COCOMO2_MultiplicativeDrivers_PAM_Data.Load();
         return COCOMO_Tables.COCOMO2_MultiplicativeDrivers_PAM_Data.ToList();
      }
      #endregion

      // Обчислення базового рівня
      public List<double> BasicLevel_Calculation(string type, double size, double u_cost)
      {
         List<COCOMO_Basic> BasicList = new List<COCOMO_Basic>();
         List<double> result = new List<double>();
         double A = 0;
         double B = 0;
         double C = 0;
         double D = 0;
         double PM = 0;// PM (People×Month) – трудомісткість (люд. × міс.);
         double TM = 0;// TM (Time at Month) - час розробки в календарних місяцях;
         double SS = 0;// SS - середня чисельність персоналу;
         double dev_cost = 0;
         double P = 0; // P - продуктивність;

         BasicList = BasicToList();
```

```
    switch (type)
    {
        case "Organic":

            A = BasicList[0].A;
            B = BasicList[0].B;
            C = BasicList[0].C;
            D = BasicList[0].D;

            break;
        case "Semi-detached":

            A = BasicList[1].A;
            B = BasicList[1].B;
            C = BasicList[1].C;
            D = BasicList[1].D;

            break;
        case "Embedded":

            A = BasicList[2].A;
            B = BasicList[2].B;
            C = BasicList[2].C;
            D = BasicList[2].D;

            break;

    }

    PM = A * Math.Pow(size, B);
    TM = C * Math.Pow(PM, D);
    SS = PM / TM;
    P = size / PM;
    dev_cost = PM * u_cost;

    result.Add(PM);
    result.Add(TM);
    result.Add(SS);
    result.Add(dev_cost);
    result.Add(P);

    return result;

}

// Обчислення рівня Intermediate
public List<double> IntermediateLevel_Calculation(string type, List<string> ratings,  double size, double u_cost)
{
    List<COCOMO_Intermediate> IntermediateList = new List<COCOMO_Intermediate>();
    List<COCOMO_Basic> BasicList = new List<COCOMO_Basic>();
    List<COCOMO_Intermediate_Ratings> IntermediateRatingsList = new List<COCOMO_Intermediate_Ratings>();
    List<double> ratingValues = new List<double>();
    List<double> result = new List<double>();
    double A = 0;
    double B = 0;
    double C = 0;
    double D = 0;
    double PM = 0;// PM (People×Month) – трудомісткість (люд. × міс.);
    double TM = 0;// TM (Time at Month) - час розробки в календарних місяцях;
    double EAF = 1;// Effort Adjustment Factor
    double dev_cost = 0;

    IntermediateList = IntermediateToList();
```

```
IntermediateRatingsList = IntermediateRatingsToList();
BasicList = BasicToList();

switch (type)
{
   case "Organic":

      A = IntermediateList[0].A;
      B = IntermediateList[0].B;
      C = BasicList[0].C;
      D = BasicList[0].D;

      break;
   case "Semi-detached":

      A = IntermediateList[1].A;
      B = IntermediateList[1].B;
      C = BasicList[1].C;
      D = BasicList[1].D;

      break;
   case "Embedded":

      A = IntermediateList[2].A;
      B = IntermediateList[2].B;
      C = BasicList[2].C;
      D = BasicList[2].D;

      break;

}

for (int i = 0; i < ratings.Count; i++)
{
   switch (ratings[i])
   {
      case "Very Low":
         ratingValues.Add(Convert.ToDouble(IntermediateRatingsList[i].Very_Low));
         break;
      case "Low":
         ratingValues.Add(Convert.ToDouble(IntermediateRatingsList[i].Low));
         break;
      case "Nominal":
         ratingValues.Add(Convert.ToDouble(IntermediateRatingsList[i].Nominal));
         break;
      case "High":
         ratingValues.Add(Convert.ToDouble(IntermediateRatingsList[i].High));
         break;
      case "Very High":
         ratingValues.Add(Convert.ToDouble(IntermediateRatingsList[i].Very_High));
         break;
      case "Extra High":
         ratingValues.Add(Convert.ToDouble(IntermediateRatingsList[i].Extra_High));
         break;
   }
}

for (int i = 0; i < ratingValues.Count; i++)
{
   EAF *= ratingValues[i];
}

PM = EAF * A * Math.Pow(size, B);
```

```
        TM = C * Math.Pow(PM, D);

        dev_cost = PM * u_cost;

        result.Add(PM);
        result.Add(TM);
        result.Add(dev_cost);
        result.Add(EAF);

        return result;
    }

    // Обчислення COCOMO2 EDM (Early Design Model)

    public List<double> COCOMO2_EDM_Calculation(List<string> scaleFactors, List<string> multiplicativeDrivers,
double size, double u_cost)
    {
        List<COCOMO2_MultiplicativeDrivers_EDM> multiplicativeDriverList = new
List<COCOMO2_MultiplicativeDrivers_EDM>();
        List<double> multiplicativeDriverValues = new List<double>();
        List<double> scaleFactorValues = new List<double>();
        List<double> result = new List<double>();

        double A = 2.94;
        double B = 0.91;
        double C = 3.67;
        double D = 0.28;
        double PM = 0;// PM (People×Month) – трудомісткість (люд. × міс.);
        double PMns = 0;// PM without SCED
        double TM = 0;// TM (Time at Month) - час розробки в календарних місяцях;
        double dev_cost = 0;
        double powerForTM = 0;
        double E = 0;
        double SF_sum = 0;// SF  (Scale Factors) - чинники масштабу
        double EAF = 1;// Effort Adjustment Factor
        double EAF_Without_SCED = 0;
        double SCED = 0;// Effort Adjustment Factor

        multiplicativeDriverList = EDMToList();

        for (int i = 0; i < multiplicativeDrivers.Count; i++)
        {
            switch (multiplicativeDrivers[i])
            {
                case "Extra Low":
                    multiplicativeDriverValues.Add(Convert.ToDouble(multiplicativeDriverList[i].Extra_Low));
                    break;
                case "Very Low":
                    multiplicativeDriverValues.Add(Convert.ToDouble(multiplicativeDriverList[i].Very_Low));
                    break;
                case "Low":
                    multiplicativeDriverValues.Add(Convert.ToDouble(multiplicativeDriverList[i].Low));
                    break;
                case "Nominal":
                    multiplicativeDriverValues.Add(Convert.ToDouble(multiplicativeDriverList[i].Nominal));
                    break;
                case "High":
                    multiplicativeDriverValues.Add(Convert.ToDouble(multiplicativeDriverList[i].High));
                    break;
                case "Very High":
                    multiplicativeDriverValues.Add(Convert.ToDouble(multiplicativeDriverList[i].Very_High));
                    break;
```

```
                case "Extra High":
                    multiplicativeDriverValues.Add(Convert.ToDouble(multiplicativeDriverList[i].Extra_High));
                    break;
            }
        }

        scaleFactorValues = COCOMO2_ScaleFactor_Values(scaleFactors);
        SF_sum = COCOMO2_ScaleFactor_Sum(scaleFactorValues);
        EAF = COCOMO2_EAF_Multiplication(multiplicativeDriverValues);
        SCED = multiplicativeDriverValues[(multiplicativeDriverValues.Count - 1)];
        EAF_Without_SCED = EAF / SCED;

        E = B + 0.01 * SF_sum;

        PM = EAF * A * Math.Pow(size, E);
        PMns = EAF_Without_SCED * A * Math.Pow(size, E);

        powerForTM = D + 0.2 * (E - B);

        TM = SCED * C * Math.Pow(PMns, powerForTM);
        dev_cost = PM * u_cost;

        result.Add(PM);
        result.Add(TM);
        result.Add(dev_cost);
        result.Add(EAF);
        result.Add(EAF_Without_SCED);

        return result;
    }


    // Обчислення COCOMO2 PAM (Post Architecture Model)

    public List<double> COCOMO2_PAM_Calculation(List<string> scaleFactors, List<string> multiplicativeDrivers,
double size, double u_cost)
    {
        List<COCOMO2_MultiplicativeDrivers_PAM> multiplicativeDriverList = new
List<COCOMO2_MultiplicativeDrivers_PAM>();
        List<double> multiplicativeDriverValues = new List<double>();
        List<double> scaleFactorValues = new List<double>();
        List<double> result = new List<double>();

        double A = 2.94;
        double B = 0.91;
        double C = 3.67;
        double D = 0.28;
        double PM = 0;// PM (People×Month) – трудомісткість (люд. × міс.);
        double PMns = 0;// PM without SCED
        double TM = 0;// TM (Time at Month) - час розробки в календарних місяцях;
        double powerForTM = 0;
        double E = 0;
        double SF_sum = 0;// SF  (Scale Factors) - чинники масштабу
        double EAF = 1;// Effort Adjustment Factor
        double EAF_Without_SCED = 0;
        double dev_cost = 0;
        double SCED = 0;// Effort Adjustment Factor

        multiplicativeDriverList = PAMToList();

        for (int i = 0; i < multiplicativeDrivers.Count; i++)
        {
            switch (multiplicativeDrivers[i])
```

```
        {
            case "Very Low":
                multiplicativeDriverValues.Add(Convert.ToDouble(multiplicativeDriverList[i].Very_Low));
                break;
            case "Low":
                multiplicativeDriverValues.Add(Convert.ToDouble(multiplicativeDriverList[i].Low));
                break;
            case "Nominal":
                multiplicativeDriverValues.Add(Convert.ToDouble(multiplicativeDriverList[i].Nominal));
                break;
            case "High":
                multiplicativeDriverValues.Add(Convert.ToDouble(multiplicativeDriverList[i].High));
                break;
            case "Very High":
                multiplicativeDriverValues.Add(Convert.ToDouble(multiplicativeDriverList[i].Very_High));
                break;
            case "Extra High":
                multiplicativeDriverValues.Add(Convert.ToDouble(multiplicativeDriverList[i].Extra_High));
                break;
        }
    }

    scaleFactorValues = COCOMO2_ScaleFactor_Values(scaleFactors);
    SF_sum = COCOMO2_ScaleFactor_Sum(scaleFactorValues);
    EAF = COCOMO2_EAF_Multiplication(multiplicativeDriverValues);
    SCED = multiplicativeDriverValues[(multiplicativeDriverValues.Count - 1)];
    EAF_Without_SCED = EAF / SCED;

    E = B + 0.01 * SF_sum;

    PM = EAF * A * Math.Pow(size, E);
    PMns = EAF_Without_SCED * A * Math.Pow(size, E);

    powerForTM = D + 0.2 * (E - B);

    TM = SCED * C * Math.Pow(PMns, powerForTM);

    dev_cost = PM * u_cost;

    result.Add(PM);
    result.Add(TM);
    result.Add(dev_cost);
    result.Add(EAF);
    result.Add(EAF_Without_SCED);


    return result;
}




private List<double> COCOMO2_ScaleFactor_Values(List<string> scaleFactors)
{
    List<COCOMO2_ScaleFactors> scaleFactorsList = new List<COCOMO2_ScaleFactors>();
    List<double> scaleFactorValues = new List<double>();

    scaleFactorsList = ScaleFactorsToList();

    for (int i = 0; i < scaleFactors.Count; i++)
    {
        switch (scaleFactors[i])
```

```
            {
                case "Very Low":
                    scaleFactorValues.Add(Convert.ToDouble(scaleFactorsList[i].Very_Low));
                    break;
                case "Low":
                    scaleFactorValues.Add(Convert.ToDouble(scaleFactorsList[i].Low));
                    break;
                case "Nominal":
                    scaleFactorValues.Add(Convert.ToDouble(scaleFactorsList[i].Nominal));
                    break;
                case "High":
                    scaleFactorValues.Add(Convert.ToDouble(scaleFactorsList[i].High));
                    break;
                case "Very High":
                    scaleFactorValues.Add(Convert.ToDouble(scaleFactorsList[i].Very_High));
                    break;
                case "Extra High":
                    scaleFactorValues.Add(Convert.ToDouble(scaleFactorsList[i].Extra_High));
                    break;
            }
        }


        return scaleFactorValues;
    }

    private double COCOMO2_ScaleFactor_Sum(List<double> scaleFactorValues)
    {
        double sum = 0;

        for (int i = 0; i < scaleFactorValues.Count; i++)
        {
            sum += scaleFactorValues[i];
        }

        return sum;
    }

    private double COCOMO2_EAF_Multiplication(List<double> multiplicativeDriverValues)
    {
        double EAF = 1;

        for (int i = 0; i < multiplicativeDriverValues.Count; i++)
        {
            EAF *= multiplicativeDriverValues[i];
        }

        return EAF;
    }

  }
}
```

ДОДАТОК 3

Автоматизована система розрахунку вартості програмного забезпечення

ОПИС ПРОГРАМНОГО МОДУЛЮ

УКР.КПІм.ІгоряСікорського_ТЕФ_АПЕПС_ДА3229_19Б

Аркушів 9

Київ 2019

АНОТАЦІЯ

Додаток містить опис системи, що розроблена для розрахунку вартості розробки програмного забезпечення.

В додатку виконуються такі функції:

— введення вхідних даних для обчислень ;

— обчислення;

— виведення результатів обчислень.

Вхідні та вихідні дані отримуються на формі Windows Forms через елементи керування.

Система створена в Visual Studio Community 2015 з використанням мови програмування C#. Графічний інтерфейс користувача розроблений за допомогою API Windows Forms. Для збереження даних використана система управління базами даних (СУБД) Microsoft SQL Server. Для доступу до бази даних та автоматичного створення сутностей та управлінням ними був застосований Entity Framework 6.1.3.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

У цьому додатку описано систему розрахунку вартості програмного забезпечення.

Модулі системи надано в ДОДАТКУ 2.

Розроблений додаток працює в операційних системах Windows7, Windows8, Windows10.

Компоненти необхідні для установки: Microsoft SQL LocalDB, Microsoft SQL Server 2012 Native Client.

Використана мова програмування — C#.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблена система обчислює трудомісткість, час розробки та вартість розробки програмного забезпечення з використанням алгоритмічних моделей оцінки COCOMO та COCOMO II.

Розроблений програмний додаток рекомендований для використання менеджерами та керівниками проектів під час приймання рішень щодо управління розробкою програмного забезпечення.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Програмний додаток розроблений за схемою дворівневої архітектури «Клієнт-Сервер». Доступ до бази даних здійснюється за допомогою Entity Framework.

На початку роботи, після запуску програмного додатку з’являється головне вікно форми. Далі необхідно вибрати рівень моделі COCOMO, за яким необхідно здійснити обчислення. Введені дані передаються до бібліотеки, обчислюються, результати обчислень виводяться на форму.

ВИКОРИСТОВУВАНІ ТЕХІЧНІ ЗАСОБИ

Для забезпеченні роботи створеного додатку було обрано Visual Studio Community 2015 (мова імплементації — C#).

Розроблений додаток працює в операційних системах Windows7, Windows8, Windows10 та потребує встановлення компонентів: Microsoft SQL LocalDB, Microsoft SQL Server 2012 Native Client.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Для запуску розробленого програмного додатку необхідно запустити виконуваний файл COCOMO_App.exe, інсталяція системи не потрібна.

Після запуску програми користувач отримує доступ у головну форму, з якої може починати роботу в додатку та виконувати необхідні обчислення.

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідні дані мають вигляд цілого числа або числа з плаваючою комою.

Вхідні дані для роботи програмного додатку вводяться та зчитуються з textBox та comboBox.

Вихідні дані мають вигляд числа з плаваючою комою та виводяться в textBox для результатів обчислень.

Вихідні дані, які отримуються в результаті роботи програми є трудомісткість, час розробки та вартість розробки програмного забезпечення.

ДОДАТОК 4

Автоматизована система розрахунку вартості програмного забезпечення

АКТ ВПРОВАДЖЕННЯ

УКР.КПІм.ІгоряСікорського_ТЕФ_АПЕПС_ДА3229_19Б

Аркушів 2

Київ 2019

«Затверджую»
Директор
ТОВ «ВЕБ.ЮЕЙ»
Кузьмін І.І.

АКТ ВПРОВАДЖЕННЯ

результатів дипломної роботи бакалавра Кисельова Ю.Ю.
на тему «Автоматизована система розрахунку вартості програмного забезпечення» яка виконана в Національному технічному університеті України «Київський політехнічний інститут ім. Ігоря Сікорського»
«_____» _____ 2019 р.

Нами, представниками кафедри автоматизації проектування енергетичних процесів і систем НТУУ «Київський політехнічний інститут ім. Ігоря Сікорського» та ТОВ «ВЕБ.ЮЕЙ», складено даний акт про те, що для використання в роботі при прийнятті рішень щодо розробки нових проектів при визначенні вартості програмного забезпечення, що розробляється фахівцями ТОВ «ВЕБ.ЮЕЙ», прийнято результати дипломної роботи бакалавра Кисельова Ю.Ю., а саме: програмне забезпечення — автоматизована система розрахунку вартості програмного забезпечення у вигляді запису на CD; документація програмного супроводу.

Представник кафедри АПЕПС
НТУУ «КПІ ім. Ігоря Сікорського» _____

Керівник дипломної роботи Сегеда І. В. _____

Директор ТОВ «ВЕБ.ЮЕЙ» Кузьмін І.І. _____