Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
магістр
(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ГРУПОВИХ ЕКСПЕРТНИХ МЕТОДІВ ПРИЙНЯТТЯ РІШЕНЬ ДЛЯ ОЦІНКИ ЯКОСТІ ПРОГРАМНИХ ІНТЕРФЕЙСІВ
RESEARCH AND SOFTWARE IMPLEMENTATION OF GROUP EXPERT DECISION-MAKING METHODS FOR QUALITY ASSESSMENT OF SOFTWARE INTERFACES

Виконав: студент(ка) 2 курсу, групи КНТ-129м
Спеціальності 121 Інженерія програмного забезпечення
(код і найменування спеціальності)

Освітня програма (спеціалізація) Інженерія програмного забезпечення

Коломоєць В.М.
(прізвище та ініціали)
Керівник Степаненко О.О.
(прізвище та ініціали)
Рецензент Гофман Є.О.
(прізвище та ініціали)

2020

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
_____С.О. Субботін
“____” _____2020 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Коломойця Вадима Михайловича
(прізвище, ім’я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація групових експертних методів прийняття рішень для оцінки якості програмних інтерфейсів. Research and Software Implementation of Group Expert Decision-Making Methods for Quality Assessment of Software Interfaces
керівник проєкту (роботи) Степаненко Олександр Олексійович, к.т.н., доцент ,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від “03” листопада 2020 року № 301
2. Строк подання студентом проєкту (роботи) 5 грудня 2020 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Основні рішення щодо реалізації компонентів системи. 4. Експлуатація, тестування та експериментальне дослідження програми. 5. Економіко-організаційна частина. 6. Охорона праці та безпека в надзвичайних ситуаціях. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Степаненко О.О., доцент		
5	Пожуєва Т.О. проф.		
6	Коробко О. В., ст., викладач		
Нормоконтроль	Калініна М.В., ассист.		

7. Дата видачі завдання “ 29 ” лютого 2020 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2-3 тижні	Розділ 1
3	Розробка та удосконалення методів, моделей й алгоритмів вирішення задачі.	4-5 тижні	Розділ 2
4	Розробка архітектури програми.	6 тиждень	Розділи 2, 3
5	Розробка програми.	7-8 тижні	Розділ 3
6	Тестування та експериментальне дослідження програмного забезпечення.	9 тиждень	Розділ 4
7	Розробка організаційно-економічної частини	10 тиждень	Розділ 5
8	Розробка заходів з охорони праці та цивільної безпеки.	11 тиждень	Розділ 6
9	Оформлення пояснювальної записки та документів до неї.	12-13 тижні	
10	Нормоконтроль та рецензування.	14-15 тижні	
11	Захист роботи.	16 тиждень	

Студент(ка) _____ Коломоєць В.М.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи) _____ Степаненко О.О.

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра: 144 с., 17 табл., 20 рис., 3 дод., 57 джерел.

АНАЛІЗ ІЄРАРХІЙ, ПРИЙНЯТТЯ РІШЕНЬ, ІНТЕРФЕЙС КОРИСТУВАЧА, ШАБЛОН, ПРОЄКТУВАННЯ, МЕТРИКИ, СТАНДАРТ, ЯКІСТЬ.

Об’єкт дослідження – процес проєктування програмних інтерфейсів.

Предмет дослідження – якість проєктування програмних інтерфейсів.

Мета роботи – підвищити якість проєктування інтерфейсів користувача шляхом застосування експертних методів прийняття рішень на етапі їх проєктування, автоматизації роботи проєктувальника при виборі альтернативних варіантів проєктних рішень.

Матеріали, методи та технічні засоби: методи прийняття рішень; структурне та об’єктно-орієнтоване програмування; стандарти якості программних засобів, середовище розробки Microsoft Visual Studio 2015; мова програмування C#; операційна система Windows 7 або вище.

Результати. Розроблено метод оцінки якості проєктування програмних інтерфейсів, який грунтується на відомому методі аналіза ієрархій Т. Сааті та комплекс нових моделей програмного забезпечення, що автоматизує цей метод.

Висновки. Використання методу оцінки якості проєктування програмних інтерфейсів забезпечує підвищення якості проєктування цих інтерфейсів. Економічна ефективність використання засобів автоматизації полягає у заощаджені до 29193 гривень на рік при терміні окупності в 1,89 року.

Галузь використання – проєктування інтерфейсів користувача.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 144 pages, 17 tables, 20 figures, 3 appendixes, 57 sources.

HIERARCHY ANALYSIS, DECISION MAKING, INTERFACE, PATTERN, DESIGN, METRICS, STANDARD, QUALITY.

Object of study is a processes of of software interfaces development.

Subject of study are quality of software interface design.

The purpose of the work is to improve the quality of software interface design by applying expert decision-making methods at the design stage, automating the designer's work when choosing alternative design solutions.

Materials, methods, and tools: decision-making methods; structural and object-oriented programming; software quality standards, Microsoft Visual Studio 2015 development environment; C # programming language; operating system Windows 7 or higher.

Results. A method for assessing the quality of software interface design has been developed, which is a further development of the well-known T. Saati's method of analysis of hierarchies and a set of new software models that automate this method.

The implementation of the method for assessing the design quality of software interfaces improves the design quality of these interfaces. The economic efficiency of using automation means saved up to 29193 hryvnia per year with a payback period of 1,89 years.

The field of use is a software interface design.

ЗМІСТ

С.

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

АНР – Analytic Hierarchy Process;

ГЕО – групове експертне оцінювання;

ГГц – гігагерц;

ЕОМ – електронно-обчислювальна машина;

Кб – кілобайт;

МАІ – метод аналізу ієрархій;

Мб – мегабайт;

МГц – мегагерц;

МПП – матриця попарних порівнянь;

НВП – нормалізований вектор пріоритетів;

ОПР – особа що приймає рішення;

ОС – операційна система;

ПК – персональний комп’ютер;

ПП – програмний продукт;

ПРЗ – програмний засіб;

СППР – система підтримки прийняття рішень;

ТЗ – технічне завдання;

ЯПЗ – якість програмних засобів.

ВСТУП

Основна маса сучасних завдань з проєктування та дизайну інтерфейсів користувача є проєктами зі створення застосунків на базі однієї з програмних платформ. Найбільш затребувані платформи – мобільні та настільні операційні системи (ОС), та веб (браузери). В останні роки модель сторонніх застосунків найшла розповсюдження і в інших сферах, які раніше були закритими – «розумні» телевізори, автомобілі, побутова техніка [1]. Тому простір для роботи сучасного проєктувальника і дизайнера інтерфейсів активно розширюється як за рахунок створення нових продуктів, так і за рахунок перенесення продукту на нові платформи. Ця тенденція призводить до пошуку нових наукоємних і високотехнологічних підходів для створення інтерфейсів.

В області розробки призначених для користувача інтерфейсів існує ряд проблем, таких як трудомісткість розробки, проблема семантичного розриву, проблема адаптивності інтерфейсу до зовнішніх умов, проблема зручності використання інтерфейсу. З іншого боку, існує ряд підходів до побудови інтерфейсу, які намагаються вирішити ту чи іншу проблему [2].

Проєктувальника інтерфейсів, з точки зору теорії систем та системного аналізу, можна назвати особою що приймає рішення (ОПР), яка у своїй діяльності неодноразово стикається з необхідністю побудови варіантів досягнення мети (множини альтернатив) проєктування та вибору з них оптимального. Від цього безпосередньо залежить і якість розробленого програмного інтерфейсу.

На даний момент немає загальновизнаного підходу до проєктування орієнтованих на людину інтерфейсів та оцінки їх якості. Існуючі стандарти щодо людино-машинної взаємодії та орієнтованого на людину проєктування, надають лише загальні рекомендації, а не готові проєктні рішення [3].

Один з базових підходів ефективного людино-орієнтованого проєктування інтерактивних систем – це використання патернів (шаблонів) [4]-[5] на етапі «Розробка проєктних рішень» при розробці програмних інтерфейсів

користувача, якому передують етапи «Розуміння і визначення умов використання» та «Визначення вимог користувачів» [6].

Виходячи з цього актуальним є дослідження та програмна реалізація інформаційної технології групового експертного оцінювання (ГЕО) для оцінки якості програмних інтерфейсів, що дозволило б обгрунтовано, но базі кількісних показників, вирішити задачу підвищення якості проєктування інтерфейсів користувача. Саме ГЕО надає можливість збільшити об’єктивність прийняття групових рішень та зменшити суб’єктивність міркувань експертів.

Інформаційні технології багатокритеріального ГЕО, що використовуються у колективних СППР, часто є роз'єднаними, тобто можуть бути виконані відокремлено, у вигляді незв'язаних компонент.

Відомо багато різних способів синтезу моделей ГЕО, що базуються на підході аналізу ієрархій: способів формування МПП; способів агрегування експертних оцінок; способів синтезу підсумкового рішення; обраних шкал для синтезу відображення експертних оцінок тощо.

Значну роль у формуванні та розвитку методів та систем підтримки прийняття рішень на основі аналізу ієрархій відіграють роботи зарубіжних і вітчизняних авторів T. L. Saaty, E. Triantaphyllou, J. P. Brans, М. З. Згуровського, Н. Д. Панкратової, Н. І. Недашківської та ін. [7]-[11].

Відомі також підходи для кількісної оцінки ряду метрик якості програмних засобів (ЯПЗ), що описані в серії міжнародних стандартів ISO/IEC 9126 «Software Engineering. Product quality», а також оцінки часу виконання завдань людиною з знаряддям праці у вигляді програмного інтерфейсу – метод GOMS і GOMS - сімейство (KLM-GOMS, CMN-GOMS, NGOMSL CPM-GOMS [12]. Однак їх неможливо застосувати для комплексної оцінки якості інтерфейсу, що розробляється за кількома метриками одночасно.

Об’єкт дослідження даної випускної кваліфікаційної роботи магістра – процес проєктування програмних інтерфейсів.

Предмет дослідження – якість проєктування програмних інтерфейсів.

Мета роботи – підвищити якість проєктування інтерфейсів користувача шляхом застосування групових експертних методів прийняття рішень на етапі їх проєктування, автоматизації роботи проєктувальника при виборі альтернативних варіантів проєктних рішень.

Для досягнення поставленої мети у роботі необхідно вирішити такі завдання:

– розглянути класифікацію задач прийняття рішень;

– дослідити існуючі методи групового експертного оцінювання та обрати необхідний для забезпечення ГЕО якості програмних інтерфейсів;

– провести аналіз моделей ЯПЗ та обрати метрики, необхідні для кількісної оцінки якості програмних інтерфейсів;

– розробити моделі програмного забезпечення для ГЕО якості програмних інтерфейсів;

– розробити програмне та інформаційне забезпечення компонентів системи, що автоматизує роботи проєктувальника;

– валідація розробленого програмного забезпечення на прикладі ГЕО вибору шаблонів компоновки екранних форм програмних інтерфейсів за критеріями якості.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Класифікація задач та методів прийняття рішень

У повсякденному житті ми стикаємося з величезною кількістю різних задач прийняття рішень. З теорії систем відомо, що будь-який об'єкт можна класифікувати за різними ознаками. В даному випадку, наприклад, за кількістю та релевантності вихідної для прийняття рішення інформації.

Задачу прийняття рішень Z представимо кортежем, що наведений у формулі (1.1) [13]:

$$Z = <T, A, K, X, R, E, W>, \quad (1.1)$$

де T – постановка задачі прийняття рішень;

A – множина допустимих альтернативних варіантів;

K – множина критеріїв вибору;

X – множина методів оцінювання переваг;

R – відображення множини можливих альтернатив у множину критеріальних оцінок;

E – система переваг експерта;

W – вирішуюче правило, що описує E.

У якості класифікаційної ознаки задач може бути обраний будь-який з наведених вище елементів кортежу.

Відомі наступні класифікації:

– по виду відображення R. Відображення множини A і K може мати однозначний характер, стохастичний або невизначений вид. У зв’язку з цим серед множини задач прийняття рішень виділяють задачі в умовах ризику та в умовах невизначеності;

– потужність множини K. Кількість цих критеріїв може бути від одного елементу до кількох. У зв’язку з цим розрізняють задачі прийняття

рішень зі скалярним та з векторним критеріями (у цьому випадку прийняття рішень вважається багатокритеріальним);

– тип системи переваг *E*. Вподобання може формувати як одна людина, так і група. Відповідно, множину завдань прийняття рішень можна класифікувати на завдання індивідуального та завдання колективного прийняття рішень.

До класу задач прийняття рішень в детермінованих умовах відносяться задачі, для вирішення яких є релевантна та достовірна інформація. У таких випадках прийнято використовувати методи математичного програмування, основна ідея яких – це пошук оптимальних рішень на базі математичної моделі об'єкта. Методи математичного програмування прийнято використовувати у наступних випадках [14]:

– є якась уніфікована цільова функція, що дозволяє оцінити альтернативні варіанти;

– для вихідної задачі вже існує формальний опис у вигляді математичної моделі;

– існує можливість розрахунку кількісних параметрів цільової функції;

– задача може бути оптимізована шляхом зміни (у деяких межах) параметрів функціонування системи.

Існують варіанти, коли вихідні результати можна описати за допомогою певного розподілу ймовірностей. У такому разі ми маємо справу з прийняттям рішень в умовах ризику. Для формалізації вихідних даних у вигляді розподілу ймовірностей необхідно видаляти експертні знання, або організовувати збір статистичних даних. Традиційно у даному випадку застосовуються методи теорії одновимірної або багатовимірної корисності. Такі завдання розташовуються на межі завдань прийняття рішень в детермінованих умовах та стохастичних.

Завдання невизначеності виникають, коли вихідна для прийняття рішень інформація, характеризується неповнотою, нечіткістю, не є кількісною, а

система, що досліджується описується за допомогою складних математичних моделей. У цьому випадку, традиційно для вирішення проблеми прийняття рішень використовують експертні знання. Але, для вирішення проблем прийняття рішень знання експертів описуються у вигляді кількісних даних, які мають назву "уподобання". У цьому полягає відміна від підходу, прийнятого в інтелектуальних системах.

Важливо відзначити, що однією з умов існування задачі прийняття рішень є наявність хоча б двох альтернатив, з яких вибирають оптимальну. Для випадку наявності тільки одної альтернативи - задача прийняття рішень відсутня.

Задача прийняття рішень називається тривіальною, якщо вона характеризується тільки одним критерієм K та всім альтернативам A_i приписані конкретні числові оцінки відповідно до значень зазначеного критерію.

Задача прийняття рішень не вважається тривіальною навіть при одному критерію K, якщо кожній альтернативі A_i відповідає не точна оцінка, а інтервал можливих оцінок або розподіл $f(K/A_i)$ на значеннях зазначеного критерію.

Нетривіальною вважається задача при наявності декількох критеріїв прийняття рішень незалежно від виду відображення множини альтернатив у множину критеріальних оцінок їх наслідків.

Отже, при наявності ситуації вибору, багатокритеріальності та здійсненні вибору в умовах невизначеності або ризику задача прийняття рішень є нетривіальною [14]. Таким чином, задачі, що вирішуються проєктувальником програмних інтерфейсів слід віднести до нетривіальних.

Існує різна класифікація методів прийняття рішень, залежно від класифікаційних критеріїв. Класифікація методів за критеріями вмісту та типу отримуємої експертної інформації наведена в табл. 1.1 [14].

Таким чином, використаний принцип класифікації дозволив визначити чотири групи методів. Перші три групи (№1, №2 і №4 з табл.1.1) можна віднести до прийняття рішень в умовах визначеності, а четверта (№3 з табл.1.1) – до прийняття рішень в умовах невизначеності.

Таблиця 1.1 – Класифікація методів прийняття рішень

№ п/п	Вміст інформації	Тип інформації	Метод прийняття рішень
1	2	3	4
1	Експертна інформація не обов'язкова		Метод домінування [15] Метод глобальних критеріїв [16]
2	Інформація про переваги на множині критеріїв	Якісна інформація Кількісна оцінка критеріїв Кількісна інформація про заміщення	Лексикографічне впорядкування [15] Порівняння різниць критеріальних оцінок [17] Метод «пріпасовиванія» [18] Методи "ефективність-вартість "[18] Методи згортки на ієрархії [19] Методи порогів [18] Методи теорії цінності [18]
3	Інформація про переваги на множині критеріїв і про наслідки альтернатив	Відсутність інформації про переваги; кількісна та/або інтервальна інформація про наслідки. Якісна інформація про переваги і кількісна про наслідки. Якісна (порядкова) інформація про переваги та наслідки. Кількісна інформація про переваги і наслідки	Методи з дискретизацією невизначеності [16]. Стохастичне домінування [22]. Методи прийняття рішень в умовах ризику та невизначеності, що грунтуються на глобальних критеріях [22] Метод аналізу ієрархій [23] Методи теорії нечітких множин [13] Метод практичного прийняття рішень [18]. Методи вибору статистично ненадійних рішень [22]. Методи дерев рішень [22].

Продовження таблиці 1.1

1	2	3	4
4	Інформація про перевагу альтернатив	Оцінка переважності парних порівнянь	Методи математичного програмування [20]. Лінійна та нелінійна згортка при інтерактивному способі визначення її параметрів [21]

З множини відомих методів і підходів до прийняття рішень найбільший інтерес для оцінки якості програмних інтерфейсів представляють ті, які дають можливість враховувати багатокритеріальність і невизначеність, а також дозволяють здійснювати вибір рішень з множини альтернатив різного типу при наявності критеріїв, що мають різні типи шкал вимірювання (ці методи відносяться до третьої групи).

У свою чергу, серед методів, що утворюють третю групу, найбільш перспективними є декомпозіційні методи теорії очікуваної корисності, методи аналізу ієрархій і теорії нечітких множин.

Даний вибір визначений тим, що ці методи в найбільшій мірі задовольняють вимогам універсальності, врахування багатокритеріальності вибору в умовах невизначеності з дискретної або безперервної множини альтернатив, простоти підготовки і переробки експертної інформації [14].

Охарактеризувати досить повно всі методи експертного оцінювання, що стосуються способів експертного оцінювання в умовах невизначеності, в рамках даної роботи магістра неможливо, тому в подальшому розглядаються тільки два підходи в умовах невизначеності, які отримали найбільш широке застосування в колективних системах підтримки прийняття рішень (СППР), а саме: підходи, засновані на методах аналізу ієрархій (MAI) та ранжування альтернатив.

1.2 Аналіз методів оцінювання альтернатив

Серед методів, що найкраще враховують специфіку процесу ГЕО та найбільш перспективними є методи аналізу ієрархій, методи ранжування альтернатив (методи розробки індексів попарного порівняння альтернатив [24], методи визначення переваг однієї альтернативи перед іншою [25]), метод упорядкованого переваги через схожість з ідеальним рішенням [26]).

Виконаємо порівняльний аналіз існуючих методів оцінювання альтернатив. Існуючі методи оцінювання альтернатив в умовах невизначеності відрізняються механізмом своєї реалізації і характером проблем, для вирішення яких вони призначені. Отже, потрібно вибрати метод, найбільш адаптований до задачі ГЕО.

Одним з важливих критеріїв, що впливають на вибір методу, є можливість застосування методу для задач ГЕО. Іншими, не менш важливими критеріями, є відповідність інтуїтивним уявленням вирішення проблеми, наявність етапу декомпозиції проблеми на складові, наявність засобів перевірки узгодженості оцінок експертів. MAI задовольняє даним критеріям.

Розглянемо методи ранжування альтернатив.

Відомі методи ELECTRE (методи розробки індексів попарного порівняння альтернатив) [27], TOPSIS [28] (метод упорядкованого переваги через схожість з ідеальним рішенням), PROMETHEE [29] (метод визначення переваги однієї альтернативи перед іншою) дозволяють визначати переваги альтернатив, охарактеризованих певним числом критеріїв, без побудови узагальненого критерію.

У методі ELECTRE розроблена процедура багатокритеріального вибору найбільш бажаних об'єктів, що включає наступні етапи:

– для кожного з критеріїв вводиться дискретна шкала можливих значень цього критерію, вагові коефіцієнти критеріїв;

– для кожного з критеріїв будується граф, вершинами якого є окремі об'єкти множини, а дуги вказують на відношення домінування між об'єктами відповідно до даного критерію;

– з урахуванням важливості критеріїв та перевагу об'єктів обчислюються матриці значень спеціальних коефіцієнтів, так званих індекси згоди і незгоди;

– для кожної пари об'єктів $(x, y) \in X$ вважається встановленим відношення переваги, скажімо x над y, якщо значення відповідного індексу згоди більше деякого порогового значення, а індекс незгоди – менше відповідного порогового значення;

– будується узагальнений граф переваги, структура якого залежить від обраних граничних значень.

Основна ідея сімейства методів TOPSIS досить проста: після визначення «ідеального» і «ідеально-негативного» очікуваного стану проводиться спроба пошуку такого рішення, яке б дозволяло максимально наблизитися до «ідеального» станом і залишатися максимально віддаленим від «ідеально-негативного». Тобто, методи TOPSIS передбачають розв’язання багатокритеріальної задачі оптимізації у такій постановці [30].

Нехай є множина з A альтернативних варіантів деякої системи, кожний з яких характеризується множиною з K показників, за якими оцінюється її функціонування. Окрім того, є множина вагових коефіцієнтів $w_j (j = 1, ..., K)$, елементи якої характеризують важливість кожного з показників, а також для кожного показника визначений критерії його оптимізації (на максимум або мінімум). За такими даними необхідно побудувати пріоритетний ряд наявних альтернативних варіантів відповідно до ступеня їх відносної переваги. Отже, метод TOPSIS орієнтований на оцінювання альтернативи відносно найкращої та найгіршої точок. Крім того, рішення задачі оптимізації з використанням TOPSIS передбачає необхідність перекладу значень якісних лінгвістичних змінних, що виражають ступінь задоволення тієї або іншої альтернативи критеріям, в нечіткі числа.

Таким чином, методи, що розглянуті вище, у випадку, коли необхідно проаналізувати проблему, в якій критерії характеризуються підкритеріям, не можуть бути застосовані, тому що відсутній етап декомпозиції проблеми на складові.

У методах ELECTRE, PROMETHEE, TOPSIS передбачається, що вподобання експертів формуються при аналізі проблеми. Можливі уточнення введених оцінок здійснюються на етапі перевірки чутливості, тобто на заключному етапі застосування методу. На відміну від них MAI дозволяє проводити узгодження, перегляд і корекцію оцінок на етапі формування матриць попарних порівнянь (МПП), тобто під час аналізу проблеми.

Методи ELECTRE і PROMETHEE не надають можливість урахування різних точок зору, тобто не підтримують рішення задач ГЕО (дослідження показують, що проблема створення системи індексів, які гарантують задані бажані властивості методів, близька до проблеми побудови правил колективного вибору).

Порівняльний аналіз методів показує, що MAI дозволяє забезпечити найбільш узгоджену колективну оцінку, отриману в результаті участі різних фахівців (експертів) або осіб, зацікавлених у вирішенні проблеми [31].

Відзначимо, що ще одним з важливих критеріїв вибору даного методу, є побудова задачі ГЕО у вигляді ієрархії – така форма графічного представлення інформації відповідає ментальним образним уявленням вирішення проблеми, а наявність етапу декомпозиції проблеми на складові дозволяє деталізувати аспекти, що впливають на підсумкове прийняття групового рішення. У традиційному MAI використовується такий тип ієрархії як холлархія – простий вид ієрархії, в якій припустимі зв'язки «багато до багатьох», що обмежує побудову множини задач ГЕО.

Для досягнення мети даної кваліфікаційної роботи магістра проведемо аналіз існуючих модифікацій МАІ, розглянемо їх класифікацію та структуризацію.

1.3 Огляд методу аналізу ієрархій та його модифікацій

Необхідно відзначити, що для оцінки якості програмних інтерфейсів стандартами ISO рекомендується цілий ряд метрик якості, які неможливо розрахувати кількісно. Прикладом таких метрик можуть бути наступні атрибути характеристики «Зручність використання (usability) або практичність» [3], [32]:

– зрозумілість (understandability)– показник, обернений до зусиль, що витрачені користувачами на розуміння логічної концепції цього програмного засобу (ПРЗ);

– опановність (learnability) – показник, обернений зусиллям, що витрачається користувачами на опанування правил застосування ПРЗ;

– керованість (operability) – показник, обернений зусиллям, що використовують користувачі безпосередньо на експлуатацію й керування функціонуванням ПРЗ;

– привабливість (attractiveness) – здатність ПРЗ бути привабливим для користувачів.

У випадках прийняття рішень в ситуаціях, коли, наприклад, для ідей, почуттів, емоцій визначаються деякі кількісні показники, що забезпечують числову шкалу переваг для можливих альтернативних рішень – доцільно використання методу аналізу ієрархій Т. Л. Сааті [23].

Проведемо огляд МАІ для виявлення переваг та недоліків цього підходу.

МАІ включає наступні етапи [23]:

а) формулювання задачі та визначення мети плану;

б) побудова ієрархії мета → критерії → альтернативи;

в) побудова множини МПП (критеріїв, альтернатив і т.п.), уточнення шкали порівняння;

г) обчислення векторів пріоритетів, індексів узгодженості і відносин узгодженості. Повторення пп. в), г) методу для всіх рівнів ієрархії;

д) ієрархічний синтез всієї ієрархії.

Слід зазначити, що класичний МАІ має такі недоліки [31]:

– нездатність адекватно представляти неточність і невизначеність, пов'язані з представленням суджень експертів у вигляді точкових оцінок [33];

– складність алгоритмів обробки великих обсягів числових значень [34] (різке збільшення кількості оцінок зі збільшенням набору елементів, багаторазова процедура попарних порівнянь, процедура коригування оцінок в разі їх неузгодженості);

– обмеження кількості одночасно порівнюваних об'єктів [14] (не рекомендується більше 9), що пов'язано з встановленим психологами фактом, що звичайній людині важко здійснювати раціональний вибір, якщо число об'єктів вибору перевищує 7 ± 2 (магічне число Міллера);

– виникнення реверсу рангів [34], тобто зміна рангів альтернативних рішень при додаванні нових альтернатив або видаленні існуючих, причина появи реверсу рангів полягає в використанні лінійної згортки критеріїв;

– відсутність загальних правил для формування структури моделі ухвалення рішення [14];

– формування структури моделі ГЕО є трудомістким процесом [7], [14].

Отже, такий підхід не дозволяє вирішувати задачі ГЕО в умовах наявності великої кількості порівнюваних об'єктів (альтернатив, критеріїв).

В даний час існує багато модифікацій МАІ. В результаті проведеного дослідження існуючих модифікацій МАІ [31] були виділені наступні ознаки, за якими можна класифікувати ці методи:

– за способом вимірювання переваг експертів (використання різних шкал вимірювання);

– за типом системи вподобань експертів;

– за типом ієрархій;

– за методом формування МПП;

– за способом оцінки і коригування експертних суджень;

– за методом формування підсумкового рішення;

– за структурою методу.

Опис існуючих модифікацій МАІ наведено в табл. 1.2 [31].

Розглянуті модифіковані методи МАІ за ознакою «тип системи вподобань експертів» можна розділити на два класи. Вподобання можуть формуватися або однією особою, або колективом, отже класи МАІ:

– для задач індивідуального прийняття рішень;

– для задач колективного прийняття рішень.

Проведений аналіз модифікацій МАІ показав, що велика частина методів призначена для вирішення задач індивідуального прийняття рішень, які можна адаптувати для вирішення задач ГЕО.

Для вирішення задачі багатокритеріального ГЕО складної ієрархічної структури застосовують такі модифікації МАІ в залежності від типу ієрархій [31]:

– на основі ієрархії з однаковою кількістю та функціональним складом альтернатив під критеріями (ієрархії першого типу);

– на основі ієрархії з різною кількістю та функціональним складом альтернатив під критеріями (ієрархії другого типу);

– на основі ієрархії зі зворотним зв'язком (третій тип ієрархій).

МАІ на основі ієрархії першого типу дозволяють отримати підсумкове рішення для більш детальної ієрархії (наприклад, коли в ієрархії присутні критерії нижніх рівнів – підкритерії). МАІ на основі ієрархії другого типу надають можливість отримання підсумкового рішення для ієрархії з різним числом та функціональним складом альтернатив підкритеріями, для ієрархій, що складаються з декількох гілок, що дозволяє вирішувати практичні задачі складної ієрархічної структури.

Таблиця 1.2 – Модифіковані методи аналізу ієрархій

Назва методу	Ідентифікатор методу	Тип ієрархії	Спосіб вимірювань переваг експертів	Метод формування МПП	Метод формування вектору пріоритетів	Методи оцінки узгодженості МПП	Методи формування підсумкового рішення
1	2	3	4	5	6	7	8
Класичний MAI	ahp_1	перший	точкова оцінка	метод Сааті	для точкових МПП	перегляд оцінок в МПП; точкові; одноосібний перегляд значень оцінок самим експертом	лінійна згортка; ієрархічний синтез; точкові оцінки
MAI, що використовує метод порівняння об'єктів щодо стандартів	ahp_2	перший	точкова оцінка	метод парних порівнянь щодо стандартів	для точкових МПП	перегляд оцінок в МПП; точкові; одноосібний перегляд значень оцінок самим експертом	лінійна згортка; ієрархічний синтез; точкові оцінки

Продовження таблиці 2.1

1	2	3	4	5	6	7	8
MAI, що використовує метод порівняння об'єктів копіюванням	ahp_3	перший	точкова оцінка	метод парних порівнянь об'єктів копіюванням	для точкових МПП	перегляд оцінок в МПП; точкові; одноосібний перегляд значень оцінок самим експертом	лінійна згортка; ієрархічний синтез; точкові оцінки
MAI на основі ієрархії з різним числом та функціональним складом альтернатив під критеріями	ahp_4	другий	точкова оцінка	метод парних порівнянь об'єктів копіюванням	для точкових МПП	перегляд оцінок в МПП; точкові; одноосібний перегляд значень оцінок самим експертом	лінійна згортка; ієрархічний синтез; точкові оцінки
Метод аналізу мереж	ahp_5	третій	точкова оцінка	метод Сааті	для точкових МПП	–	синтез для мережевої моделі; точкові оцінки

Продовження таблиці 2.1

1	2	3	4	5	6	7	8
Мультиплікативний МАІ	ahp_6	перший	точкова оцінка	метод Сааті	для точкових МПП	перегляд оцінок в МПП; точкові; одноосібний перегляд значень оцінок самим експертом	мультиплікативна згортка; точкові оцінки
МАІ, що використовує скорочену процедуру формування МПП	ahp_7	перший	точкова оцінка	скорочена процедура формування МПП	для точкових МПП	перегляд оцінок в МПП; точкові; одноосібний перегляд значень оцінок самим експертом	лінійна згортка; ієрархічний синтез; точкові оцінки
Модифікація МАІ О.П. Ротштейна	ahp_8	перший	точкова оцінка	метод Сааті	–	–	спеціальні співвідношення; точкові оцінки

Продовження таблиці 2.1

1	2	3	4	5	6	7	8
Модифікація MAI, що використовує метод попарних порівнянь для випадків з великою кількістю параметрів, що оцінюються	ahp_9	перший	точкова оцінка	модифікація методу Сааті для випадків з великою кількістю параметрів, що оцінюються	для точкових МПП	перегляд оцінок в МПП; точкові; одноосібний перегляд значень оцінок самим експертом	лінійна згортка; ієрархічний синтез; точкові оцінки
Модифікація MAI, що використовує спосіб узгодження експертних суджень, що полягає у зміні структури ієрархії	ahp_{10}	перший	точкова оцінка	метод Сааті	для точкових МПП	зміна структури ієрархії; точкові; одноосібний перегляд значень оцінок самим експертом	лінійна згортка; ієрархічний синтез; точкові оцінки
Модифікація MAI, що використовує нелінійну згортку критеріїв	ahp_{11}	перший	точкова оцінка	скорочена процедура формування МПП	для точкових МПП	перегляд оцінок в МПП; точкові; одноосібний перегляд значень оцінок самим експертом	нелінійна згортка; точкові оцінки

Метод аналізу мереж на основі третього типу ієрархії [14], [35] є узагальненням МАІ для випадку, коли взаємодія компонентів ієрархії і/або їх елементів є важливою. Багато проблем прийняття рішень можна представити ієрархічними структурами, тому що в них існують залежності і взаємодії між елементами різних рівнів ієрархії.

Модифікації МАІ в залежності від типу ієрархій дозволяють сформувати складну ієрархічну структуру задачі ГЕО та узагальнити правило побудови моделі багатокритеріального ГЕО.

Залежно від типу ієрархій в подальшому обираються методи формування та обробки МПП, формування підсумкового рішення.

Після визначення типу ієрархічної структури задачі ГЕО формується множина МПП, кількість яких залежить від кількості об'єктів в ієрархії.

Для формування множини МПП виділені наступні модифікації МАІ [31]:

– метод порівняння об'єктів щодо стандартів;

– метод копіюванням;

– скорочена процедура формування МПП;

– модифікація методу парних порівнянь для випадків з великою кількістю оцінюваних параметрів.

Метод порівняння об'єктів щодо стандартів [14] дозволяє зняти обмеження щодо оцінки та порівняння більш ніж дев'яти об'єктів (критеріїв, альтернатив), усунути явище реверсу рангів, тобто зміни порядку раніше виконаних порівнянь альтернатив при додаванні нових альтернатив або видаленні існуючих. Реверс рангів небажаний при вирішенні ряду прикладних задач, пов'язаних зі значними фінансовими, матеріальними та соціальними витратами на подолання наслідків прийнятих рішень або можливістю виникнення конфліктної ситуації між експертами, які готують і обґрунтовують рішення, і особами, які приймають рішення, які несуть відповідальність за прийняті рішення та їх наслідки.

Метод копіюванням [14] аналогічний методу порівняння об'єктів щодо стандартів як такий, що дозволяє зберігати попередній порядок альтернатив, які

вже були проранжовані, при додаванні нових, які є копіями альтернатив, що вже були проранжовані.

Крім того, кількість альтернатив, що аналізуються, при додаванні копій може перевищувати граничне значення, рівне дев'яти, встановлене для методу попарного порівняння. Даний метод дозволить істотно скоротити час експертів на підготовку вихідних даних для аналізу та зменшити ймовірність внесення в них як випадкових, так і логічних помилок.

Скорочена процедура формування МПП [36] забезпечує усунення наступного недоліку класичного МАІ – різкого збільшення кількості оцінок зі збільшенням набору елементів; і, як наслідок, дозволить зменшити трудомісткість формування матриць та підвищити ефективність їх обробки. Процедура дозволяє автоматично формувати узгоджену МПП ("ідеальну") на підставі отриманих оцінок, винесених тільки щодо першого об'єкта порівняння (в цьому випадку експерт формує судження). Таку матрицю можна розглядати як свого роду опорну матрицю при формуванні експертних суджень.

Модифікація методу парних порівнянь для випадків з великою кількістю оцінюваних параметрів [37] заснована на використанні скороченої процедури формування МПП та поділі результатів експертиз випадковим чином на групи, потім для кожної групи та вибірки в цілому розраховують вектор пріоритетів на підставі середньоарифметичного, медіани і середньогеометричного оцінок експертів.

1.4 Висновки до першого розділу

Якість ПЗ, а отже, програмних інтерфейсів, як їх складової частини, визначається прийнятими проєктувальником варіантами рішень проблемних ситуацій, що виникають на різних стадіях проєктування.

Проведений огляд класифікації задач та методів прийняття рішень показав, що задачі, що вирішуються проєктувальником програмних інтерфейсів слід віднести до нетривіальних.

З множини відомих методів і підходів до прийняття рішень найбільший інтерес для оцінки якості програмних інтерфейсів представляють ті, які дають можливість враховувати багатокритеріальність і невизначеність, а також дозволяють здійснювати вибір рішень з множини альтернатив різного типу при наявності критеріїв, що мають різні типи шкал вимірювання: декомпозіційні методи теорії очікуваної корисності; методи аналізу ієрархій і теорії нечітких множин.

Було розглянуто два підходи, що використовуються в колективних СППР – MAI та ранжування альтернатив (методи ELECTRE, PROMETHEE, TOPSIS). Для задачі ГЕО якості програмних інтерфейсів було обрано MAI.

На основі проведеного аналізу модифікацій MAI можна зробити висновок, що задачі ГЕО якості програмних інтерфейсів можна віднести до ієрархії першого типу (див. п. 1.3), тому що для оцінки використовуються стандарти якості в яких присутня ієрархія критеріїв нижніх рівнів (характеристики якості мають підхарактеристики – атрибути). Виходячи з того, що для вирішення задачі ГЕО якості програмних інтерфейсів у більшості випадків досить не більше дев'яти одночасно порівнюваних об'єктів, вважаємо, що достатньо використання класичного MAI (при необхідності, кількість об'єктів може бути підвищена за рахунок автоматизації процесу оцінювання).

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Аналіз стандартів якості програмних засобів

У наш час інтенсивно розвивається напрямок, що виник у процесі рішення задач, пов’язаних із проєктуванням апаратних і ПРЗ обчислювальної техніки – «юзабіліті» (Usability Engineering) – науково-прикладна дисципліна, що слугує підвищенню ефективності, продуктивності й зручності використання інструментів діяльності. Питання кількісної оцінки якості програмних засобів та юзабіліті обговорюються і досліджуються вже декілька десятків років. Наприклад, Шнейдерман вважає, що одним з основних показників зручності використання ПРЗ є [38]:

- швидкість виконання, тобто міра часу, необхідного для виконання певного набору завдань;

- відсоток помилок користувачів – метрика, яка оцінює помилки користувачів. Вимірювання частоти помилок, а також типу помилки корисно для виявлення проблем в експлуатації.

Найджел Беван серед факторів, що сприяють зручності використання системи, виділяє таку характеристику якості, як ефективность використання [39]. Однак розрізненість методів оцінки та відсутність єдиної концепції якості та юзабіліті ускладнюєїх використання проєктувальниками.

Для визначення якості ПРЗ, наявності технічних можливостей системи для взаємодії, її вдосконалення та розвитку необхідно використовувати стандарти у галузі оцінки характеристик їх якості.

Аналіз стандартів, що визначають основні терміни в області ІТ т номенклатуру метрик (показників) юзабіліті програмних продуктів (ПП) показав [3], що частина стандартів застаріла або не погоджена між собою внаслідок неправильного перекладу первинних джерел – стандартів ISO, що призводить до різниці в трактуваннях основних термінів або до їхнього протиріччя.

Визначимо спочатку основні поняття інформаційних технологій, необхідні для подальшого використання, а потім розглянемо підходи до оцінки юзабіліті програмних засобів.

Розгляд почнемо з терміна «програмне забезпечення» (ПЗ). Відповідно до ГОСТ 19781-90 «Обеспечение систем обработки информации программное. Термины и определения», програмне забезпечення – «наукова та практична діяльність зі створення програм». Проте стандарт Росії ГОСТ Р ИСО/МЭК 9126-93 визначає ПЗ (software), як «програми, процедури, правила й будь-яка відповідна документація, що відносяться до роботи обчислювальної системи». У ГОСТ Р ИСО/МЭК-12119-2000, що містить повний автентичний текст стандарту ISO/IEC 12119-94, а також у ГОСТ 28806-90 з поняттям ПЗ пов’язаний інший термін – програмний засіб. Причому даний термін визначений як «усі або частина програм, процедур, правил і будь-якої відповідної документації системи обробки інформації». Очевидно, що в такому трактуванні він фактично збігається з наведеним вище визначенням ПЗ за ГОСТ Р ИСО/МЭК 9126-93. Відмінності тільки в об’єкті використання (обчислювальна система в ГОСТ Р ИСО/МЭК 9126-93 і система обробки інформації в 12119-2000 і ГОСТ 28806-90). Згідно з поясненнями до термінів ГОСТ 28806-90, обсяг поняття, що виражається терміном ПРЗ, містить у собі як окремий випадок обсяг поняття ПЗ, обумовленого за ГОСТ 19781-90, а еквівалентом терміна ПРЗ англійською мовою є термін software, що використовується у своєму збиральному значенні (наприклад mathematical software – програмні засоби для математичних задач) [3].

Надалі будемо використовувати трактування термінів ПРЗ і ПЗ відповідно ГОСТ 19781-90 – як найбільш збалансоване щодо інших стандартів. Два наступні терміни в нормативних документах трактуються практично однаково.

Програмний продукт (software product) – програмний засіб, призначений для постачання, передачі або продажу користувачеві.

Якість ПРЗ (software quality) – сукупність властивостей ПРЗ, які обумовлюють його придатність задовольняти задані (або які маються на увазі) потреби у відповідність із його призначенням.

Основою регламентування показників якості ПРЗ є стандарт ISO 9126. Він має загальний заголовок «Інформаційна технологія. Якість програмного продукту» і складається із чотирьох частин: Частина 1. «Модель якості»; Частина 2. «Зовнішні метрики якості»; Частина 3. «Внутрішні метрики якості»; Частина 4. «Метрики якості у використанні».

Стандарт ISO 9126–01 визначає три зв'язані моделі якості ПРЗ: «внутрішньої якості», пов'язаної з характеристиками ПРЗ самих по собі, без урахування їх поведінки; «зовнішньої якості», що характеризує ПРЗ з точки зору їх поведінки; «якість ПРЗ у використанні» (quality in use) в різних контекстах – тієї якості, яка відчувається користувачами при конкретних сценаріях роботи ПРЗ.

Згідно [32], для системи, якою управляє користувач, поєднання функціональності, надійності, зручності використання і ефективності можна вимірювати ззовні за якістю у використанні.

Для опису внутрішньої й зовнішньої якості ПРЗ використовується багаторівнева модель. На верхньому рівні виділено шість груп базових характеристик. Кожна група описується за допомогою декількох підхарактеристик. Для кожної підхарактеристики визначається набір метрик, що дозволяють її оцінити (рис. 2.1).

Програмний продукт являє собою об'єкт, якому притаманні різні властивості, що часто ускладнює виконання вимірювань самого об'єкта. Ось чому необхідно використовувати якісні характеристики. Однак, найбільшу цінність для оцінки якості ПРЗ представляють кількісні характеристики і субхарактерістікі, а не якісні, які не піддаються безпосередньому виміру або розрахунку.

Наприклад, багато атрибутів програмних інтерфейсів, такі як меню, що випадає або повідомлення про помилку сприяють підвищенню зручності

використання. Тому в стандарті пропонується, щоб ці атрибути були виміряні з використанням базових показників. Потім виміряні значення можуть бути перетворені, з використанням формули, і вказувати на характеристику або субхарактеристику, тобто метрику якості.

Рисунок 2.1 – Основні аспекти якості ПРЗ [3]

Змінна, яка є складовою частиною формули не завжди може бути отримана шляхом вимірювання атрибута цільового об'єкта. Наприклад, міра, яка вказує на безпеку програмного продукту (цілі), вираховується за допомогою вимірювання кількості невдалих спроб (визначається поведінкою користувача) і кількості спроб несанкціонованого доступу.

Характеристика «якість ПРЗ у використанні» (рис. 2.2) фактично збігається з концепцією «зручності використання» (юзабіліті), що формально визначається як «ступінь, в якій продукт може бути використаний певними користувачами для досягнення певних цілей з ефективністю, дієвістю та задоволеністю в конкретному контексті використання» [40]. ISO 9241-11 дає такі визначення підхарактеристик якості ПРЗ при використанні:

– ефективність (effectiveness) – здатність програмного продукту надавати користувачам можливість вирішувати їх задачі з необхідною точністю й повнотою при використанні в заданому контексті;

– продуктивність (productivity) – здатність програмного продукту надавати користувачам можливість витрачати відповідну кількість ресурсів для досягнення ефективності в зазначеному контексті використання (наприклад, час рішення задачі, зусилля, витрачаємі користувачем, матеріали або витрати на експлуатацію);

– безпечність (safety) – здатність програмного продукту забезпечувати припустимий рівень ризику від його застосування для людей, бізнесу, ПРЗ, майна або навколишнього середовища;

– задоволеність користувачів (satisfaction) – здатність програмного продукту приносити задоволення користувачам при використанні в заданому контексті.

Рисунок 2.2 – Модель якості ПРЗ у використанні

У 2006 році серія стандартів ISO 9241 була перейменована під більш загальною назвою "Ергономіка взаємодії людини і системи".

Недоліки серії стандартів ISO 9126 наступні:

– відсутність методики кількісного виміру характеристик і їх зіставлення з вимогами специфікацій;

– відсутність рекомендацій, на яких етапах життєвого циклу ПРЗ їх доцільно застосовувати.

Перелік відповідних робіт з оцінки якості програмних продуктів визначений серією стандартів ISO14598-98 «Software Product Evaluation»:

- частина 1 «Загальний огляд»;
- частина 2 «Планування і управління»;
- частина 3 «Процес для розробників»;
- частина 4 «Процес для покупців»;
- частина 5 «Процес для тестувальників»;
- частина 6 «Модуль оцінки».

Стандарти серій ISO 9126 і 14598 набули широкого поширення. Незважаючи на те, що стандарти ISO 9126-2, 3, 4 залишаються актуальними і обидві серії мають загальні нормативні, довідкові та функціональні коріння, багато дослідників виявили ряд обмежень і проблем цієї серії і вказали на необхідність їх поліпшення:

- не можуть підтримувати специфікацію вимог до програмного забезпечення на всіх етапах життєвого циклу програмного продукту і не мають стандарту, відповідного аналізу вимог якості програмного продукту [41];
- існує потреба в новій унікальній архітектурі і загальному керівництві, базових метриках, керівництві по використанню метрик, стандарті вимог до якості (думка редактора і члена комітету ISO Йоргена Боега).

Ці проблеми були усунуті за допомогою введення нової серії стандартів, позначених номером 25000. Стандарти складаються з п'яти секцій: управління якістю, модель якості, вимірювання якості, вимоги до якості і оцінка якості.

Стандарт ISO/IEC 25010.2–2008 «Інформаційна технологія. Програмний продукт. Характеристики якості й оцінка. Модель якості» вводить скореговану модель якості програмного продукту (рис. 2.3). Запропонований розширений аналіз якості у використанні і введені наступні три групи характеристик (рис. 2.4): зручність і простота у використанні (usability in use), яка визначається тими ж характеристиками, що й у стандарті ISO 9241-11 (ступінь ефективності, продуктивність і задоволеність); гнучкість у використанні (flexibility in use), є мірою придатності програмного продукту до застосування у всіх можливих контекстах використання, включаючи доступність; безпечність (safety), яка

стосується зменшення небажаних наслідків від використання програмного продукту [3].

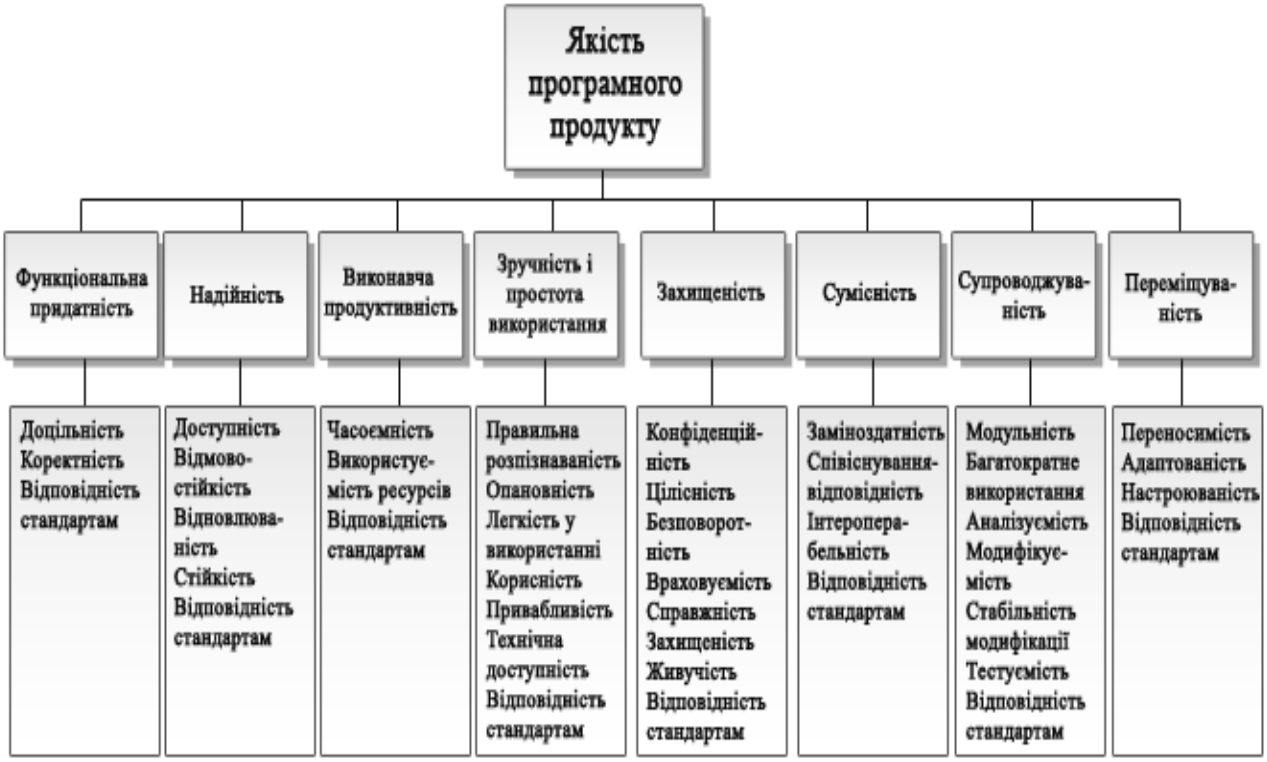


Рисунок 2.3 – Модель якості програмного продукту [3]

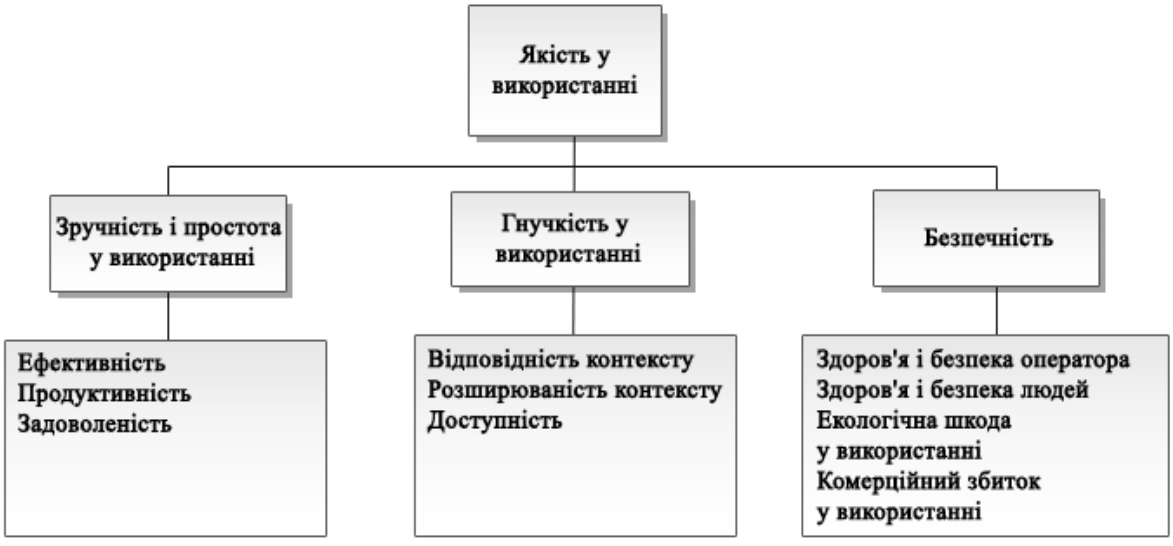


Рисунок 2.4 – Модель якості у використанні [3]

Гнучкість у використанні визначається наступними підхарактеристиками:

– відповідність контексту у використанні (context conformity in use – ступінь відповідності вимог до зручності й простоти у використанні у всіх передбачуваних контекстах використання);

– розширюваність контексту у використанні (context extendibility in use – ступінь зручності й простоти використання в контекстах, які не були спочатку визначені);

– доступність у використанні (accessibility in use – ступінь зручності й простоти у використанні для користувачів з певними обмеженими можливостями, у тому числі віковими).

Безпечність визначається наступними характеристиками:

– здоров'я й безпека оператора (operator health and safety);

– здоров'я й безпека людей (public health and safety);

– екологічна шкода у використанні (environmental harm in use);

– комерційний збиток у використанні (commercial damage in use).

Стандарти рекомендують використовувати десятки показників для вимірювання якості. Тому у розробників часто виникають труднощі з їх застосуванням і інтерпретацією.

Основним показником того, що користувач виконав свої завдання в реальному світі, є якість використання реального продукту. Більш того, відповідно до стандарту ISO 9126-01 для програмного продукту, яким керує користувач, якість використання вимірюється сукупним ефектом від простоти використання, функціональності, ефективності і надійності.

Багато авторів (наприклад, [42]) зазначають, що підвищення якості використання призводить до поліпшення таких характеристик, як ефективність, продуктивність, прийнятність, і знижує такі характеристики, як «людська помилка», а також час і зусилля на навчання користувачів. Якість використання і зручність використання грають важливу роль і повинні прийматися до уваги на всіх етапах процесу проєктування. Тому оцінка і прогноз якісних характеристик на ранніх етапах проєктування стає актуальним завданням.

Вирішення цієї проблеми дозволить значно скоротити терміни проєктування і вартість проєкту.

Взаємозв'язок між стандартами ISO 9126 та ISO 14598 та історію їх заміни показано на рис. 2.5. Стандарт ISO 9126 позначений у верхній частині пунктиром, оскільки в 1991 році не було введено в експлуатацію моделей внутрішньої та зовнішньої якості та якості у використанні.

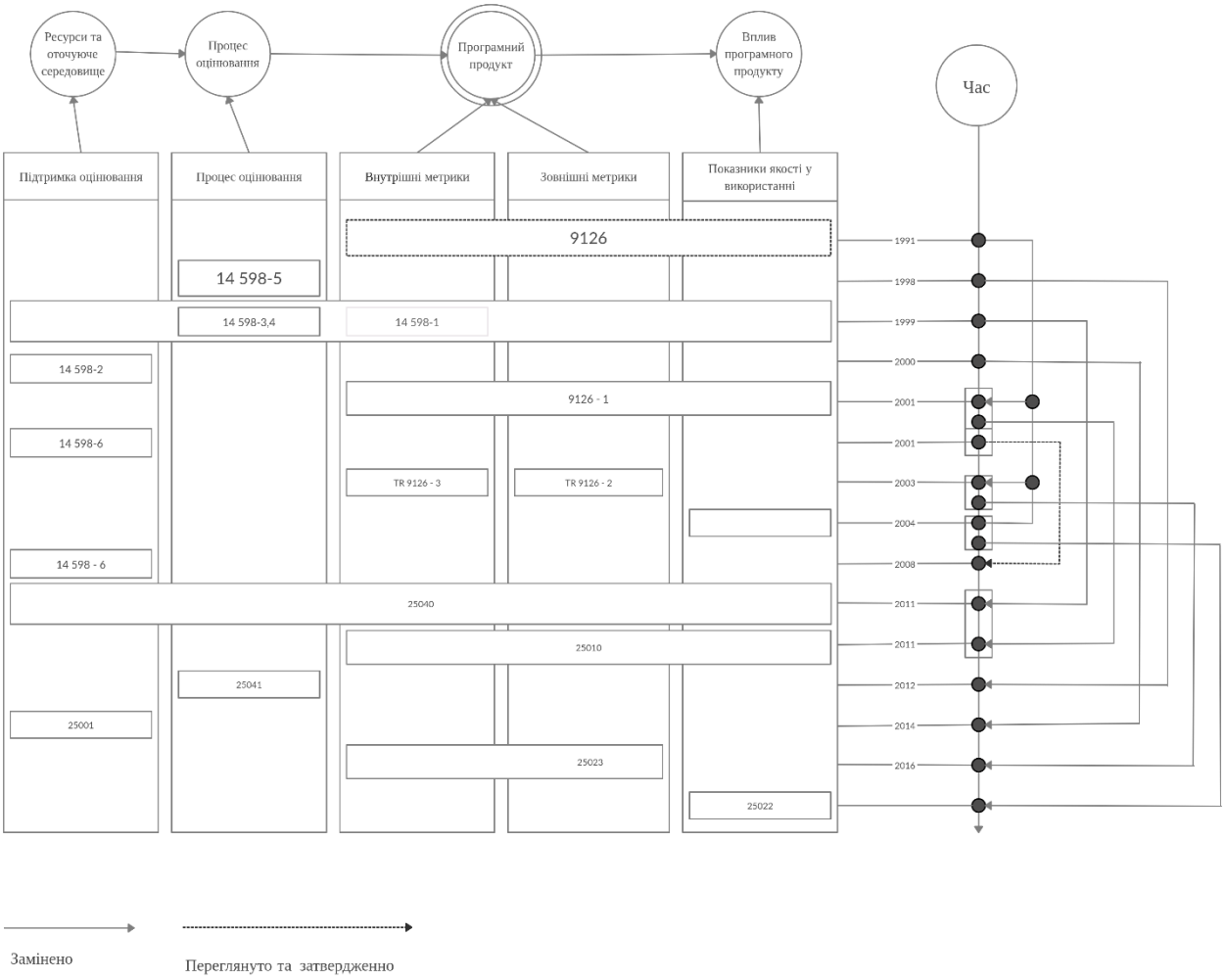


Рисунок 2.5 – Взаємозв'язок між стандартами ISO 9126 та ISO 14598

Проведений аналіз стандартів показав, що юзабіліті-метрики містяться в стандартах ISO 9126-4, ISO 9241-11 і ISO/IEC 25010.2. Їхній перелік і вміст наведено в таблиці 2.1 [3].

Таблиця 2.1 – Групи юзабіліті-метрик

ISO 9126-4	ISO 9241-11	ISO/IEC 25010.2
1	2	3
Ефективність (effectiveness): оцінює результати виконання користувачем задач із певною точністю й повнотою	**Ефективність** (effectiveness): точність і повнота, з якою користувачі досягають поставлених цілей	**Ефективність у використанні** (effectiveness in use) – збігається з ISO 9241-11
Продуктивність (productivity): оцінює витрати користувачів при одержуваній ефективності	**Продуктивність** (efficiency): відношення витрачених ресурсів до точності й повноті, з якою користувачі досягають поставлених цілей	**Продуктивність у використанні** (efficiency in use) – збігається з ISO 9241-11
Безпечність (safety): оцінює рівень ризику, шкоди людям, бізнесу, ПЗ, власності або навколишньому середовищу	**Група відсутня**	**Безпека** (safety): оцінює рівень ризику, шкоди людям, бізнесу, ПЗ, власності або навколишньому середовищу. Введені нові субхарактеристики
Задоволеність (satisfaction): оцінює відношення користувача до роботи із програмним продуктом	**Задоволеність** (satisfaction): комфорт і прийнятність використання. Її можна оцінювати як відношення до використання продукту, так і сприйняття користувачем таких показників, як продуктивність, корисність або легкість у вивченні	**Задоволеність у використанні** (satisfaction in use) – збігається з ISO 9241-11

Продовження таблиці 2.1

1	2	3
Група відсутня	**Група відсутня**	**Гнучкість у використанні (**flexibility in use**):** оцінює рівень зручності застосування продукту у всіх можливих контекстах використання

Як видно з таблиці 2.1, у стандартах збігаються перша й четверта групи (хоча й з деякими змістовими відмінностями), по-різному називається друга група. В ISO 9241-11 відсутня група «Безпечність», а група «Гнучкість у використанні» відсутня в ISO 9241-11 і ISO 9126-4. Таким чином, у стандартах є розбіжності вже на цьому, самому загальному, рівні опису юзабіліті-метрик.

Самі метрики, що входять у перераховані стандарти, досить довільні по складу й розмиті по змісту. При цьому ISO 9241-11 пропонує тільки приклади метрик залежно від мети тестування: загальна юзабіліті, припустимість для кваліфікованих користувачів, припустимість для новачків, мінімізація вимог підтримки, стійкість до помилок і т.п.

2.2 Метод аналізу ієрархій

Як було розглянуто у п. 1.1 використання MAI дозволяє здійснювати вибір рішень з множини альтернатив різного типу при наявності критеріїв, що мають різні типи шкал вимірювання. Тобто, цей метод доцільно застосовувати до прийняття рішень в ситуаціях ГЕО якості інтерфейсу, коли, наприклад, для ідей, почуттів, емоцій визначаються деякі кількісні показники, що забезпечують числову шкалу переваг для можливих альтернативних рішень.

Постановка задачі багатокритеріальної оптимізації, що розв'язується за допомогою МАІ, як правило, виглядає наступним чином:

– задана загальна мета (або цілі), які обумовлюються призначенням системи, що аналізується;

– задані *m* альтернативних рішень для досягнення мети;

– задані *n* критеріїв, за якими оцінюються наявні альтернативи в форматі завдань оптимізації системи, що аналізується;

– потрібно: вибрати найкращу альтернативу.

Метод МАІ передбачає реалізацію наступних етапів для вирішення завдань певного типу [14], [23], [43].

Етап 1. Структуризація ієрархії. На першому етапі вихідна задача представляється у вигляді відповідної ієрархічної структури за рівнями: «цілі - критерії - альтернативи».

Етап 2. Попарне порівняння. Другий етап полягає в реалізації попарних порівнянь для елементів кожного рівня з урахуванням специфіки їх оцінки елементами (критеріями, цілями) попереднього вищого рівня ієрархії. Результати такого порівняння (для кожної пари елементів одного рівня ієрархії за оцінками одного типу) представляються відповідною матрицею порівняння. Процедури формалізації таких матриць порівняння дозволяють враховувати переваги ОПР. Для зручності реалізації процедур попарного порівняння в форматі МАІ розроблена спеціальна шкала відношень. (табл. 2.2). Дана шкала дозволяє ОПР ставити у відповідність ступенями переваги одного порівнюваного об'єкта перед іншим деякі числа.

Правомочність цієї шкали доведена теоретично при порівнянні з багатьма іншими шкалами [14]. При використанні зазначеної шкали ОПР, порівнюючи два об'єкти в сенсі досягнення мети, розташованої на вищележачому рівні ієрархії, повинен поставити у відповідність цього порівняння число в інтервалі від 1 до 9 або зворотне значення чисел. У тих випадках, коли важко розрізнити стільки проміжних градацій від абсолютної до слабкої переваги або цього не потрібно в конкретному завданні, може

Таблиця 2.2 – Шкала відношень

Ступінь значущості	Визначення	Пояснення
1	Однакова значимість	Два дії вносять однаковий внесок у досягнення мети
3	Деяка перевага значущості однієї дії над іншою (слабка значимість)	Існують міркування на користь переваги однієї з дій, однак ці міркування недостатньо переконливі
5	Істотна або сильна значимість	Є надійні дані або логічні судження для того, щоб показати перевагу однієї з дій
7	Очевидна або дуже сильна значимість	Переконливе свідчення на користь однієї дії перед іншою
9	Абсолютна значимість	Свідчення на користь переваги однієї дії над іншою надзвичайно переконливі
2,4,6,8	Проміжні значення між двома сусідніми судженнями (1-3, 3-5, 5-7, 7-9)	Ситуація, коли необхідне компромісне рішення
Зворотні величини наведених вище ненульових величин	Якщо дії *i* при порівнянні з дією *j* приписується одне з визначених вище ненульових чисел, то дії *j* при порівнянні з дією *i* приписується зворотне значення	Якщо узгодженість була постульована при отриманні *N* числових значень для утворення матриці

використовуватися шкала з меншим числом градацій. У найпростішому випадку шкала має дві оцінки: 1 – об'єкти рівнозначні; 2 – перевага одного об'єкта над іншим.

Далі будується множина МПП. Для цього в ієрархії виділяють елементи двох типів: елементи-«батьки» та елементи-«нащадки». Елементи-«нащадки» впливають на відповідні елементи вищого рівня ієрархії, що є по відношенню до перших елементами-«батьками». МПП будуються для всіх елементів-«нащадків», що відносяться до відповідного елементу-«батька». Елементами-«батьками» можуть бути елементи, що належать будь-якому ієрархічному рівню, крім останнього, на якому розташовані, як правило, альтернативи. Парні порівняння проводяться в термінах домінування одного елемента над іншим. Отримані судження виражаються в цілих числах з урахуванням дев'ятибальної шкали (див. таблицю 2.2).

Заповнення квадратних МПП здійснюється за наступним правилом. Якщо елемент E_1 домінує над елементом E_2, то клітина матриці, відповідна рядку E_1 і стовпцю E_2, заповнюється цілим числом, а клітина, відповідна рядку E_2 і стовпцю E_1, заповнюється зворотним до нього числом. Якщо елемент E_2 домінує над E_1, то ціле число ставиться в клітку, відповідну рядку E_2 і стовпцю E_1, а дріб проставляється в клітку, відповідну рядку E_1 і стовпцю E_2. Якщо елементи E_1 і E_2 мають рівні переваги, то в обидві позиції матриці ставляться одиниці.

Для отримання кожної матриці експерт або ОПР виносить $n(n - 1)/2$ суджень (де n – порядок МПП).

Розглянемо в загальному вигляді приклад формування МПП [14].

Нехай $E_1, E_2, ..., E_n$ – множина з n елементів (альтернатив) і $v_1, v_2, ..., v_n$ - відповідно їх ваги, або інтенсивності. Порівняємо попарно ваги, або інтенсивність, кожного елемента з вагою, або інтенсивністю, будь-якого іншого елемента множини по відношенню до загальної для них властивості або цілі (по відношенню до елементу-«батьку»). Матриця попарних порівнянь [E] у вигляді таблиці наведена у таблиці 2.3.

Таблиця 2.3 – Матриця попарних порівнянь

	E_1	E_2	…	E_n
E_1	v_1/v_1	v_1/v_2	…	v_1/v_n
E_2	v_2/v_1	v_2/v_2	…	v_2/v_n
…	…	…	…	…
E_n	v_n/v_1	v_n/v_2	…	v_n/v_n

Матриця попарних порівнянь має властивість зворотної симетрії, тобто $a_{ij}=1/a_{ji}$, де $a_{ij}=v_i/v_j$.

При проведенні попарних порівнянь слід відповідати на такі питання: який із двох порівнюваних елементів важливіше або має більший вплив, який більш ймовірний і який краще.

При порівнянні критеріїв зазвичай запитують, який з критеріїв більш важливий; при порівнянні альтернатив по відношенню до критерію – яка з альтернатив більш краща або більш імовірна.

Етап 3. Ваги і коефіцієнти важливості. Перевірка суджень на узгодженість.

Для кожної матриці порівнянь визначаються її власні вектори (реалізуються наближені методи їх знаходження). Знайдені значення компонент власних векторів для кожної МПП дозволяють визначити ваги і коефіцієнти важливості для порівнюваних елементів відповідного рівня ієрархії. Далі при порівнянні критеріїв (за їх важливостю/ефективністю для досягнення мети) показники такого типу (на їх основі може бути відновлена МПП) будемо називати «вагами». При порівнянні альтернативних рішень (по конкретному критерію) такі показники будемо називати «коефіцієнтами важливості». Результати зазначених процедур оформляються у вигляді спеціальних таблиць. При цьому перевіряється узгодженість суджень ОПР.

Розглянемо яким чинов відбувається ранжування елементів.

Ранжування елементів, які аналізуються з використанням МПП [E], здійснюється на підставі головних власних векторів, що отримуються в результаті обробки матриць.

Обчислення головного власного вектора W додатної квадратної матриці [E] проводиться на підставі рівняння, формула (2.1) [14]:

$$EW=\lambda_{max}W, \tag{2.1}$$

де λ_{max} – максимальне власне значення матриці [E].

Максимальна власне значення обчислюється за формулою (2.2):

$$\lambda_{max}=\mathrm{e}^{T}[E]W. \tag{2.2}$$

Для додатної квадратної матриці [E] правий власний вектор W, який відповідає максимальному власному значенню λ_{max}, з точністю до постійного співмножника C можна обчислити за формулою (2.3):

$$\lim_{k\to\infty}\frac{[E]^{k}e}{e^{T}[E]^{k}e}=CW, \tag{2.3}$$

де $e=\{1,1,1,\ldots 1\}^{T}$ – одиничний вектор;

k = 1, 2, 3, ... – показник ступеня;

C – константа;

T – знак транспонування.

Узгодженість МПП перевіряється наступним чином.

Якщо все стовпчики нормализованной матриці ідентичні, то вихідна матриця порівняння є узгодженою. Узгодженість означає, що рішення буде погоджено з визначеннями парних порівнянь критеріїв або альтернатив. З математичної точки зору узгодженість матриці A означає, що $a_{ij}a_{jk} = a_{ik}$ для всіх i, j і k.

Якщо матриця парних порівнянь не є узгодженою необхідно з'ясувати, чи є рівень узгодженості припустимим. Для цього необхідно визначити відповідну кількісну міру, тобто, коефіцієнт узгодженості *CR*, який знаходиться за формулою (2.4):

$$CR = \frac{CI}{RI}, \tag{2.4}$$

де $CI = \frac{\lambda_{max} - \lambda}{\lambda - 1}$ – індекс узгодженості МПП;

$RI = \frac{1{,}98(\lambda - 2)}{\lambda}$ – стохастичний коефіцієнт узгодженості МПП.

Стохастичний коефіцієнт узгодженості *RI* визначається емпіричним шляхом як середнє значення коефіцієнта *CI* для великої вибірки згенерованих випадковим чином матриць порівняння *A*.

Коефіцієнт узгодженості *CR* використовується для перевірки узгодженості матриці порівняння *A* наступним чином. Якщо *CR*<0,1, рівень неузгодженості є прийнятним. В іншому випадку рівень неузгодженості матриці порівняння *A* є високим, і особі, що приймає рішення, рекомендується перевіряти елементи парного порівняння a_{ij} матриці *A* з метою отримання більш узгодженої матриці.

Таким чином, результатом реалізації процедур другого і третього етапів методу MAI буде знаходження:

– вагових коефіцієнтів (або вагів) w_i для всіх елементів одного рівня ієрархії, що відноситься до заданих критеріїв оцінки альтернатив (сума вагів заданих критеріїв має дорівнювати одиниці);

– коефіцієнтів важливості v_{ki} для *k*-ої альтернативи по *i*-му критерію (сума коефіцієнтів важливості всіх альтернатив, що аналізуються за допомогою одного з критеріїв також має дорівнювати одиниці).

Етап 4. Пріоритети альтернатив. Нарешті, обчислюються підсумкові кількісні індикатори якості для кожної з альтернатив. Їх називають пріоритетами. Зазначені показники дозволяють визначити найкраще

альтернативне рішення для відповідного програмного інтерфейсу. Це – альтернатива з найвищим пріоритетом.

Позначимо пріоритети альтернатив *k* як V_k. На розміченому графі ієрархії пріоритети «приписуються» альтернативам, що аналізуються. Для кожної альтернативи показник її пріоритету синтезується за коефіцієнтами важливості (v_{ki}) цієї альтернативи (для всіх заданих критеріїв оцінки альтернатив) з урахуванням ваг (w_i) цих критеріїв. Зазначений синтез здійснюється за формулою (2.5):

$$V_k = \sum w_i \cdot v_{ki}\,, \tag{2.5}$$

де V_k – підсумковий показник якості або пріоритет *k*-ої альтернативи в рамках аналізу заданої ієрархічної структури за методом МАІ;

w_i – вага *i*-го критерію, тобто *i*-а компонента нормованого власного вектора-стовпця для МПП заданих критеріїв по їх важливості для досягнення мети;

v_{ki} – показник важливості *k*-ої альтернативи по *i*-му критерію, тобто *k*-а компонента нормованого власного вектора-стовпця для відповідної МПП альтернатив з позицій *i*-го критерію.

За значеннями знайдених пріоритетів альтернатив, що аналізовались і визначається рішення, яке приймається в якості оптимального або найкращого за методом МАІ. Це буде рішення з найбільшим пріоритетом.

Технічне завдання (ТЗ) на розробку програми, що автоматизує ГЕО якості програмних інтерфейсів на базі методу МАІ наведено у Додатку А.

2.3 Висновки до другого розділу

Аналіз стандартів якості програмних засобів дозволив виявити три зв’язані моделі якості ПРЗ: внутрішньої якості; зовнішньої якості та якості ПРЗ у використанні.

Згідно стандарту ISO/IEC 9126-1 для системи, якою управляє користувач, поєднання функціональності, надійності, зручності використання і ефективності можна вимірювати ззовні за якістю у використанні. Характеристика «якість ПРЗ у використанні» фактично збігається з концепцією «зручності використання» (юзабіліті) введеною стандартом ISO/IEC 9126-11. Згідно цього стандарту «якість ПРЗ у використанні» визначається наступними підхарактеристиками: ефективність; продуктивність; безпечність та задоволеність користувачів.

У більш новому стандарті ISO/IEC 25010.2–2008 запропоновано розширений аналіз якості у використанні і введені наступні групи характеристик: зручність і простота у використанні, яка визначається тими ж характеристиками, що й у стандарті ISO 9241-11 (ступінь ефективності, продуктивність і задоволеність); гнучкість у використанні, яка є мірою придатності програмного продукту до застосування у всіх можливих контекстах використання, включаючи доступність та безпечність, яка стосується зменшення небажаних наслідків від використання програмного продукту.

Метрики, запропоновані в стандартах, неповні, часто не збігаються між собою, іноді суперечливі, їх набір довільний і не охоплює багато аспектів діяльності. Все це знижує практичну значущість стандартів. По-перше, юзабіліті-фахівці повинні самостійно займатися розробкою метрик і, отже, неминуче буде відсутня їхня спільність, уніфікація. По-друге, результати отримані з використанням різних варіантів метрик, буде важко порівнювати. Тому актуальним є автоматизація ГЕО якості програмних інтерфейсів на базі методу MAI.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

В даному розділі представлено опис розробки програми Analytic Hierarchy Process (AHP) - програмного забезпечення для комп'ютерної підтримки процесу ГЕО якості програмних інтерфейсів на основі методу MAI. Для розробки застосовувалася мова програмування C# та середовище розробки Microsoft Visual Studio 2015.

3.1 Моделювання програмного забезпечення

Аналіз ТЗ показав, що для реалізації програми необхідно виділити основні дії, які повинні бути виконані для вирішення поставлених завдань, а саме:

- зберігати значення критеріїв і альтернатив;
- зберігати назви критеріїв;
- зберігати і оперувати значеннями МПП;
- підтримувати коректність введених даних;
- виконувати побудову та відображення ієрархічної структури.

Взаємодія користувача з системою AHP, що необхідна для виконання функцій ГЕО наведена у вигляді діаграми прецендентів на рис. 3.1.

Основними діями користувача є ввод-вивід необхідних для ГЕО даних та візуальне сприйняття сформованих системою відповідей, а основними діями системи є проведення відповідних обчислень і відображення даних в необхідному вигляді.

Після цього користувачеві необхідно дати назви критеріям. На основі введених даних про критерії та альтернативи програма генерує МПП. Користувач вводить оцінки в матрицю і, після натискання на відповідну кнопку, програма обчислює значення локальних пріоритетів, а також параметрів: DIM (розмір МПП); LAM (максимальне власне значення МПП – розраховується за

формулою (2.2)); CI (індекс узгодженості МПП) та CR (коефіцієнт узгодженості – розраховується за формулою (2.4)). Процес «генерація - введення - обчислення» повторюється для кожного критерію оцінки.

На основі діаграми прецедентів і послідовності кроків методу класичного MAI був розроблений алгоритм роботи програми AHP, представлений на рис. 3.2.

Алгоритм програми складається з послідовного введення необхідних даних і обчислень на підставі цих даних. Початок роботи алгоритму полягає в тому, що користувач вводить кількість критеріїв і альтернатив.

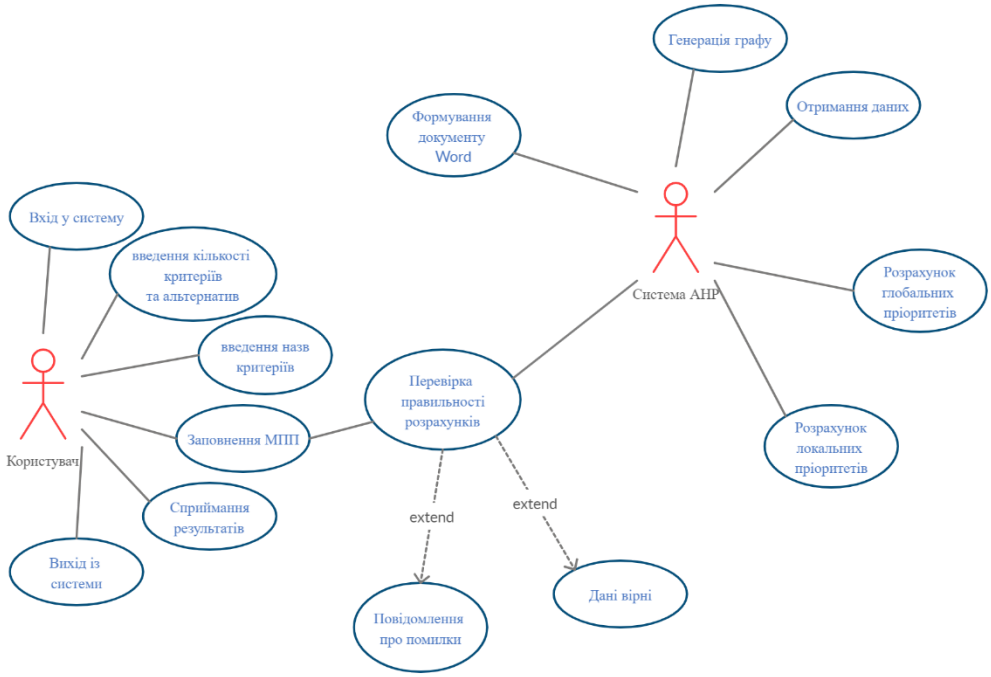


Рисунок 3.1 – Діаграма прецедентів системи AHP

Коли для всіх критеріїв будуть обчислені значення локальних пріоритетів, програма відображає результати за кожним критерієм. На наступному кроці виконується розрахунок глобальних пріоритетів і побудова ієрархічної структури з виділенням альтернативи, глобальний пріоритет якої, має найбільше значення. Після побудови і обчислення програма відображає результати. Алгоритм завершується збереженням отриманих результатів у файл формату документа Word.

Процедури введення і виведення даних користувачем у програму відбуваються за допомогою графічного інтерфейсу.

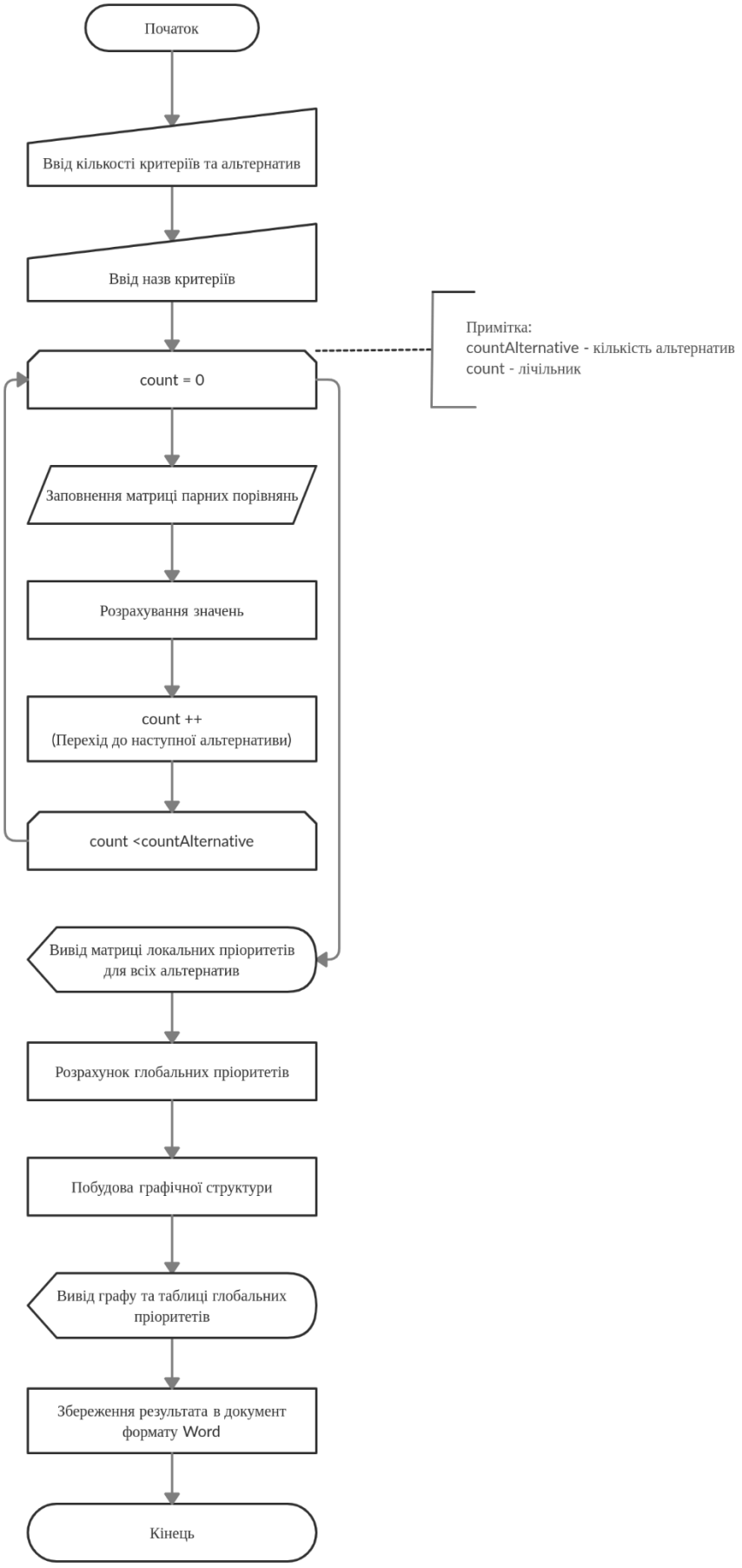


Рисунок 3.2 – Алгоритм роботи програми AHP

Для забезпечення виконання вимог ТЗ було розроблено чотири класи: InfoCriteria, ResultRatingTable, IValidation, Figures, Graph (форма). Діаграма класів представлена на рис. 3.3.

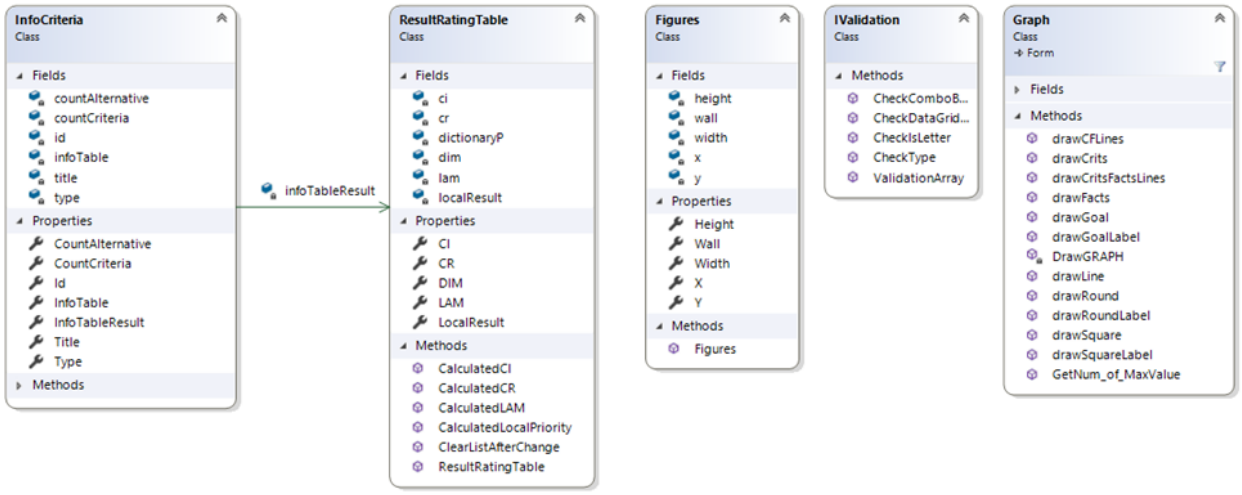


Рисунок 3.3 – Діаграма класів програми

Більшість класів є незалежними один від одного і не мають зв'язків. Однак, існує асоциативний зв'язок has-a між класами InfoCriteria та ResultRatingTable. Розглянемо розроблені класи більш детально.

3.2 Розробка класів програми

3.2.1 Клас InfoCriteria

Клас InfoCriteria – основний клас програми, який є контейнером для всіх даних необхідних для забезпечення роботи програми. Тим самим він реалізує в сукупності перші три завдання, що описані в п. 3.1.

Програмний код класу InfoCriteria з необхідними коментарями наступний:

```
public class InfoCriteria
    {
        static private int countCriteria = 0;
        static private int countAlternative = 0;
```

```
                private int id;                         // Ідентифікатор таблиці
                private string type = String.Empty;     // Тип шкали
                private string title = null;            // Назва крітерию
                private string[,] infoTable;            // Значення таблиці
                private ResultRatingTable infoTableResult;
                public InfoCriteria() { }
                public InfoCriteria(InfoCriteria o)
                {
                    Id = o.Id;
                    InfoTable = o.InfoTable;
                    infoTableResult = o.infoTableResult;
                    Title = o.Title;
                }
                public InfoCriteria(int id, ResultRatingTable infoTableResult,
string title, string type, string[,] arr)
                {
                    Id = id;
                    InfoTableResult = infoTableResult;
                    Title = title;
                    Type = type;
                    InfoTable = arr;
                }
                public int Id { get => id; set => id = value; }
                public ResultRatingTable InfoTableResult { get =>
infoTableResult; set => infoTableResult = value; }
                public string Title { get => title; set => title = value; }
                public static int CountCriteria { get => countCriteria; set =>
countCriteria = value; }
                public static int CountAlternative { get => countAlternative;
set => countAlternative = value; }
                public string Type { get => type; set => type = value; }
                public string[,] InfoTable { get => infoTable; set => infoTable
= value; }
        }
```

Параметри класу наведені в табл. 3.1.

Таблиця 3.1 – Параметри класу InfoCriteria

Параметр	Опис
static private int countCriteria	Кількість критеріїв
static private int countAlternative	Кількість альтернатив
private int id	Ідентифікатор таблиці
private string type	Тип шкали виставлення оцінки
private string title	Назва критерія
private string[,] infoTable	Значення МПП
private ResultRatingTable infoTableResult	Змінна, яка містить обчислені значення

Методи, що присутні в класі InfoCriteria відповідають за повернення і прийняття значення (getter/setter).

3.2.2 Клас ResultRatingTable

Клас ResultRatingTable – це клас програми, який є контейнером для обчислюваних значень: локальних пріоритетів для кожного критерію; параметрів DIM, LAM, CI, CR (див. п.3.1). Клас містить функції по обчисленню кожного з них.

Програмний код класу ResultRatingTable з необхідними коментарями наступний:

```
    public class ResultRatingTable
    {
        // МПП - матриця попарних порівнянь
        private List<double> localResult;   // Список локальних пріоритетів
        private double dim;                 // Розмір МПП
        private double lam;                 // Максимальне власне значення МПП
        private double ci;                  // Індекс узгодженності МПП
        private double cr;                  // Коефіцієнт узгодженості МПП
        public ResultRatingTable(List<double> localResult, double dim, double
lam, double ci, double cr)
        {
            this.localResult = localResult;
            this.dim = dim;
            this.lam = lam;
            this.ci = ci;
            this.cr = cr;
        }
        public List<double> LocalResult { get => localResult; set =>
localResult = value; }
        public double DIM { get => dim; set => dim = value; }
        public double LAM { get => lam; set => lam = value; }
        public double CI { get => ci; set => ci = value; }
        public double CR { get => cr; set => cr = value; }
        public void CalculatedLocalPriority(string[,] array, int size)
        {
            if (this.LocalResult.Count > 0)
                LocalResult.Clear();
            double srgeom = 0;
            string tempS = "";
            double tempMiddleGeometry = 1;
            List<double> listMiddleGeometry = new List<double>();
            DataTable a = new DataTable();
            for (int i = 0; i < size; i++)
            {
                for (int j = 0; j < size; j++)
```

```
                {
                    srgeom = 0;
                    tempS = array[i, j];
                    if (tempS.Length > 1)
                        tempMiddleGeometry *=
Convert.ToDouble(a.Compute(tempS, ""));
                    else
                        tempMiddleGeometry *= Convert.ToDouble(tempS);
                }
                srgeom = Math.Pow(tempMiddleGeometry, 1.0 / size);
                listMiddleGeometry.Add(srgeom);
                tempMiddleGeometry = 1;
            }
            double tempSumMidleGeometryc = 0;
            for (int i = 0; i < listMiddleGeometry.Count; i++)
                tempSumMidleGeometryc += listMiddleGeometry[i];
            for (int i = 0; i < size; i++)
                this.LocalResult.Add((listMiddleGeometry[i] /
tempSumMidleGeometryc));
            dim = size;
            CalculatedLAM(array, size);
            CalculatedCI(size);
            CalculatedCR(size);
        }
        public void CalculatedLAM(string[,] array, int size)
        {
            double llam = 0;
            double sum = 0;
            DataTable a = new DataTable();
            string tempS = "";
            for (int i = 0; i < size; i++)
            {
                for (int j = 0; j < size; j++)
                {
                    tempS = array[j, i];
                    if (tempS.Length > 1)
                        sum += Convert.ToDouble(a.Compute(tempS, ""));
                    else
                        sum += Convert.ToDouble(tempS);
                }
                llam += sum * this.LocalResult[i];
                sum = 0;
            }
            lam = llam;
        }
        public void CalculatedCI(int size)
        {
            ci = (lam - size) / (size - 1);
        }
        private static Dictionary<int, double> dictionaryP = new
Dictionary<int, double>()
        {
            {2, 0}, {3, 0.58}, {4, 0.90}, {5, 1.12}, {6, 1.24} , {7, 1.32}
        };
        public void CalculatedCR(int size)
        {
```

```
        double P = dictionaryP[size];
        if (P!=0)
            cr = (ci / P);
        else
            cr = 0;
    }
    public void ClearListAfterChange()
    {
        LocalResult.Clear();
    }
```

Параметри класу ResultRatingTable наведені в табл. 3.2.

Таблиця 3.2 – Параметри класу ResultRatingTable

Параметр	Опис
private List<double> localResult	Список локальних пріоритетів.
private double dim	Розмір МПП
private double lam	Максимальне власне значення МПП
private double ci	Індекс узгодженності МПП
private double cr	Коефіцієнт узгодженості МПП

Для кожного поля класу існують методи getter/setter для отримання або установки значення полів.

Методи класу ResultRatingTable наведені в табл. 3.3.

Таблиця 3.3 – Методи класу ResultRatingTable

№	Метод	Опис
1	public void CalculatedLocalPriority(string[,] array, int size)	Метод обчислює значення локальних пріоритетів для кожного критерію. На вході отримує МПП і її розмір. Для обчислення вектора локальних пріоритетів використовується спрощений спосіб

Продовження таблиці 3.3

№	Метод	Опис
2	public void CalculatedLAM (string[,] array, int size)	Метод для обчислення максимального власного значення МПП
3	public void CalculatedCI(int size)	Метод для обчислення індексу узгодженності МПП
4	public void CalculatedCR(int size)	Метод для обчислення коефіцієнту узгодженості МПП

Розглянемо послідовність розрахунків для методів класу ResultRatingTable.

У методі №1 розрахунки виконуються у наступній послідовності.

Крок 1. Для кожного рядка матриці парних порівнянь знаходимо середнє геометричне її елементів за формулою (3.1):

$$a_{ij} = (a_{ij}^1 \times a_{ij}^2 \times ... \times a_{ij}^s)^{\frac{1}{s}}. \quad (3.1)$$

Крок 2. Знаходимо суму всіх цих середніх геометричних.

Крок 3. Ділимо кожне середнє геометричне на їх суму («нормування на одиницю»).

У методі №2 максимальне власне значення знаходимо за формулою (3.2):

$$\sum_{i=1}^{n}(g_i \times \sum_{j=1}^{n} a_{ij}), \quad (3.2)$$

де n – кількість стовпців МПП;

g_i – i-тий компонент нормалізованого вектора пріоритетів (НВП);

У методі №3 індекс узгодженості знаходимо за формулою (3.3):

$$CI = \frac{\lambda_{max} - \lambda}{\lambda - 1}. \quad (3.3)$$

У методі №4 коефіцієнт узгодженості знаходимо за формулою (2.4).

Значення ймовірностей в програмі представлено словником dictionaryP.

3.2.3 Клас IValidation

Клас IValidation реалізований для контролю коректності введення даних в застосунок.

Програмний код класу IValidation з необхідними коментарями наступний:

```
    public class IValidation
    {
// Перевірка наявність введених у ComboBox значень
        public static bool CheckComboBoxValue(Object o)
        {
            if (o != null)
                return true;
            else
                return false;
        }
// Перевірка коректності введених у таблицю назв критеріїв
        public static bool CheckDataGridTextBox(String s)
        {
            bool check = false;
            if (s.Length >= 3 && s != "")
                check = true;
            return check;
        }

// Перевірка відповідності числа шкалі оцінювання
        public static bool CheckType(int num, string type)
        {
            bool check = true;
            if (type == "Sati")
                if (num > 9)
                    check = false;
            return check;
        }
// Перевірка ведених у рядок значень в МПП
        public static bool CheckIsLetter(String s)
        {
            bool check = true;
            foreach (char c in s)
                if (Char.IsLetter(c) || c == '.' || c == '\\' || c == ' ')
                    check = false;
            return check;
        }
```

```
// Перевірка правильності заповнених значень в МПП
        public static List<string> ValidationArray(string[,] array, int size)
        {
            List<string> wrongType = new List<string>();
            for (int i = 0; i < size; i++)
            {
                for (int j = i; j < size; j++)
                {
                    string tempS = array[i, j];
                    if (tempS != "" && tempS != null)
                    {
                        if (CheckIsLetter(tempS))
                        {
                            int number = 0;
                            if (Int32.TryParse(tempS, out number))
                            {
                                if (!CheckType(number, "Sati"))
                                    wrongType.Add(number.ToString());
                            }
                            else
                            {
                                int first =
Convert.ToInt32(tempS.Split('/')[1]);
                                int second =
Convert.ToInt32(tempS.Split('/')[0]);
                                if (first > 9 || second > 9)
                                    wrongType.Add(tempS);
                            }
                        }
                    }
                    tempS ="";
                }
            }
            return wrongType;
        }
       }
```

Методи класу IValidation наведені в табл. 3.4.

Таблиця 3.4 – Методи класу IValidation

№	Метод	Опис
1	public static bool CheckComboBoxValue(Object o)	Метод перевіряє на наявність даних в полях ComboBox;
2	public static bool CheckDataGridTextBox (String s)	Перевіряє валідність назви критеріїв

Продовження таблиці 3.4

№	Метод	Опис
3	public static bool CheckType (int num, string type)	Перевіряє відповідність числа шкалі оцінювання
4	public static bool CheckIsLetter(String s)	Перевіряє правильність заповненого значення в МПП
5	public static List<string> ValidationArray (string[,] array, int size)	Перевіряє відповідність МПП при переході від однієї шкали до іншої, в разі коректності даних, повертає список значень

3.2.4 Клас Figures

Клас Figures призначений для візуалізації отриманих глобальних пріоритетів за допомогою графа, вузли якого є критеріями відбору, а «листя» - альтернативами.

Клас містить набір атрибутів, що характеризують елементи графа. Значення атрибутів екземплярів класу Figures, що були отримані в результаті обчислення глобальних пріоритетів використовуються для побудови і відображення ієрархічної структури.

Скорочений програмний код класу IValidation з необхідними коментарями наступний:

```
public class Figures
    {
        private float x;
        private float y;
        private float width;
        private float height;
        private float wall;
// Параметри фігур, що візуалізуються за допомогою класу Figures
        public Figures(float x, float y, float width, float height, float
wall)
        {
            this.x = x;
            this.y = y;
            this.width = width;
```

```
            this.height = height;
            this.wall = wall;
        }
        public float X { get => x; set => x = value; }
        public float Y { get => y; set => y = value; }
        public float Width { get => width; set => width = value; }
        public float Height { get => height; set => height = value; }
        public float Wall { get => wall; set => wall = value; }
// Призначений для візуалізації відповідної кількості критеріїв
   public List<float[]> drawCrits(int count, Figures goal)
        {
            float[] XY = new float[2];
            List<float[]> rounds = new List<float[]>();
            Figures crit = new Figures(goal.X + goal.Width / 2 - 25, 150, 50,
50, 4);
            if (count % 2 == 1)
            {
                drawRound(count / 2 + 1, crit, goal);
                XY[0] = crit.X; XY[1] = crit.Y;
                rounds.Add((float[])XY.Clone());
                for (int i = count / 2; i >= 1; i--)
                {
                    crit.X = crit.X - 100;
                    drawRound(i, crit, goal);
                    XY[0] = crit.X; XY[1] = crit.Y;
                    rounds.Add((float[])XY.Clone());
                }
                crit.X = goal.X + goal.Width / 2 - crit.Width / 2;
                for (int i = count / 2 + 2; i <= count; i++)
                {
                    crit.X = crit.X + 100;
                    drawRound(i, crit, goal);
                    XY[0] = crit.X; XY[1] = crit.Y;
                    rounds.Add((float[])XY.Clone());
                }
            }
            else
            {
                float temp = goal.X + goal.Width / 2 - crit.Width / 2 + 50;
                crit.X = temp;
                for (int i = count / 2; i >= 1; i--)
                {
                    crit.X = crit.X - 100;
                    drawRound(i, crit, goal);
                    XY[0] = crit.X; XY[1] = crit.Y;
                    rounds.Add((float[])XY.Clone());
                }
                crit.X = temp - 100;
                for (int i = count / 2 + 1; i <= count; i++)
                {
                    crit.X = crit.X + 100;
                    drawRound(i, crit, goal);
                    XY[0] = crit.X; XY[1] = crit.Y;
                    rounds.Add((float[])XY.Clone());
                }
            }
```

```
            return rounds;
        }
// Призначений для відображення критеріїв у вигляді кола
public void drawRound(int num, Figures crit, Figures goal)
        {
            Bitmap bmp = new Bitmap(pBoxGraph.Image);
            Graphics g = Graphics.FromImage(bmp);
            Pen pen = new Pen(Color.OrangeRed, crit.Wall);
            g.DrawEllipse(pen, crit.X, crit.Y, crit.Width, crit.Height);
            pen.Dispose();
            g.Dispose();
            pBoxGraph.Image = bmp;
            drawRoundLabel(num, crit);
            drawLine(crit, goal);
        }
        public void drawRoundLabel(int i, Figures crit)
        {
            Bitmap bmp = new Bitmap(pBoxGraph.Image);
            Graphics g = Graphics.FromImage(bmp);
            System.Drawing.Font titlefont = new System.Drawing.Font("Consolas",
(float)36 / 100 * crit.Width);
            g.DrawString("K" + i, titlefont, Brushes.Black, crit.X + (float)15
/ 100 * crit.Width, crit.Y + (float)22 / 100 * crit.Width);
            g.Dispose();
            pBoxGraph.Image = bmp;
        }
// Призначений для відображення ліній графу
        public void drawLine(Figures crit, Figures goal)
        {
            Bitmap bmp = new Bitmap(pBoxGraph.Image);
            Graphics g = Graphics.FromImage(bmp);
            Pen pen = new Pen(Color.Black, 3);
            g.DrawLine(pen, goal.X + goal.Width / 2, goal.Y + goal.Height + 2,
crit.X + crit.Width / 2, crit.Y);
            pen.Dispose();
            g.Dispose();
            pBoxGraph.Image = bmp;
        }
// Призначений для з’єднання критеріїв з альтернативами
public void drawCFLines(float[] crit, float[] fact)
        {
            Bitmap bmp = new Bitmap(pBoxGraph.Image);
            Graphics g = Graphics.FromImage(bmp);
            Pen pen = new Pen(Color.Black, 2);
            g.DrawLine(pen, crit[0] + 25, crit[1] + 50, fact[0] + 25,
fact[1]);
            pen.Dispose();
            g.Dispose();
            pBoxGraph.Image = bmp;
               }

       }
```

Параметри фігур, що візуалізуються за допомогою класу Figures наведено у табл. 3.5.

Таблиця 3.5 – Параметри фігур

№	Параметр	Опис
1	private float x	Координата по x
2	private float y	Координата по y
3	private float width	Ширина фігури
4	private float height	Висота фігури
5	private float wall	Товщина виділення

Основні методи класу Figures наведені в табл. 3.6.

Таблиця 3.6 – Основні методи класу Figures

№	Метод	Опис
1	public List<float[]> drawCrits(int count, Figures goal)	Візуалізує відповідну кількість критеріїв
2	public void drawRound(int num, Figures crit, Figures goal)	Відображає критерії у вигляді кола
3	public List<float[]> drawFacts(int count, Figures goal, int finded)	Візуалізує відповідну кількість альтернатив
4	public void drawSquare(int num, Figures fact, Figures goal, int finded)	Відображає альтернативу у вигляді квадрата
5	public void drawCFLines (float[] crit, float[] fact)	З’єднує критерії та альтернативи

Програмний код основних класів розробленого застосунку наведено у Додатку А.

3.3 Висновки до третього розділу

Проведено моделювання програмного забезпечення, що розробляється у вигляді застосунку AHP. Розроблено дві моделі в нотації мови UML: діаграма прецендентів, що описувала взаємодію користувача з системою AHP та діаграма класів цієї системи.

На основі діаграми прецедентів і послідовності кроків методу класичного MAI був розроблений алгоритм роботи програми AHP. Представлений опис розробки класів та їх основних методів для програми AHP, необхідних для автоматизації роботи проєктувальника при виборі альтернативних варіантів проєктних рішень.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

У даному розділі вирішується завдання валідації розробленого програмного забезпечення (програми Analytic Hierarchy Process) на прикладі ГЕО вибору шаблонів компоновки екранних форм програмних інтерфейсів за критеріями якості.

4.1 Приклад вибору альтернатив та критеріїв оцінювання

У якості приклада розглянемо вирішення задачі вибору оптимального шаблону компонування сторінок для інтерфейсу екранних форм програми, яка була реалізована в рамках даної магістерської роботи для вирішення завдань, пов'язаних з методом аналізу ієрархій.

Для її вирішення необхідно виділити деякі альтернативи шаблонів компонування, а також кілька критеріїв, за якими обрані альтернативи будуть оцінюватися.

4.1.1 Вибір проєктних альтернатив

Після аналізу можливостей інструментальних елементів одного з інтерфейсів програмування застосунків Microsoft .NET Framework Windows Forms, на основі якого створювалася програма, були визначені наступні альтернативи шаблонів компонування:

- Center Stage (Центральна сцена) - A1;
- Visual Framework (Візуальна схема) - A2;
- Card Stack (Пачка карток) - A3;
- Movable Panels (Панелі, що переміщуються) - A4.

Шаблон Center Stage (рис. 4.1) використовується для демонстрації користувачеві логічно пов'язаного вмісту, дозволяє йому відредагувати

документ або виконати певне завдання. «Центральну сцену» можна використовувати в більшості програм – сюди входять текстові таблиці, форми, веб-сторінки і графічні редактори.

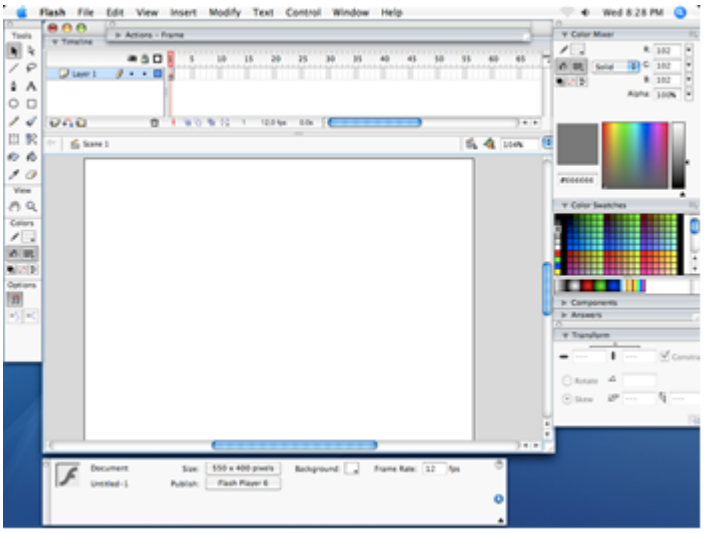


Рисунок 4.1 – Приклад шаблону «Центральна сцена» [44]

Погляд користувача необхідно відразу ж привернути до початку найбільш важливої інформації. Чітко виражений центральний елемент приковує увагу.

Шаблон Visual Framework використовується переважно при створенні веб-сайтів з великою кількістю сторінок або користувальницького інтерфейсу з багатьма вікнами, тобто практично в будь-якому складному програмному забезпеченні. Використання інтерфейсу і навігація повинні бути простими.

Для дизайну кожної сторінки використовується один і той самий базовий макет, набір кольорів та таблиць стилів, щоб забезпечити достатню динамічність та поміщати на сторінку різний вміст.

Прикладом даного шаблону може бути загальне представлення вікон у ОС оскільки колір, шрифт і схема стандартна. Наприклад вікна параметрів сторінки та пошукова система Google (рис. 4.2).

Рисунок 4.2 – Приклад шаблону «Візуальна схема»

Коли в інтерфейсі використовується один колір, шрифт, макет сторінки і посилання що знаходиться на одному і тому місці, користувачі знають де вони знаходяться і де шукати те що їм потрібно. Їм не потрібно знайомитися з новим інтерфейсом при кожному переключенні контексту, чи переході з однієї сторінки на іншу.

Шаблон Card Stack (рис. 4.3) використовується у тому випадку, коли на сторінці дуже багато матеріалу. Множина елементів управління або блоків тексту розподілені по всьому інтерфейсу і їх неможливо організувати в жорстку структуру: увага користувача розсіюється.

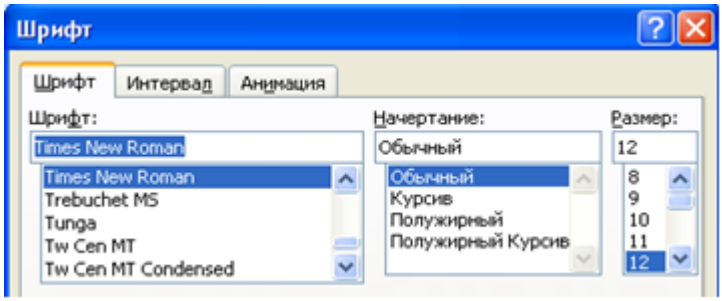


Рисунок 4.3 – Приклад шаблону «Пачка карток» [44]

Структура з іменованими «картками» дозволяє розбити вміст на зрозумілі розділи, уявлення про кожний з яких можна буде скласти з одного погляду.

Шаблон Movable Panels (рис. 4.4) використовується у тому випадку, коли на сторінці є кілька пов'язаних «фрагментів» інтерфейсу, які не обов'язково повинні знаходитися в одній загальній конфігурації.

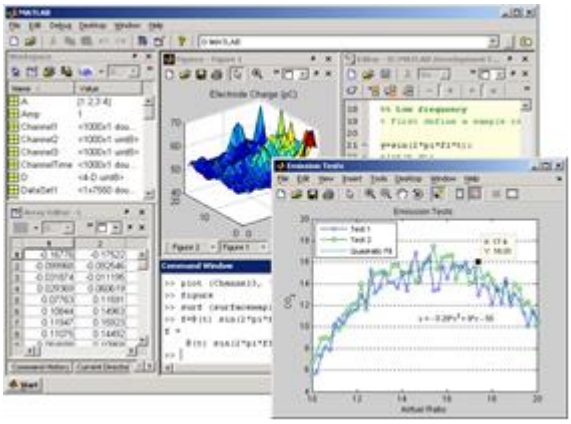


Рисунок 4.4 – Приклад шаблону «Панелі, що переміщуються» [44]

Коли користувачам доводиться довго над чим-небудь працювати в них виникає бажання реорганізувати середовище для оптимальної відповідності їх своєму стилю роботи. Вони можуть помістити більш необхідні інструменти ближче до місця роботи, приховати непотрібні речі.

4.1.2 Вибір критеріїв оцінювання

Згідно проведеному у п. 2.1 аналізу стандартів якості програмних засобів виявлено, що стандарт ISO/IEC 25010.2–2008 регламентує наступні групи характеристик: зручність і простота у використанні, яка визначається тими ж характеристиками, що й у стандарті ISO 9241-11 (ступінь ефективності, продуктивність і задоволеність); гнучкість у використанні, яка є мірою придатності програмного продукту до застосування, включаючи доступність та безпечність.

Відповідно до стандарту ISO/IEC 25010.2-2008, «зручність і простота використання» (operability) – це властивість ПП, що характеризує ступінь зрозумілості, навченості та привабливості ПП для користувача, при його використанні в зазначених умовах.

Для оцінювання проєктних альтернатив шаблонів проєктування з цього аспекту, згідно даних рис. 2.4, було обрано таку характеристику якості як продуктивність.

Згідно моделі якості програмного продукту (див. рис. 2.3), підхарактеристиками виконавчої продуктивності є:

– часоємність (time behaviour), тобто сукупність властивостей ПРЗ, що характеризують забезпечувані при його функціонуванні час реакції на запити, швидкість обробки даних і пропускну здатність;

– використання ресурсів (resource utilisation), тобто здатність вирішувати потрібні задачі з використанням певних обсягів ресурсів (оперативна й довгострокова пам'ять, мережні з'єднання, обладнання вводу й виводу тощо).

Згідно цієї ж моделі якості програмного продукту, для оцінювання проєктних альтернатив шаблонів проєктування було обрано наступні підхарактеристики «зручності та простоти використання»:

– правильна розпізнаваність (appropriateness recognisability), тобто властивість ПП, що характеризує ступінь визнання користувачами програмного продукту таким, що підходить для задоволення їх потреб;

– опанованість (learnability), тобто показник, обернений зусиллям, що витрачається користувачами на опанування правил застосування ПРЗ;

– легкість використання (ease of use), тобто властивість ПП, що характеризує ступінь легкості роботи й керування їм користувачем.

Обрані характеристики будуть виступати в якості критеріїв оцінювання проєктних альтернатив шаблонів та якості програмного інтерфейсу програми Analytic Hierarchy Process.

4.2 Приклад використання системи Analytic Hierarchy Process

Після завантаження програми відкривається форма для введення кількості альтернатив і кількості критеріїв оцінювання (рис. 4.5).

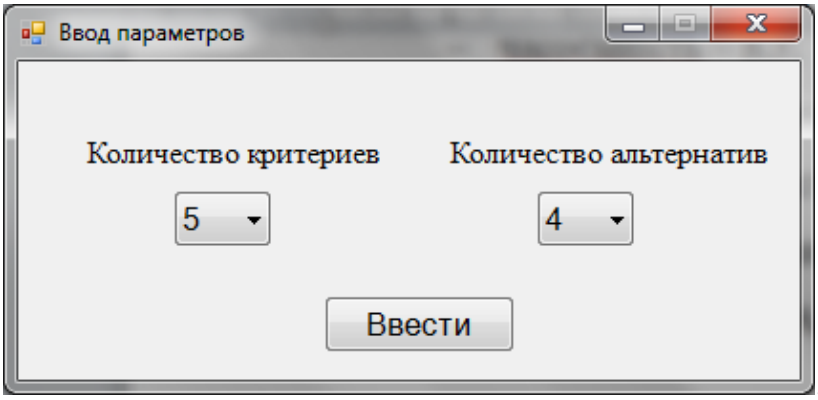


Рисунок 4.5 – Форма для введення кількості критеріїв і альтернатив

Після введення даних відкривається форма для введення назв критеріїв для оцінювання альтернатив (рис. 4.6). Після заповнення рядків введення і натиснення кнопки "Далее" відкривається вкладка з МПП критеріїв одного рівня (рис. 4.7).

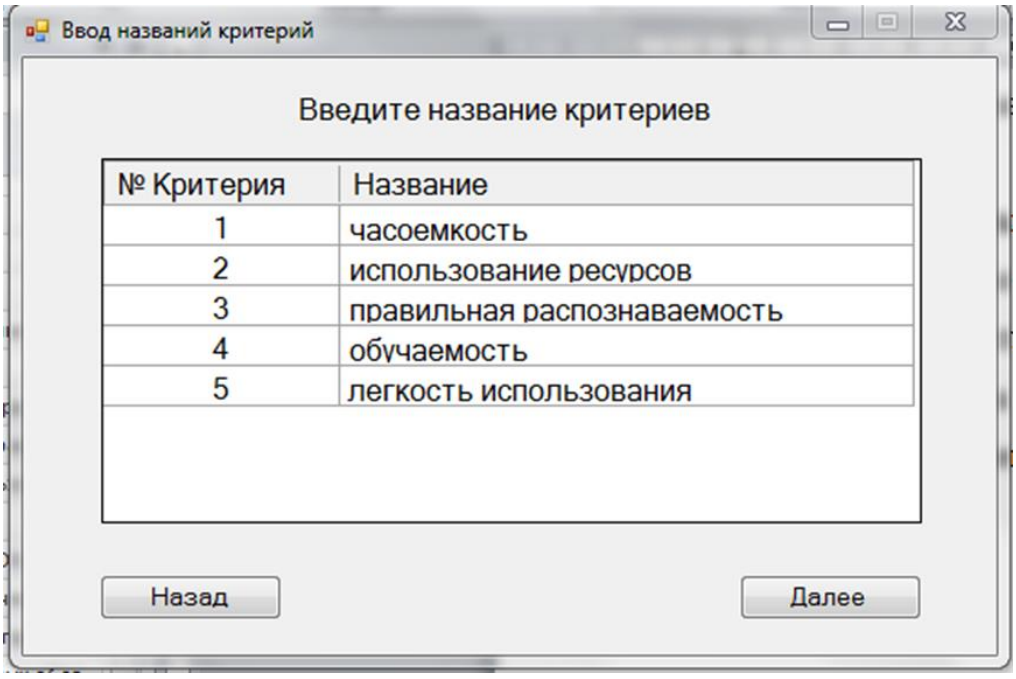


Рисунок 4.6 – Форма для введення назв критеріїв

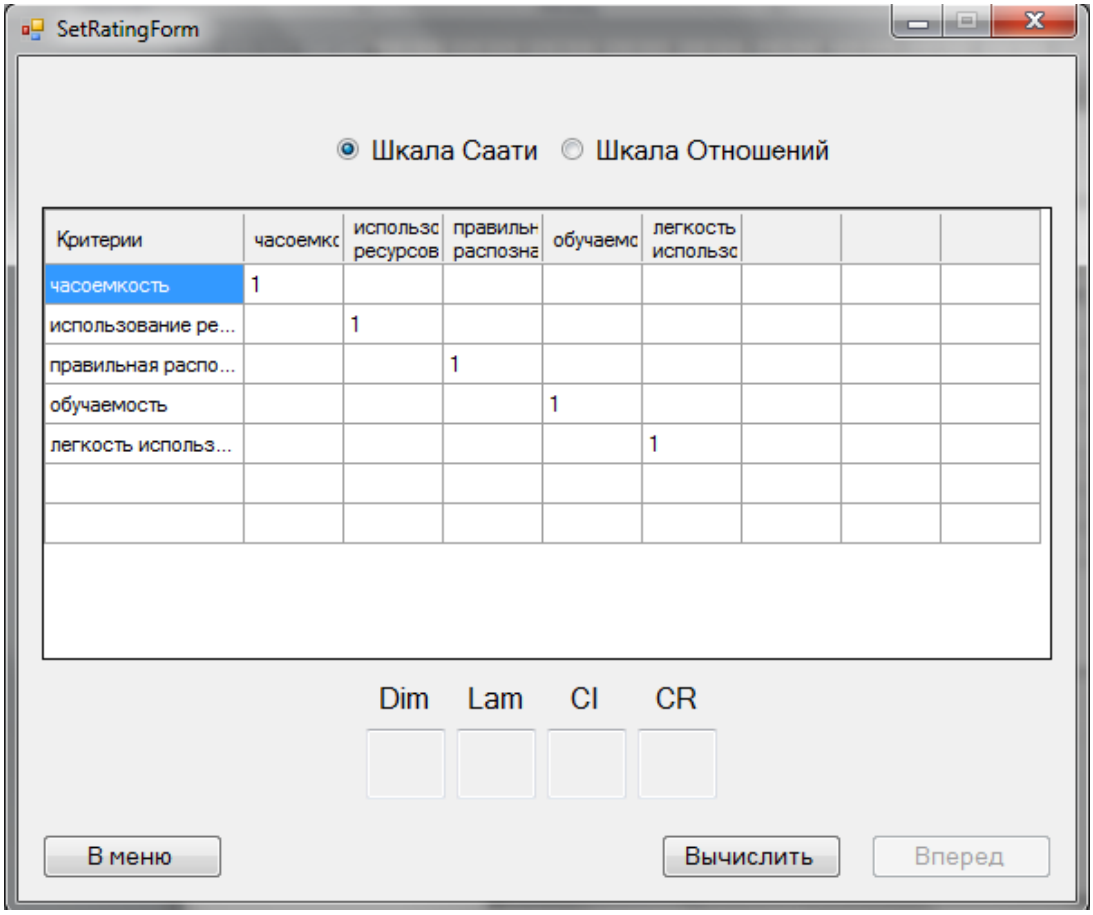


Рисунок 4.7 – Вкладка критеріїв МПП

Кожний критерій необхідно попарно порівняти між собою за допомогою шкали Сааті або шкали відношень. Попарні порівняння критеріїв на основі шкали Сааті (табл. 2.2) допускають значення тільки від 1 до 9 та зворотні до них значення від 1 до 1/9 відповідно, а значення шкали відношень допускають будь-які натуральні числа або значення у вигляді дробу правильних або неправильних дробів.

На думку експерта, що проводить порівняння, значення попарних порівнянь критеріїв між собою має вигляд, зображений на рис 4.8.

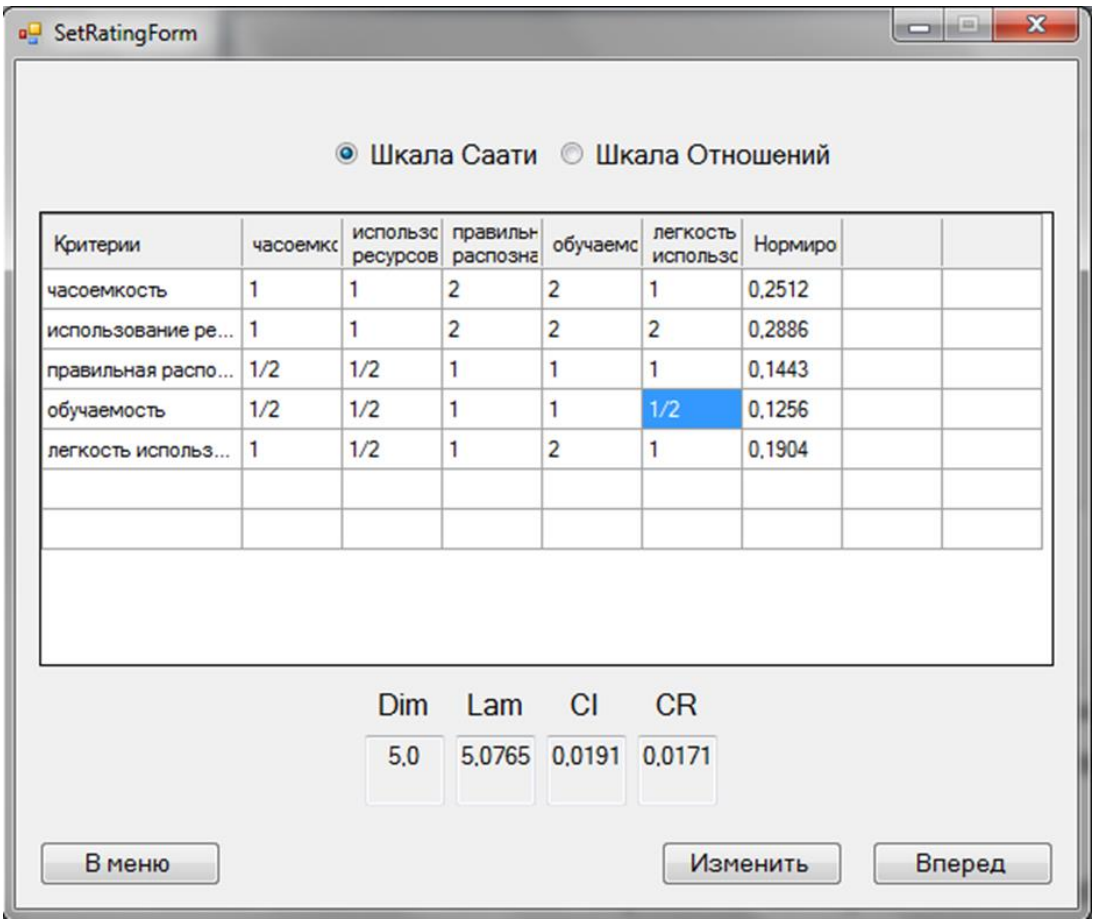


Рисунок 4.8 – Вкладка МПП критеріїв з введеними значеннями

Після введення значень переваг в комірки МПП відбувається їх верифікація, якщо значення не вірно, з'являється повідомлення про помилку з проханням повторити введення. Після натиснення кнопки "Вычислить" вона змінюється на «Изменить» для можливості внесення змін і стає активною кнопка «Вперед», а також відбувається розрахунок індексів.

Далі для всіх зазначених критеріїв проводиться аналіз для кожної альтернативи. В даному прикладі, зокрема, аналіз був проведений наступним чином.

4.2.1 Аналіз критерію «Часоємність»

Критерій «Часоємність» для даних альтернатив полягає у середньому значенні кількості секунд, витрачених у ході виконання програми від моменту,

як користувач побачив початкове вікно програми до отримання бажаного результату за умови того, що користувач не витрачає час на роздуми на попарне порівняння критеріїв або альтернатив. Оскільки кількість часу, що необхідний для взаємодії користувача з інтерфейсом, напряму залежить і від кількості критеріїв та альтернатив, проаналізуємо шаблони для однакової кількості критеріїв та альтернатив. Для аналізу критерію часоємності також зручно застосувати шкалу відношень і занести в таблицю кількість часу, витраченого на взаємодію з інтерфейсом, побудованого за відповідним шаблоном, причому в зворотному порядку, оскільки чим менше часу необхідно користувачу для виконання однакових дій, тим краще (рис 4.9).

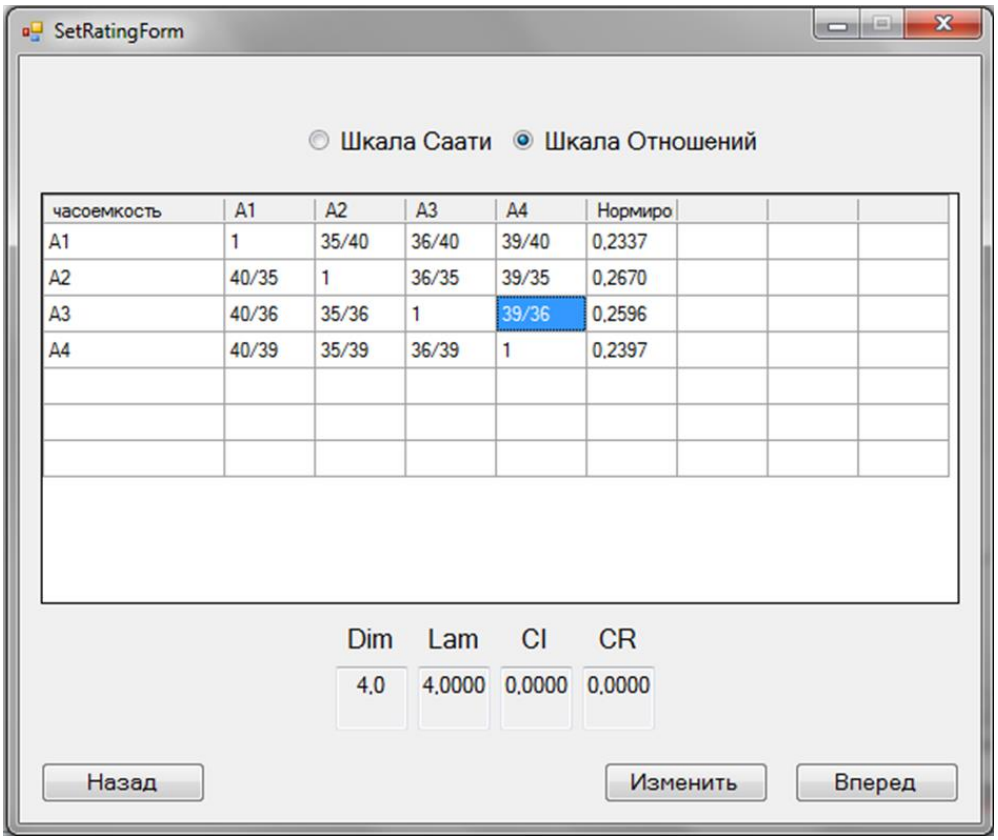


Рисунок 4.9 – Вкладка МПП для критерію «Часоємність»

4.2.2 Аналіз критерію «Використання ресурсів»

Критерій ресурсоємності, або використання ресурсів для даних альтернатив полягає у середньому значенні об’єму оперативної пам’яті, яка

необхідна під час взаємодії користувача з інтерфейсом. Для цього критерію зручно застосувати шкалу відношень і занести в таблицю кількість кілобайтів оперативної памяті, які використовує той чи інший інтерфейс, побудований за відповідним шаблоном, при чому в зворотньому порядку, оскільки чим менше ресурсів використовує програма для реалізації однакових функцій, тим краще (рис 4.10).

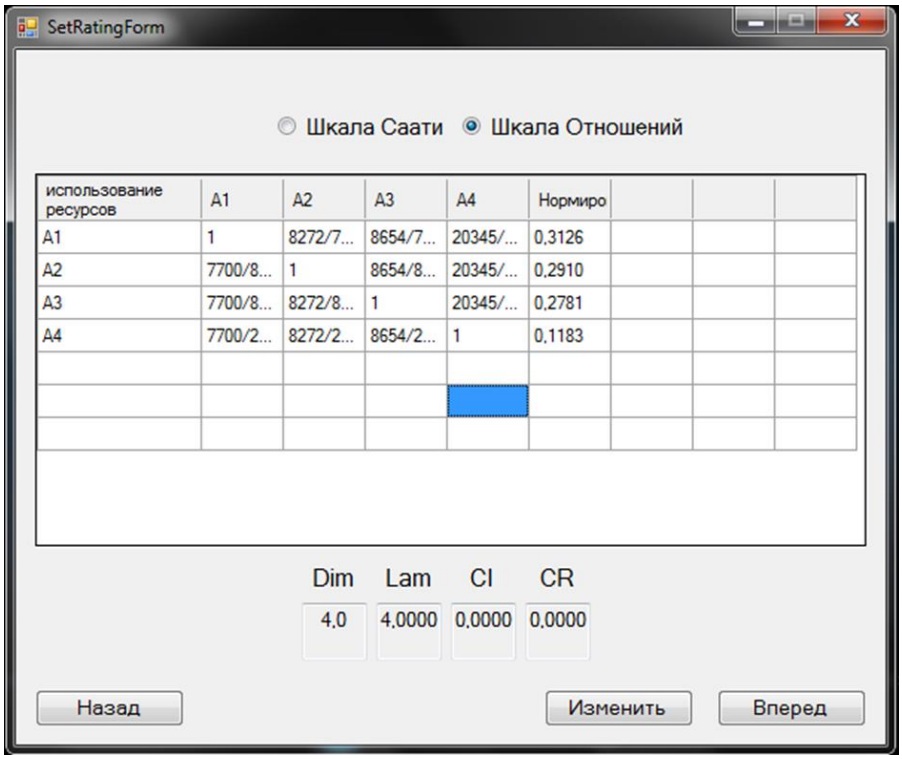


Рисунок 4.10 – Вкладка МПП для критерію «Використання ресурсів»

4.2.3 Аналіз критерію «Правильна розпізнаваність»

Критерій «Правильна розпізнаваність» полягає у правильній ідентифікації користувачем даних, яких система очікує до введення. Зокрема, шаблон «Пачка карток» і «Центральна сцена» надає користувачу можливість бачити всі критерії одразу, що є одним із зручних елементів для правильності розпізнавання, а шаблон «Панелі, що переміщуються» програє у цій оцінці, оскільки наявність великої кількості вікон може заплутати користувача. Оцінка експертом попарних порівнянь зображена на рис. 4.11.

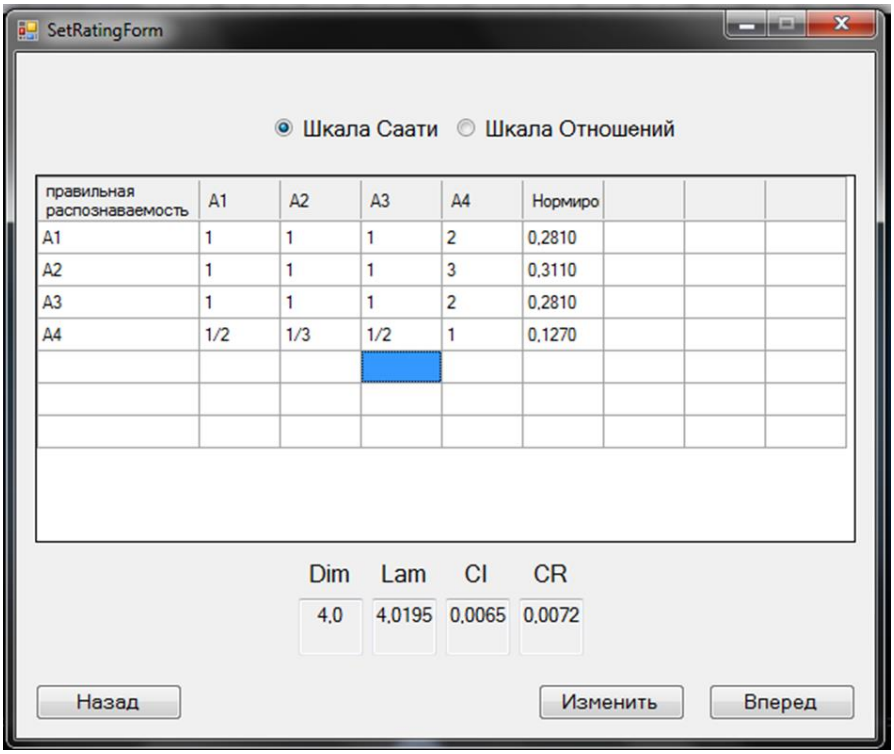


Рисунок 4.11 – Вкладка МПП для критерію «Правильна розпізнаваність»

4.2.4 Аналіз критерію «Опанованість»

Програма вимагає певного алгоритму типових дій користувача: ввести дані, проаналізувати критерії, проаналізувати альтернативи по кожному критерію, отримати локальний і глобальний пріоритет і графічне зображення, згенерувати файл з результатами виконання методу аналізу ієрархій. Знання і розуміння цього алгоритму істотно спрощує роботу з розробленою системою. Тому альтернатива «Центральна сцена» програє «Візуальній схемі» 1/3, оскільки дотримання загального стилю і типових елементів навігації підвищує наочність програми і здатність до навчання користувача в порівнянні з особливостями «Центральної сцени». Аналогічне ставлення спостерігається між «Центральної сценою» і «Пачкою карток», які надають розподіл змісту на окремі панелі. Однак порівнюючи «Центральну сцену» з «Панелі, що переміщуються» можна віддати перевагу 2/1, оскільки більшу частину інтерфейсу займає типовий елемент DataGrid, в якому змінюється тільки зміст, а застосування шаблону «Movable Panels» призвело б в такому випадку до

надмірного захаращення робочого простору програми. Порівнюючи «Візуальну схему» з «Пачкою карток» можна зробити висновок, що для наочності, а, відповідно, і для опанованості, обидва шаблону є рівнозначними. А якщо порівнювати «Візуальну схему» з «Панелі, що переміщуються», то очевидним є вибір «Візуальної схеми» (5/1). Наочність шаблону «Пачка карток» також вище наочності альтернативи А4, тому тут маємо значення 4/1 (рис. 4.12).

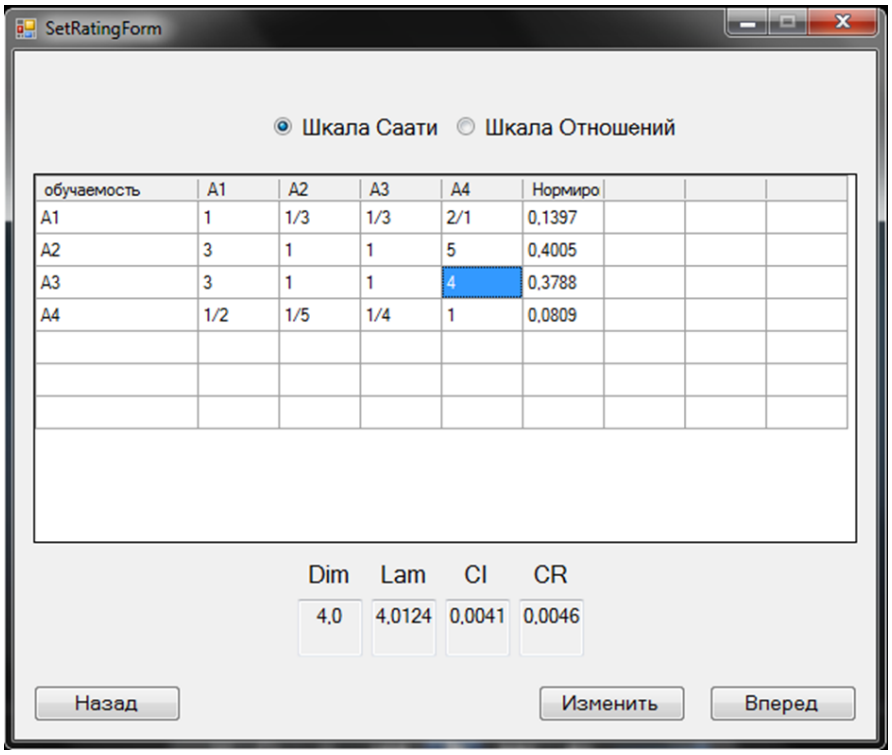


Рисунок 4.12 – Вкладка МПП для критерію «Опанованість»

4.2.5 Аналіз критерію «Легкість використання»

Критерій «Легкість використання» полягає зручній для користувача взаємодії з інтерфейсом системи. Зокрема, шаблон «Візуальна схема» з дотримання загального стилю і типових елементів навігації є зручнішим, ніж шаблон «Панелі, що переміщуються», що програє у цій оцінці, оскільки наявність великої кількості вікон може заплутати користувача. Оцінка експертом попарних порівнянь зображена на рис. 4.13.

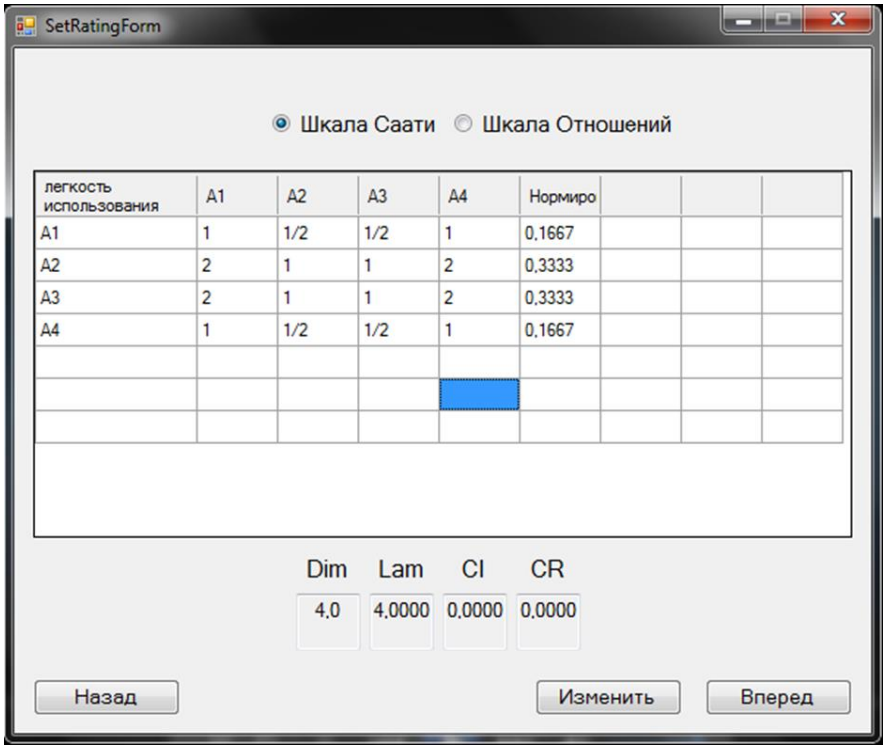


Рисунок 4.13 – Вкладка МПП для критерію «Легкість використання»

Після введення значень оцінок в МПП в усіх вкладках, відкривається вкладка з розрахованими локальними і глобальними векторами пріоритетів для кожної з альтернатив (рис. 4.14).

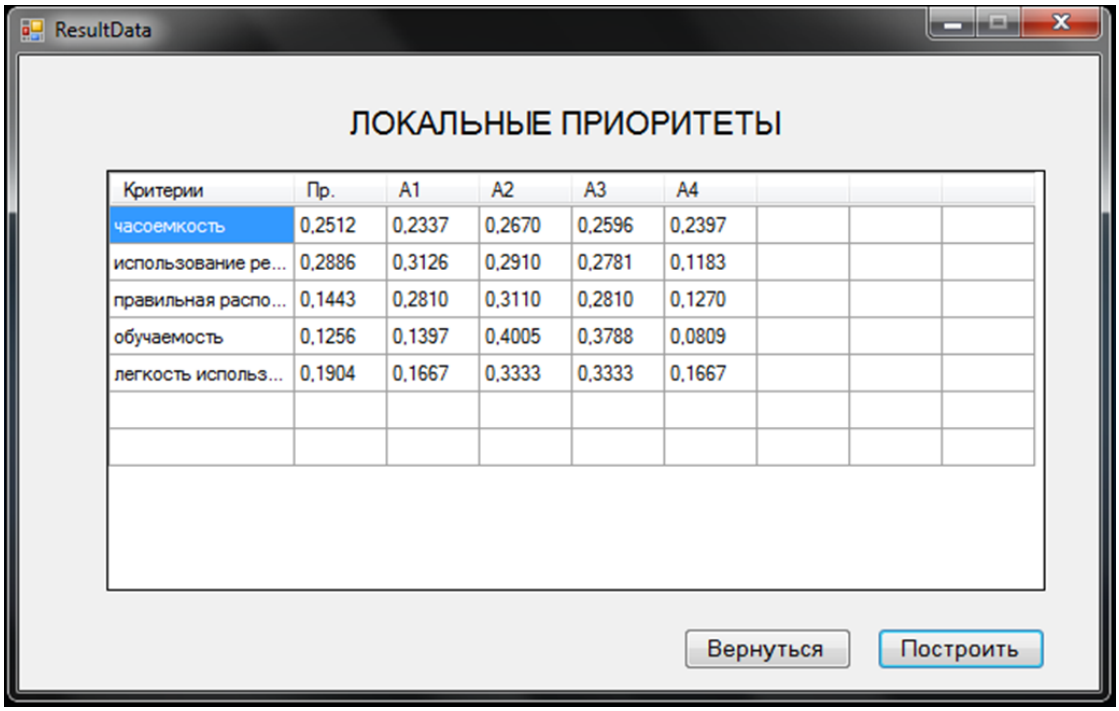


Рисунок 4.14 – Вкладка з локальними пріоритетами

Для формування ієрархії альтернатив і критеріїв користувачеві необхідно натиснути на кнопку "Построить". Результатом буде форма, на якій відображається побудована ієрархія критеріїв і альтернатив (рис. 4.15).

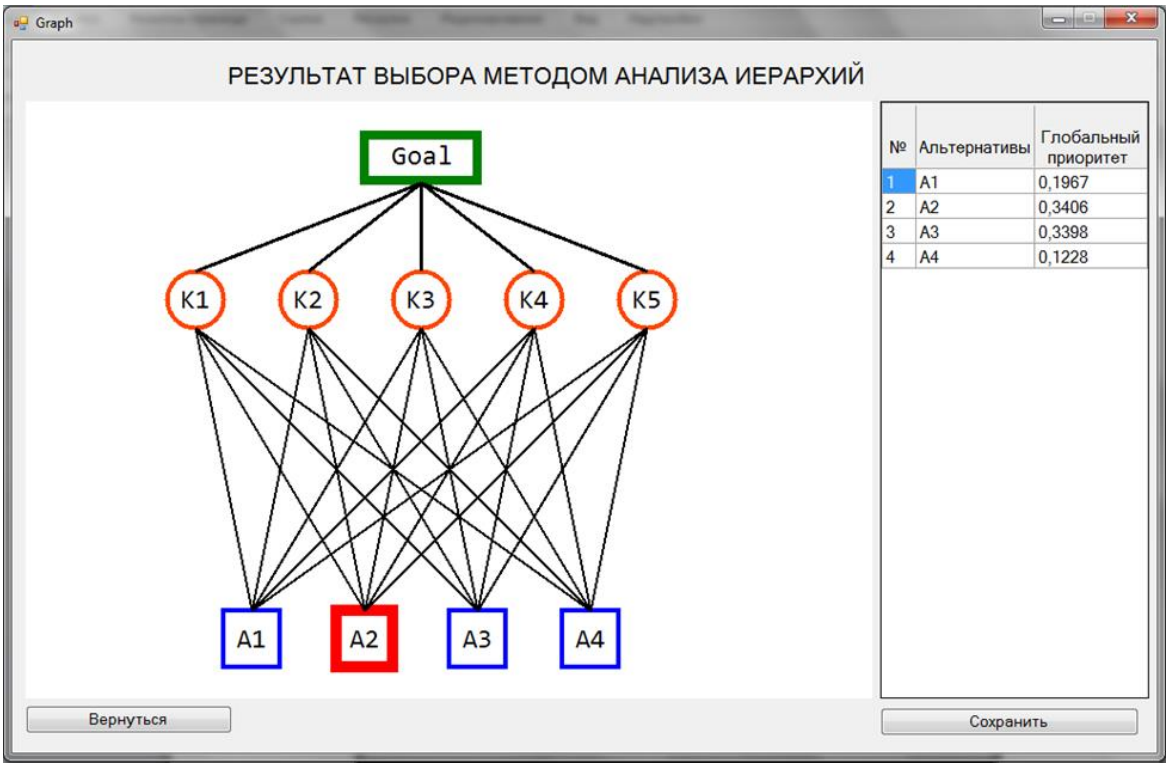


Рисунок 4.15 – Побудована ієрархія альтернатив і критеріїв

Максимальне значення глобального вектору пріоритетів відповідає альтернативі №2 "Візуальна схема". Таким чином, ця альтернатива є оптимальним рішенням для даного набору оцінюючих критеріїв. Результат ГЕО у вигляді побудованої ієрархії, зображений зліва на рис.4.15. Оптимальна альтернатива А2 виділена червоним кольором.

Далі шляхом натискання на кнопку «Сохранить» формується документ Word з результатами аналізу. Його вміст наведений у Додатку Б.

Таким чином, даний приклад демонструє спосіб вирішення завдачі вибору оптимального шаблону компонування сторінок інтерфейсу екранних форм програми. Якщо при розробці інтерфейсу проєктувальником було обрано шаблон «Візуальна схема» (з чотирьох альтернатив) - це рішення забезпечує найбільшу якість інтерфейсу.

4.3 Висновки до четвертого розділу

У даному розділі представлена методика оцінки якості проєктування інтерфейсів користувача шляхом застосування експертних методів прийняття рішень на ранніх етапах їх проєктування.

Послідовність кроків методики визначається, як описом, наведеним в п. 4.1, так і алгоритмом класичного MAI, який автоматизується за допомогою розробленого програмного забезпечення у вигляді програми Analytic Hierarchy Process.

Валідація розробленого програмного забезпечення, на прикладі ГЕО вибору шаблонів компоновки екранних форм програмних інтерфейсів за критеріями якості, показала що розроблений продукт здатен задовольняти вимогам щодо встановленого або передбаченого використання.

5 ЕКОНОМІКО-ОРГАНІЗАЦІЙНА ЧАСТИНА

Проєкт «Analytic Hierarchy Process», що розробляється у дипломній кваліфікаційній роботі магістра призначений для оптимізації процесу прийняття різних видів рішень при оцінці якості проєктування програмних інтерфейсів. Програмнийзастосунок покликаний автоматизувати роботу проєктувальника при виборі альтернативних варіантів проєктних рішень. Програма дозволяє скоротити час і трудомісткість процесу тестування програмних інтерфейсів, завдяки тому, що на ранніх етапах проєктування, на основі введених даних та відповідних параметрів для розрахунку отримуються математично обґрунтовані (оптимальні) проєктні рішення.

В даний час ринок подібного програмного забезпечення представлений дещо схожими програмними продуктами, однак вони спрямовані на інші предметні області, що не дозволяє використовувати їх у даному випадку, а отже є необхідність в розробці власного.

Економічна ефективність програми досягається за рахунок скорочення часу розрахунків на ЕОМ з 1000 до 450 годин на рік і автоматизації додаткової роботи по обслуговуванню аналітика

У розробці беруть участь програміст із заробітною платою 7000 грн. на місяць і консультант із заробітною платою 4500 грн. на місяць. На розробку проєкту було витрачено три місяці. Розробка програми розпочалася третього вересня 2020 року і має бути виконана до першого грудня 2020 року.

5.1 Планування розробки програмного продукту

Весь комплекс розробки програмного продукту можна розділити на етапи. Для кожного етапу вказуються трудомісткість, кількість виконавців і тривалість робіт [45]. Тривалість робіт визначають за формулою (5.1):

$$T_{\text{ц}} = \frac{Q}{R} \quad (5.1)$$

де $T_{\text{ц}}$ – тривалість циклу, днів;

Q - трудомісткість, людино-днів;

R - кількість виконавців, чол.

Отримана інформація зведена в таблицю 5.1.

Таблиця 5.1 - Завдання та обов'язки по розробці програми

Назва роботи	Трудомісткість		Виконавці	Трива-лість, днів
	люд.-дні	% до підсумку		
1. Аналіз предметної області (ПО)	10	9,3	програміст консультант	5
2. Визначення вимог до програмного продукту	20	18,5	програміст консультант	10
3. Проєктування структури програми	18	16,6	програміст консультант	9
4. Розробка схеми функціону-вання програми	10	9,3	програміст	10
5.Створення програмного коду	10	9,3	програміст	10
6.Тестування та налагодження програмного продукту	20	18,5	програміст консультант	10
7. Складання програмної документації	20	18,5	програміст консультант	5
Разом	108	100		64

За даними таблиці 5.1 складається зведений стрічковий графік планування розробки програмного продукту, який представляє собою таблицю, в першому стовпці якої розміщені в порядку збільшення термінів початку виконання всі види роботи, а навпаки - календарний період їх виконання. Даний графік наведено в таблиці 5.2.

Таблиця 5.2 – Зведений стрічковий графік планування розробки програмного продукту

Назва роботи	Календарний період, дні												
	03.09 - 07.09	10.09 - 14.09	17.09 - 21.09	24.09 - 29.09	01.10 - 05.10	08.09 - 12.10	16.10 - 20.10	22.10 - 26.10	29.10 - 02.11	05.11 - 09.11	12.11 - 17.11	19.11 - 23.11	26.11 - 30.11
1. Аналіз предметної області (ПО)	■												
2. Визначення вимог до програмного продукту		■	■										
3. Проєктування структури програми				■	■								
4. Розробка схеми функціонування						■	■						
5.Створення програмного коду								■	■				
6.Тестування та налагодження програмного продукту										■	■		
7. Складання програмної документації												■	■

■ - програміст ■ - консультант

5.2 Визначення витрат на розробку програми

Для визначення витрат на розробку програми складається калькуляція кошторисної вартості робіт, яка включає наступні статті:

- основна заробітна плата;

- додаткова заробітна плата;

- відрахування на єдиний соціальний внесок (ЄСВ);

- витрати на спеціальне обладнання (на оплату машинного часу ЕОМ для написання і налагодження програмних засобів);

- матеріали та комплектуючі вироби;

- накладні витрати;

- податки.

5.2.1 Розрахунок основної заробітної плати

Витрати за цією статтею складаються з планового фонду зарплати всіх категорій працівників, зайнятих в розробці програми. Розрахунок зарплати ведеться на підставі даних про трудомісткості, представлених в таблиці 5.3.

Таблиця 5.3 - Розрахунок основної заробітної плати

Посада виконавця	чисельність, чол.	місячний оклад, грн.	Кількість місяців роботи	Сума ЗП, грн.
Програміст	1	7000	3	21000
Консультант	1	4500	2	9000
Разом	2			30000

5.2.2 Розрахунок додаткової заробітної плати

Додаткову заробітну плату приймають рівною 10% від основної заробітної плати працівників і розраховують за формулою (5.2):

$$ЗП_{доп} = ЗП_{осн} \cdot 0{,}1. \quad (5.2)$$

Підставивши величину основної заробітної плати в дану формулу, отримуємо:

$$ЗП_{доп} = 30000 \cdot 0{,}1 = 3000 \text{ грн.}$$

5.2.3 Відрахування на єдиний соціальний внесок

Ці відрахування становлять 22% і беруться від сум основної та додаткової заробітної плати, формула (5.3):

$$ОТ = (ЗП_{осн} + ЗП_{доп}) \cdot 0{,}22. \quad (5.3)$$

Розраховуємо:

$$ОТ = (30000 + 3000) \cdot 0{,}22 = 7260 \text{ грн.}$$

5.2.4 Визначення витрат на матеріали

Використовується три найменування матеріалів: диск CD-R - 5 грн.; картридж - 850 грн. та папір 100 грн. (1 упаковка).

Витрати на матеріали розраховують за формулою (5.4):

$$М = \sum_{i=1}^{n} (Ц_i \cdot N_i \cdot (1 + К_{т.з}) - Ц_{io} \cdot N_{io}), \quad (5.4)$$

де $М$ – витрати на матеріали, покупні напівфабрикати і комплектуючі вироби,грн .;

$К_{т.з}$ - коефіцієнт, що враховує транспортно-заготівельні витрати;

$Ц_i$ - ціна *і*-го найменування матеріалу, напівфабрикату і комплектуючого, грн.;

N_i - потреба в *і*-му матеріалі, напівфабрикаті і комплектуючому;

$Ц_{io}$ - вартість зворотних відходів *і*-го найменування матеріалу, грн;

N_{io} - кількість зворотних відходів *і*-го найменування;

n - кількість найменувань матеріалів, напівфабрикатів та комплектуючих.

Проводимо розрахунок і отримуємо:

$$Ц_{io} = 0;\ N_{io} = 0;\ K_{т.з} = 0$$

$$М = (1 + 0{,}05) \cdot (5 + 850 + 100) = 1002{,}75 \text{ грн.}$$

Разом, витрати на матеріали становлять 1002,75 грн.

5.2.5 Витрати на спеціальне обладнання

У цю статтю входять витрати на придбання, транспортування, монтаж і налагодження нестандартного обладнання.

Практично, в даному випадку, в цій статті враховуються витрати на оплату машинного часу ЕОМ для написання і налагодження даної програми. Для чого необхідно скласти кошторис «витрат на утримання та експлуатацію обладнання» виходячи з якої визначиться вартість одного машино-години роботи ПК, після множення якої на машинний час пішло на написання і налагодження програми отримаємо витрати на оплату машинного часу.

Амортизаційні відрахування визначають за формулою (5.5):

$$А = \Ф_{в} \cdot \frac{Н_{а}}{1000}, \quad (5.5)$$

де $\Ф_{в}$ – балансова вартість обчислювальної техніки, грн .;

$Н_{а}$ – норма амортизаційних відрахувань на повне відновлення обчислювальної техніки,%.

Балансова вартість обчислювальної техніки становить 20000 грн.

Отримуємо:

$$А = 20000 \cdot 0{,}25 = 5000 \text{ грн.}$$

Статтю «Експлуатація обладнання» розраховують підсумовуванням витрат на електроенергію і допоміжні матеріали за формулою (5.6):

$$С_{э} = N_{Н} \cdot \Ф_{эф} \cdot К_{зв} \cdot К_{зм} \cdot Ц_{э}, \quad (5.6)$$

де $N_н$ - номінальна потужність ЕОМ, кВт;

$Ф_{эф}$- річний ефективний фонд часу роботи ЕОМ, машино-год;

$К_{зв}$ - середній коефіцієнт завантаження за часом;

$К_{зм}$ - коефіцієнт завантаження по потужності;

$Ц_э$ - ціна одного кВт ·год електроенергії, грн ./ (кВт год).

Номінальна потужність ЕОМ - 0,3 кВт. Річний ефективний фонд часу роботи ЕОМ становить 1800 годин. Середні коефіцієнти завантаження за часом і за потужністю рівні відповідно 0,9 і 0,6. Ціна однієї кіловат-години електроенергії складає 2,11 грн.

Отримуємо:

$$С_э = 0{,}2 \cdot 1800 \cdot 0{,}9 \cdot 0{,}6 \cdot 2{,}11 = 615{,}276 \text{ грн.}$$

Зарплата обслуговуючого персоналу розраховується за формулою (5.7):

$$ЗП_{обсл} = ФЗП_г \cdot \left(1 + К_{відрах.}\right) \cdot \frac{t_{обсл}}{Ф_{еф.обсл}}, \quad (5.7)$$

де $ФЗП_г$ – річний фонд заробітної плати (основної та додаткової) обслуговуючих робітників, грн.;

$К_{відрах.}$ – коефіцієнт, що враховує відрахування на соціальне страхування і в інші фонди;

$t_{обсл}$ – час протягом року, необхідне технічне обслуговування ЕОМ, годин / рік;

$Ф_{еф.обсл}$ – річний ефективний фонд часу обслуговуючого персоналу, годин / рік.

Місячна заробітна плата обслуговуючого персоналу становить 3723 грн., а річний фонд заробітної плати відповідно дорівнює 44676 грн. Річний ефективний фонд робочого часу обслуговуючого ПК працівника дорівнює 1750 год/рік. На обслуговування одного ПК витрачається по 1 годині на місяць, що в рік становить 12 годин.

Отримуємо:

$$ЗП_{обсл} = 44676 \cdot (1 + 0{,}22) \cdot 12/1750 = 373{,}95 \text{ грн.}$$

Стаття «Поточний ремонт обладнання» приймається рівною 3% від балансової вартості обладнання. Балансова вартість обчислювальної техніки становить 20000 грн. Тому сума витрат за статтею "Поточний ремонт обладнання" становить: 20000·0,03 = 600 грн.

Стаття «Інші витрати» приймається рівною п'яти відсоткам від суми всіх попередніх статей витрат на утримання і експлуатацію обладнання. Сума всіх попередніх статей дорівнює 6589,23 грн., 5% від суми складають 329,46 грн. Розраховані статті витрат на утримання і експлуатацію обладнання внесені в таблицю 5.4.

Таблиця 5.4 - Кошторис витрат на утримання і експлуатацію обладнання

Найменування статей витрат	Сума, грн.
Амортизація обладнання	5000
Експлуатація обладнання (крім витрат на поточний ремонт)	615,28
Заробітна плата основна і додаткова обслуговуючих робітників з відрахуваннями на соціальні заходи	373,95
Поточний ремонт обладнання	600
Інші витрати	329,46
Разом	6918,69

Витрати на оплату машинного часу ЕОМ для написання і налагодження програмних засобів визначаються за формулою (5.8):

$$С_{мо} = Р_{екс} \cdot t_{мо}\,, \qquad (5.8)$$

де $С_{мо}$ - витрати на оплату машинного часу, грн .;

$Р_{екс}$ - експлуатаційні витрати на одну годину машинного часу цієї цифрової ЕОМ, грн. / машино-год .;

$t_{мо}$ - машинний час цифрової ЕОМ для написання і налагодження даного програмного продукту, машино-год.

Експлуатаційні витрати на одну годину машинного часу використовуваної ЕОМ розраховують діленням суми витрат за кошторисом «Витрати на утримання та експлуатацію обладнання (ЕОМ)» (табл. 5.4) на річний ефективний фонд часу роботи ЕОМ. Річний ефективний фонд часу роботи ЕОМ дорівнює 1800 годин. В результаті експлуатаційні витрати на одну годину машинного часу рівні:

$$Р_{екс} = 6918,69/1800=3,84 \text{ грн. / машино–год.}$$

ЕОМ експлуатується 64 дня в одну зміну, що становить в сумі 512 годин. Таким чином, витрати на оплату машинного часу складуть:

$$С_{мо} = 3,84 \cdot 512=1966,08 \text{ грн.}$$

5.2.6 Розрахунок накладних витрат

До накладних витрат відносяться витрати на загальне управління і загальногосподарські потреби (заробітна плата апарату управління, канцелярські витрати і тому подібне), утримання та експлуатацію будівель. Накладні витрати включаються до вартості розробки програми непрямим шляхом - у відсотках до основної заробітної плати розробників. В даному випадку накладні витрати становлять 40% до основної заробітної плати розробників, що складає 12000 грн.

Результати визначення витрат на розробку програми у вигляді калькуляції кошторисної вартості робіт наведені в таблиці 5.5.

Таблиця 5.5 - Калькуляція кошторисної вартості робіт по розробці програми

№	Найменування статей витрат	Сума, грн	Питома вага до підсумку,%
1	Основна заробітна плата	30000,00	54,32
2	Додаткова заробітна плата	3000,00	5,432
3	ЄСВ	7260	13,15
4	Матеріали та комплектуючі	1002,75	1,82
5	Витрати на спец. обладнання	1966,08	3,56
6	Накладні витрати	12000	21,73
7	Разом	55228,83	100,00

5.3 Розрахунок економічної ефективності програмного продукту

Програма дозволяє скоротити час і трудомісткість процесу прийняття рішень шляхом побудови ієрархії критеріїв з подальшою оцінкою можливих альтернатив по кожному з обраних критеріїв. Знайдені в результаті роботи програми рішення будуть математично обґрунтованими, що може гарантувати отримання найбільш оптимального результату.

Економічна ефективність програми досягається за рахунок скорочення часу розрахунків на ЕОМ з 1000 до 450 годин на рік і автоматизації додаткової роботи по обслуговуванню аналітика, що в сумі призводить до економії заробітної плати. Річна економія по заробітній платі з урахуванням нарахувань становить, після розрахунку за формулою (5.9):

$$Э_{зп} = Т_е \cdot С_{зп} \cdot (1 + К_{отч}) \cdot (1 + К_{доп}), \quad (5.9)$$

де $Э_{зп}$ - річна економія по заробітній платі з відрахуваннями на соціальні заходи, грн .;

$С_{зп}$ - заробітна плата за одну годину роботи, грн.;

$Т_е$ - час який економиться при впровадженні програмного засобу;

$К_{доп}$ - коефіцієнт, що враховує додаткову заробітну плату.

В результаті отримуємо:

$$Э_{зп} = 550 \cdot 50 \cdot (1 + 0{,}22) \cdot (1 + 0{,}11) = 36905 \text{ грн.}$$

Річна економія визначається за формулою (5.10):

$$E = \left(T_{M_1} - T_{M_2}\right) \cdot В_{екс} + E_{н}^{а} \cdot \frac{Ф_{б.ктз} \cdot \left(T_{M_1} - T_{M_1}\right)}{Ф_{еф.ктз}} - \frac{S_{рп}}{Т_с} + Е_{зп}, \qquad (5.10)$$

де $T_{м_1}, T_{м_2}$ – машинний час, необхідний для розв'язання поставлених завдань, відповідно в старому та у новому варіантах, машино–год./рік;

$В_{екс}$ – експлуатаційні витрати, що доводяться на 1 годину машинного часу, грн/машино-год;

$E_{н}^{а}$ – нормативний коефіцієнт ефективності капітальних вкладень у засоби автоматизації;

$Ф_{б.ктз}$ – балансова вартість ЕОМ (КТЗ), грн.;

$Ф_{еф.ктз}$ – річний ефективний фонд часу роботи ЕОМ, машино–год.;

$S_{рп}$ – сумарні витрати на розробку програми, грн.;

$Т_с$ – термін служби впроваджуваної програми до її морального зношування, років. Приймаємо цей термін 5 років;

$Е_{зп}$ – економія на заробітній платі.

$$Е = (1000 – 450) \cdot 3{,}84 + 0{,}2 \cdot (20000 \cdot (1000 – 450)) / 1800 – 55228{,}83/5 + \\ +36905 = 29193{,}45 \text{ грн.}$$

Коефіцієнт ефективності капітальних вкладень $E_ф$ і термін їх окупності $T_{ок}$ розраховуються за формулами (5.11) та (5.12):

$$E_{\text{ф}} = \frac{E}{S_{\text{рп}}}, \quad (5.11)$$

$$T_{\text{ок}} = \frac{S_{\text{рп}}}{E}, \quad (5.12)$$

Таким чином:

$$E_{\text{ф}} = 29193,45 / 55228,83 = 0,53.$$

$$T_{\text{ок}} = 55228,83 / 29193,45 = 1,89 \text{ років.}$$

Зведемо отримані техніко–економічні показники в таблицю 5.6.

Таблиця 5.6 - Техніко-економічні показники

Показники	Сума	Одиниця виміру
Витрати на розробку програми	55228,83	грн.
Річна економія	29193,45	грн.
Термін окупності програми	1,89	рік

Термін окупності програми становить 1,89 року, що менше нормативного терміну окупності, який становить для машинобудування 5 років. Коефіцієнт ефективності капітальних вкладень дорівнює 0,53, що більше нормативного (0,2). Річна економія при використанні програми становить 29193,45 грн.

Таким чином, впровадження програмного продукту є економічно доцільним.

6 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

6.1 Аналіз потенційних небезпек

З розвитком науки та техніки всі сфери виробництва, змінюють предмети праці, технологію, якість, методи обробки та зберігання інформації. Також з розвитком йде удосконалення безпечності та універсальності місця праці. Але на сьогоднішній день у багатьох сферах виробництва місце праці працівника залишається небезпечним та шкідливим для його здоров’я.

При роботі, в офісному та серверному приміщенні людину оточують: обчислювальні пристрої, системи зв’язку, генератори струму та інші електричні прибори, що створюють електромагнітне випромінювання. На сьогодні вплив на організм людини електромагнітного випромінювання ще не зовсім вивчений.

Методологічною основою охорони праці є системний підхід до вивчення організації праці з точки зору її безпеки, функціонування системи взаємодії людини з середовищем, аналізу фізичних, хімічних, біологічних, психологічних та соціальних факторів безпеки виробничого процесу та особливостей впливу їх на організм людини.

Додержання правил техніки безпеки і виробничої санітарії залежить не тільки від виконання власником або уповноваженим ним органом своїх обов'язків, а й від того, наскільки кожний працівник знає і виконує їх під час роботи.

Дослідження, розробка та програмна реалізація групових методів прийняття рішень для оцінки діяльності проектувальника програмних інтерфейсів, передбачає роботу в серверному приміщені.

За природою виникнення шкідливі та небезпечні виробничі фактори поділяються на такі групи:

– фізичні (підвищена і знижена температура повітря, надмірна загазованість та запиленість повітря та ін.);

– біологічні (підвищений вміст у повітрі мікроорганізмів та ін.), хімічні (вміст у робочій зоні шкідливих речовин та ін.);

– психофізіологічні (нервово-емоційні перевантаження, розумове напруження та ін.);

Також потенційна небезпеки може бути пов’язана з порушеннями правил пожежної безпеки, наприклад: проблеми в організації заходів з пожежної безпеки, що можуть призвести до швидкого розповсюдження пожару. Проблеми в організації персоналу, можуть проявити себе і до інших проблем при прояві наслідків в надзвичайних ситуаціях.

У даному розділі проаналізовано наступні питання:

– електробезпека та пожежна безпека;
– склад повітря робочої зони та мікроклімат;
– шум та виробничі випромінювання;
– безпека щодо організації робочих місць;
– безпечність технологічного обладнання та процесу;
– виробниче освітлення.

6.2 Заходи із забезпечення безпеки

Обладнання, що використовується в приміщення обчислювального центру є споживачем електроенергії, що живиться від змінного струму 220 В, частотою стуму 50 Гц від мережі з заземленою нейтраллю, та відноситься до електроустановок до 1000 В закритого виконання.

Комп’ютери мають власні блоки живлення. Для перетворення змінного струму мережі у постійний комп’ютери використовують блоки живлення (5 В, 12 В), потужністю 350 – 1000 Вт. Так як усі прилади при підключенні до електричної мережі яка заземляється, використовують систему TN, тому обчислювальні пристрої у приміщенні відносяться до категорії I захисту від ураження електричним струмом.

Щоб зменшити ймовірність ураження людини електричним струмом живлення всіх електричних пристроїв здійснюється спеціальним кабелем, корпуси всіх електричних пристроїв виготовляються з неструмопровідних матеріалів. У приміщенні для забезпечення безпечності всі кабелі сховані в спеціальні коробки, штепсельні розетки розташовані на висоті одного та 0,2 метри від підлоги, вимикачі на стінах знаходяться на висоті 1,2 метра від підлоги, використовується електропроводка трифазна, чотирьохжильна, з подвійною ізоляцією, електрична мережа вмикається і вимикається за допомогою пускової апаратури (рубильником). Відповідно до міжнародного стандарту CE зазначені міри електробезпеки відповідають вимогам, пропонованим до побутових приладів [46].

Відповідно до ПУЕ [47] в приміщені розроблено заходи щодо забезпечення електробезпеки.

Також для запобігання ураження людини електричним струмом в приміщенні використовується:

– захисне заземлення;

– облаштування окремими розетками по 220 В кожного робочого місця з обчислювальним пристроєм;

– триполюсні вилки та розетки, для підключення обчислювальних пристроїв до струму, у яких відповідний полюс з'єднаний з шиною заземлення;

– мережевий фільтр до якого підключений кожен блок живлення обчислювального пристрою або периферійного пристрою;

– конденсатори для шунтування високочастотних перешкод живлячої мережі на землю через дріт захисного заземлення в мереживом фільтрі та при з'єднанні двох пристроїв (комп'ютера та принтера) інтерфейсним кабелем.

– окремий провід на загальний контор при з’єднані пристроїв для ліквідування проблеми різниці потенціалів.

– ізолююча підлога - лінолеум [48].

Опір захисного заземлення в обчислювальному центрі становить не більше 4 Ом згідно з «Правилами улаштування електроустановок» (ПУЕ) [47]. Також у приміщенні встановлено автоматичні вимикачі, пристрій заземлення контуру згідно з ГОСТ 12.2.007.0–75 «Изделия электротехнические. Общие требования безопасности» [49].

При виконанні робіт по ремонту і обслуговуванню ПЕОМ обслуговуючий персонал зобов'язаний керуватися «Правилами техніки безпеки при експлуатації електроустановок споживачами». До роботи не допускаються особи, які не пройшли навчання з техніки безпеки.

Втома та тунельний синдром також є розповсюдженим небезпечним фактором для здоров`я люди при роботі в обчислювальному центрі.

Для профілактики втоми робітників застосовуються специфічні методи, до яких можна віднести засоби відновлення функціонального стану зорового та опорно-рухового апарату, зменшення гіподинамії, підсилення мозкового кровообігу, оптимізацію розумової діяльності [46].

Також для профілактики втоми та запобігання статистичного навантаження робітників встановлені такі правила організації робочого місця:

– оптимальна висота клавіатури від підлоги – 65-75 см;

– можливість регулювання висоти і нахилу клавіатури (відстань від поверхні стола до середини клавіатури – не більше 30 мм, кут підйому клавіатури – від 2° до 15°);

– наявність у клавіатури підставки для рук та килимка для миші з захистом від тунельного синдрому;

– наявність стільця або крісла з підлокітниками;

– рекомендовано використовувати перерви в роботі 10 хв. через кожні дві години згідно ДНАОП 0.00-1.31-99 «Правила охорони праці під час експлуатації електронно-обчислювальних машин» [50].

6.3 Заходи з виробничої санітарії та гігієни праці

Відповідно до вимог ДСанПіН 3.3.2.007-98 «Державні стандартні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [50], «Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу» МЮУ 06.05.2014 р. за № 472/25249, і НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин» розроблені заходи щодо забезпечення виробничої санітарії та гігієни праці для обчислювального центру обладнаного обчислювальними пристроями з ВДТ [46].

Гігієнічна класифікація праці спрямована на гігієнічну оцінку умов та характеру праці на робочих місцях працівників та застосовуються на підприємствах, в установах, організаціях усіх форм власності у випадках, передбачених законодавством.

Так як робота програмістів в обчислювальному центрі виконуються сидячи і не потребує фізичного напруження, то вона відноситься до типу операторської роботи, даний тип роботи відносить до категорії 1а – легка робота. Робоче місце програміста є постійним, робота проводиться у холодний та теплий періоди року тому у приміщенні ОЦ підтримуються оптимальні показники мікроклімату [49].

Оптимальний рівень параметрів повітряного виробничого середовища підтримується у відповідності до ДСН 3.3.6-042-99 «Санітарні норми мікроклімату виробничих приміщень» [51].

Встановлені наступні оптимальні значення параметрів мікроклімату:

– у холодний період року: температура 22-24°С; відносна вологість: 40 – 60%; швидкість переміщення повітря: 0,1 м/с;

– у теплий період року: температура 23-25°С; відносна вологість: 40-60%; швидкість переміщення повітря: 0,1 – 0,2 м/с.

Оптимальні рівні позитивних (n+) і негативних (n-) іонів у повітрі приміщення з ПК нормовані згідно ГН 2152-80 «Санітарно-гігієнічні норми допустимих рівнів іонізації повітря виробничих та громадських приміщень» і становлять: n+=1500-30000 (шт. на 1см^3); n- = 3000-5000 (шт. на 1см^3). Підтримка оптимального рівня легких позитивних і негативних аероіонів у повітрі на робочих місцях забезпечено за допомогою біполярних коронних аероіонізаторів. В обчислювальному центрі передбачено: устрій системи водяного опалення приміщення – для забезпечення необхідної температури повітря в холодний період року [46].

Дотримання вимог цих документів досягається оснащенням приміщень пристроями кондиціювання і вентиляції, дезодорації повітря, опалювання згідно вимог ДБН В.2.5-67:2013 «Опалення, вентиляція та кондиціювання» [52].

Зимою значення відносної вологості повітря можуть бути нижчими за встановлені норми і становити в середньому 30–40%, що призводить до надмірного висихання слизових оболонок очей, носа, горла та до нагромадження зарядів статичної електрики, що утворюються в процесі роботи комп'ютера. В приміщені передбачена наявність 4 зволожувачів повітря, які допомагають підняти вологість до 60% для усунення таких проблем. Кількість зволожувачів зумовлена тим, що один такий апарат розраховано на приміщення площею до 15 м^2.

Літом для підтримки нормальної температури повітря в приміщенні використовуються три кондиціонери типу «спліт-система». Два кондиціонера розташовані у протилежних частинах та один по центру обчислювального центру дозволяють підтримувати температуру повітря не вище 24°С. Кількість кондиціонерів зумовлена тим, що один такий апарат розраховано на кондиціювання приміщення площею до 18 м^2.[49]

Під впливом шуму знижується концентрація уваги, порушуються фізіологічні функції, з'являється стомленість у зв'язку з підвищеними енергетичними витратами і нервово-психічною напругою, погіршується мовна комутація. Рівні звуку, рівні звукового тиску в октавних смугах частот, та

еквівалентні рівні звуку на робочих місцях у приміщенні ОЦ нормуються згідно ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [52] та ДСН 3.3.6-039-99 [51]. Оптимальній рів шуму в приміщення забезпечується за допомогою:

– використання блоків живлення ПК з вентиляторами на гумових підвісках;

– використання більш сучасного обладнання;

– розташування принтерів та різноманітного устаткування колективного користування на значній відстані від більшості робочих місць працівників;

– переведення жорсткого диска в режим сну (Standby), якщо комп'ютер не працює протягом визначеного часу.

Інтенсивне електромагнітне випромінювання постійно впливає на програміста в період роботи за дисплеєм, яке може стати причиною професійних захворювань. Комп’ютери потрапляють за визначенням в 5-й діапазон (низькі частоти). Для зниження потужність експозиційної дози рентгенівського випромінювання для роботи використовуються монітори з рідкокристалічними дисплеями [46].

Освітлення у приміщеннях з ЕОМ суміщене, при якому недостатнє за нормами природне освітлення доповнене штучним. Природне освітлення бічне, одностороннє. Вимоги до освітлення регламентовані ДБН В.2.5-28-2018 «Природне та штучне освітлення» [53].

Згідно з вимогами регламентованими в ДБН В.2.5-28-2018 в обчислювальному центрі коефіцієнт природної освітленості (КПО) має становити в середньому 1,1% (1,2% – у сонячний день та 1% – у хмарну погоду). Вікна в приміщені обладнанні сонцезахисними металевими жалюзі з системою регульована для забезпечення відносної постійності природного освітлення. коефіцієнт відбиття жалюзі становить 0,7. Робочі місця робітників розташовано так, щоб у поле зору робітника не потрапляли світлі поверхні світильників та вікон.

Проведемо розрахунок штучного освітлення в приміщенні ОЦ розміри якого складають: довжина 26 м, ширина 14 м, висота 5,4 м. Висота робочої поверхні h_p, = 0,7 м, висота звисання світильника від стелі $h_з$ = 1,7 м. Рівень нормованого значення освітлення E_H = 300 лк, колір стелі, стін, підлоги $\rho_{ст}$= 70%,, $\rho_{с,}$ = 50%, $\rho_п$ = 30%.

Розрахуємо кількість рядів світильників у приміщенні за формулою (6.1):

$$N_p = \frac{B}{(H-h_p)\cdot[L\,/\,h]} = \frac{14}{(5{,}4-0{,}7)\cdot 1{,}3} \approx 3. \quad (6.1)$$

Визначимо максимально припустиму відстань між рядами світильників за формулою (6.2):

$$L_{max} = \frac{B}{N_p}, \quad (6.2)$$

де В – ширина приміщення, м.;

$N_р$ – кількість рядів.

Розрахуємо максимально припустиму відстань між рядами світильників:

$$L_{max} = \frac{14}{3} = 4{,}7 \text{ м.}$$

Визначимо висоту підвісу світильника h над робочою поверхнею за формулою (6.3):

$$h = H - h_р - h_з, \quad (6.3)$$

де $h_р$ – висота робочої поверхні, м.;

$h_з$ – висота звисання світильника від стелі, м.;

H – висота приміщення, м.

Розраховуємо висоту підвісу світильника:

$$h = 5{,}4 - 0{,}7 - 1{,}7 = 3 \text{ м.}$$

Визначимо значення індексу приміщення і, що характеризує співвідношення розмірів освітлювального приміщення і висоти розміщення світильників за формулою (6.4):

$$\text{i} = \frac{A \cdot B}{h \cdot (A+B)}, \tag{6.4}$$

де A – довжина приміщення, м.;

B – ширина приміщення, м.;

h – висота підвісу світильника над робочою поверхнею, м.

Розрахуємо співвідношення розмірів освітлювального приміщення і висоти розміщення світильників:

$$\text{i} = \frac{26 \cdot 14}{3 \cdot (26+14)} = 3.$$

Визначимо значення коефіцієнта використання світлового потоку η, створюваного світильниками вибраного типу. Вибирається в залежності від виду джерела світла, типу обраного світильника, коефіцієнтів відбиття поверхонь приміщення та індексу приміщення.

$$\eta = 59\ \%$$

Визначимо сумарний світловий потік освітлювальної установки у даному приміщені Φ_Σ за формулою (6.5):

$$\Phi_\Sigma = \frac{E_\text{н} \cdot A \cdot B \cdot k_3 \cdot z}{\eta}, \tag{6.5}$$

де: $E_\text{н}$ – рівень нормованого загального освітлення, лк;

A – довжина приміщення, м;

B – ширина приміщення, м;

$k_з$ – коефіцієнт запасу (для люмінесцентних ламп $k_з = 1{,}4 \ldots 1{,}7$);

z – коефіцієнт нерівномірності (мінімальної) освітленості (відношення середньої освітленості до мінімальної освітленості), як правило дорівнює (для люмінесцентних ламп z =1,1);

η – коефіцієнт використання світлового потоку.

Розрахуємо сумарний світловий потік освітлювальної установки у даному приміщені:

$$\Phi_z = \frac{300*26*14*1,5*1,1}{0,59} = 305390 \text{ лм.}$$

Визначаємо умовну загальну кількість світильників у приміщені за формулою (6.6):

$$N^*_{\text{св}} = AB/L^2_{max} \qquad (6.6)$$

Розрахуємо умовну загальну кількість світильників у приміщені:

$$N^*_{\text{св}} = \frac{26 \cdot 14}{22,1} = 16 \text{ шт.}$$

Визначаємо світловий потік умовного джерела світла за формулою (6.7):

$$\Phi^*_{\text{св}} = \frac{\Phi_\Sigma}{N^*_{\text{св}}} \text{ лм,} \qquad (6.7)$$

Розрахуємо світловий потік умовного джерела світла:

$$\Phi^*_{\text{св}} = \frac{305390}{16} = 19087 \text{ лм.}$$

Обираємо *LED*-світильник типу INOX LED 80 з найближчим значенням фактичного світлового потоку $\Phi_{св}$ = 10000 лм і знаходимо коефіцієнт m (співвідношення між розрахунковим світловим потоком світильника $\Phi^*_{св}$ та фактичним світловим потоком вибраного *LED*-світильника $\Phi_{св}$):

$$m = \frac{\Phi^{*}_{\text{св}}}{\Phi_{\text{св}}},$$

$$\text{m} = \frac{19087}{10000} = 1{,}9.$$

Визначаємо оптимальну (фактичну) кількість світильників у приміщенні:

$$\text{N} = \text{N}^{*}_{св}\ \text{m}$$

$$\text{N} = 16 * 1{,}9 \approx 30 \text{ шт.}$$

Таким чином для освітлення приміщення ОЦ використовуємо 30 світильників INOX LED 80, розміщенні по 10 у кожному ряді.

Для захисту від прямих сонячних променів, які створюють прямі та відбиті відблиски на поверхні екранів і клавіатури, використовуються сонцезахисні пристрої, на вікнах встановлені жалюзі або штори.

Організація та обладнання робочого місця в робочу приміщені має відповідати ергономічним вимогам з урахуванням особливостей трудовою дальності.

В приміщені обчислювального центру одночасно працюють 6 чоловік, кожній робітник має свій:

– обчислювальний пристрій, пристрою вводу (клавіатура, миш та інше), виводу(дисплей монітору, принтер, звукова колонка та інше) та набор периферійних пристроїв (принтер, сканер та інше);

– стіл;

– два ортопедичні стільці;

– тумбу.

Також кожний робітник може скористатися шафою для одягу, шафою з приладами, двома тумбами призначеними для використанню всіма робітниками.

Робоче місце працівника має площу 6 м^2, а об’єм – 20 м^3.

Приміщення має природне та штучне освітлення. Робочі місця розташовано так, щоб природне світло падало збоку. Робочі місця та взаємне розташування всіх його елементів відповідати антропометричним, фізичним і психологічним вимогам.

Також всі робочі місця організовані за такими правилами організації робочого місця:

– всім робітникам надані ергономічні та зручних миші і клавіатури;

– висота столу забезпечує розташування клавіатури від підлоги – 65-75 см, що позитивно впливає на працездатність робітника;

– кожна клавіатура має можливість міняти висоти і нахилу клавіатури;

– кожна клавіатура має підставку для рук;

– всі килимки для миші мають захист від тунельного синдрому (спеціальний виступ забезпечує правильне положення кисті);

– екрани монітору знаходитися на оптимальній відстані 700 мм від працівника;

– кожний стілець має підлокітник [50].

6.4 Заходи безпеки у надзвичайних ситуаціях

6.4.1 Заходи з пожежної безпеки

Великою небезпекою для обчислювальному центру є пожежа. Обчислювальний центр наповнений різноманітними електричними приладами, а пожежа основний фактор для їх знищення, що може призвести до великих матеріальних втрат. Так як в приміщенні обчислювального центру присутні всі три основні чинника, необхідні для виникнення пожежі: горючі речовини, окислення і джерела запалювання, то налагодження заходів пожежної безпека в такому приміщення є першочергово важливим.

Ймовірними причинами виникнення пожару можуть бути несправність електрообладнання (кабелів, розеток), короткі замикання внаслідок виходу з

ладу чи експлуатації несправного електроустаткування (ПЕОМ, периферійних пристроїв), порушення правил протипожежної безпеки тощо.

Для забезпечення пожежної безпеки проводився пожежна профілактика, яка включає в себе комплекс організаційних і технічних заходів, спрямованих на забезпечення безпеки людей, на запобігання пожежі, обмеження її поширення, а також на створення умов для успішного гасіння пожежі.

Експлуатація ліній електромережі практично повністю унеможливлює виникнення електричного джерела загоряння в наслідок короткого замикання та перевантаження проводів, так як застосовуються дроти з важкогорючою і негорючою ізоляцією.

Для відводу теплоти від ПК у обчислювальному центрі постійно діє потужна система кондиціювання.

Обчислювальний центр відноситься до категорії «П-IIа» згідно ДСТУ Б В.1.1-36:2016 «Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою» [54], згідно ДБН В.1.1-7:2016 «Пожежна безпека об’єктів будівництва. Загальні вимоги» [55], а клас можливої пожежі визначається, як «А».

Для ліквідації пожежі у початковій стадії її розвитку силами персоналу об’єктів застосовуються первинні засоби пожежогасіння.

Так як приміщення ОЦ має площу 364 м2, тому відповідно до вимог п. 3.8 розділу «Типові норми належності вогнегасників» ДСТУ 4297:2004 «Пожежна техніка. Технічне обслуговування вогнегасників. Загальні технічні вимоги» [46] для гасіння електроустановок, що знаходяться під напругою, передбачені вуглекислотні вогнегасники типу ВВК-3,5 у кількості 18 штук (з розрахунку один вогнегасник з величиною заряду вогнегасної речовини 3 кг. і більше, на 20 м2 площі приміщення). Відстань між вогнегасниками та місцями можливих загорянь не перевищує 15 м.

Під час прийняття на роботу і в процесі праці усі працівники обчислювального центру повинні проходити протипожежний інструктаж,

перевірку знань з питань пожежної безпеки. Евакуаційні виходи і проходи утримуються постійно вільними та легкодоступними.

Відповідальний за протипожежний стан приміщень обчислювального центру повинен періодично (не рідше одного разу на місяць) перевіряти працездатність пристрою, який забезпечує автоматичне вимкнення системи вентиляції у разі пожежі, а також вогнедимозатримувальних пристроїв. Під час ремонту слід не допускати руйнування або порушення цілісності негорючих діафрагм у фальш-підлоговому просторі [56].

6.4.2 Заходи з цивільного захисту

Організація навчання працюючого та непрацюючого населення діям у надзвичайних ситуаціях.

Надзвичайна ситуація – порушення нормальних умов життя і діяльності людей на об’єкті або території, спричинене аварією, катастрофою, стихійним лихом, епідемією, епізоотією, епіфітотією, великою пожежею, застосуванням засобів ураження, що призвели або можуть призвести до людських і матеріальних втрат.

Порядок здійснення навчання населення діям у надзвичайних ситуаціях затверджено відповідною постановою Кабінету Міністрів України від 26 червня 2013 р. № 444. Цей Порядок визначає механізм організації навчання населення діям у надзвичайних ситуаціях (далі - навчання населення), його структуру, види та форми. Навчання населення здійснюється:

- за місцем роботи - працюючого населення;
- за місцем навчання - дітей дошкільного віку, учнів та студентів;
- за місцем проживання - непрацюючого населення.

Організація навчання населення покладається на працюючого та непрацюючого - на ДСНС, місцеві державні адміністрації, органи місцевого самоврядування, а дітей дошкільного віку, учнів та студентів - на МОН.

Навчально-методичне забезпечення навчання населення здійснюється ДСНС разом з МОН. Навчання населення складається з:

– навчання безпосередньо на підприємствах, в установах та організаціях;

– навчання за межами підприємств, установ та організацій керівного складу і фахівців з питань цивільного захисту та пожежної безпеки;

– практичної підготовки під час проведення спеціальних об’єктових навчань і тренувань з питань цивільного захисту;

– навчання під час здобуття відповідного освітнього рівня у навчальних закладах системи освіти;

– самостійного вивчення інформації про дії в умовах надзвичайних ситуацій.

Навчання працюючого населення здійснюється безпосередньо на підприємстві, в установі та організації згідно з програмами підготовки працівників до дій у надзвичайних ситуаціях, а також під час проведення спеціальних об’єктових навчань і тренувань з питань цивільного захисту. Програми підготовки працівників до дій у надзвичайних ситуаціях розробляються і затверджуються підприємствами, установами, організаціями на підставі програм та організаційно-методичних вказівок з підготовки населення до дій у надзвичайних ситуаціях, що розробляються і затверджуються ДСНС, місцевими державними адміністраціями, органами місцевого самоврядування. Програми навчання з питань пожежної безпеки погоджуються із ДСНС.

Програми підготовки працівників до дій у надзвичайних ситуаціях поділяються на:

– загальної підготовки працівників підприємств, установ та організацій;

– спеціальної підготовки працівників, що входять до складу спеціалізованих служб і формувань цивільного захисту;

– додаткової підготовки з техногенної безпеки працівників об’єктів підвищеної небезпеки;

– пожежно-технічного мінімуму для працівників, зайнятих на роботах з підвищеною пожежною небезпекою;

– прискореної підготовки працівників до дій в особливий період.

Підготовка працівників до дій у надзвичайних ситуаціях передбачає:

– за програмою загальної підготовки працівників підприємств, установ та організацій - вивчення інформації, що міститься у планах реагування на надзвичайні ситуації, про дії в умовах загрози і виникнення надзвичайної ситуації, а також оволодіння навичками надання першої допомоги потерпілим, користування засобами індивідуального і колективного захисту;

– за програмою спеціальної підготовки працівників, що входять до складу спеціалізованих служб і формувань цивільного захисту, - ознайомлення з обов'язками, навичками користування та матеріальною частиною техніки, приладів і табельного майна таких служб і формувань, засобами захисту, вивчення порядку приведення їх у готовність, проведення рятувальних та інших невідкладних робіт;

– за програмою додаткової підготовки з техногенної безпеки працівників об'єктів підвищеної небезпеки - поглиблення знань з питань техногенної безпеки, джерел небезпеки, що за певних обставин можуть спричинити виникнення надзвичайної ситуації на об'єкті підвищеної небезпеки, та небезпечних речовин, що виготовляються, переробляються, зберігаються чи транспортуються на його території;

– за програмою пожежно-технічного мінімуму для працівників, зайнятих на роботах з підвищеною пожежною небезпекою, - підвищення рівня загальних пожежно-технічних знань, вивчення правил пожежної безпеки з урахуванням особливостей виробництва, ознайомлення з протипожежними заходами та діями у разі виникнення пожежі, оволодіння навичками використання наявних засобів пожежогасіння;

– за програмою прискореної підготовки працівників до дій в особливий період - навчання способам захисту від наслідків надзвичайних ситуацій,

спричинених застосуванням засобів ураження в особливий період, що здійснюється підприємствами, установами та організаціями, які продовжують роботу у воєнний час, і розпочинається одночасно з уведенням в дію планів цивільного захисту на особливий період.

Навчання працівників на підприємстві, в установі та організації здійснюється шляхом:

– курсового навчання, що передбачає формування навчальних груп і здійснюється в навчальних класах або на об’єктах навчально-виробничої бази підприємства, установи та організації;

– індивідуального навчання, що передбачає вивчення теоретичного матеріалу самостійно та у формі консультацій з керівниками навчальних груп або іншими особами.

Навчальні групи комплектуються переважно з працівників, що входять до складу спеціалізованих служб і формувань цивільного захисту.

На підприємствах, в установах та організаціях із чисельністю працівників 50 і менше осіб навчання може здійснюватися шляхом проведення інструктажів за програмою загальної підготовки працівників, які проводяться особами з питань цивільного захисту, призначеними в межах штатної чисельності суб’єкта господарювання [57].

ВИСНОВКИ

В процесі виконання дипломної кваліфікаційної роботи магістра було виконано такі завдання:

– розглянуто класифікацію задач прийняття рішень;

– досліджено існуючі методи групового експертного оцінювання та обрано необхідний для забезпечення ГЕО якості програмних інтерфейсів;

– проведено аналіз моделей ЯПЗ та обрано метрики, необхідні для кількісної оцінки якості програмних інтерфейсів;

– розроблено моделі програмного забезпечення для ГЕО якості програмних інтерфейсів;

– розроблено програмне та інформаційне забезпечення компонентів системи, що автоматизує роботу проєктувальника;

– проведено валідацію розробленого програмного забезпечення на прикладі ГЕО вибору шаблонів компоновки екранних форм програмних інтерфейсів за критеріями якості.

Наукова новизна результатів дослідження полягає у тому, що розроблено метод оцінки якості проєктування програмних інтерфейсів, який грунтується на відомому методі аналіза ієрархій Т. Сааті та комплекс нових моделей програмного забезпечення, що автоматизує цей метод у вигляді use case моделі та діаграми класів.

Практична цінність роботи полягає в тому, що на підставі досліджень розроблено програмне та інформаційне забезпечення, що забезпечує автоматизацію роботи проєктувальника інтерфейсів при виборі альтернативних варіантів проєктних рішень.

Розроблена система дозволяє підвищити якість проєктування інтерфейсів користувача та зменшити витрати на етапі тестування програмного продукту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ветров, Ю. А. Проблемы промышленного проектирования интерфейсов / Ю. А. Ветров // Открытые семантические технологии проектирования интеллектуальных систем OSTIS-2013: III Международная научно-техническая конференция, Минск, 21–23 февраля 2013 г.: тезисы докладов. – Минск: БГУИР. – С. 407 – 412.

2. Анализ подходов к созданию пользовательского интерфейса: [Электрон. ресурс]. – Режим доступа: https://cyberleninka.ru/article/n/analiz-podhodov-k-sozdaniyu-polzovatelskogo-interfeysa.

3. Сердюк, С. М. Ергономічні питання проектування людино-машинних систем: навчальний посібник / С. М. Сердюк. – Запоріжжя: ЗНТУ, 2014. – 334 с.

4. Витковская, К. Шаблоны в проектировании интерфейсов [Электрон. ресурс]. – Режим доступа: https://telegraf.design/shablony-v-proektyrovanyy-ynterfejsov/.

5. Нейл, Т. Мобильная разработка. Галерея шаблонов / Т. Нейл. – СПб.: Питер, 2013. – 208 с.

6. Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems»: ISO 9241-210:2010. – [Effective from 2010-12-15]. – Geneve: ISO, 2010. – 36 p.

7. Саати, Т. Л. Принятие решений. Метод анализа иерархий [Текст] / Т. Л. Саати. – М.: Радио и связь, 1993. – 320 с.

8. Triantaphyllou, E. Two new cases o rank reversals when AHP and some of its additive variants are used that do not occur with multiplicative AHP [Текст] / E.Triantaphyllou //Journal of multicriteria decision analysis. – 2001. –№ 10. – P. 11-25.

9. Системная стратегия технологического предвидения в инновационной деятельности / М. З. Згуровский, Н. Д. Панкратова // Систем. дослідж. та інформ. технології. – 2003. – № 3. – С. 7-24.

10. Brans, J. P. Promethee Methods [Текст] / J. P. Brans, B. Mareschal // Multiple Criteria Decision Analysis: State of the Art Surveys. – 2005. – vol 78. – P. 163-195.

11. Панкратова, Н. Д. Методология обработки нечеткой экспертной информации в задачах предвидения. Ч. 1 [Текст] / Н. Д. Панкратова, Н. И. Недашковская // Проблемы управления и информатики. – 2007. – № 2. – С. 40-55.

12. Каминская, Ж. К. Метод оценки деятельности оператора автоматизированной системы управления технологическим процессом газобетонного производства / Ж. К. Каминская, С. Н. Сердюк, Кулинич Э. М. // Радіоелектроніка, інформатика, управління. 2019. No2. С.177 – 188 DOI 10.15588/1607-3274-2019-2-19 .

13. Борисов, А. Н. Принятие решений на основе нечетких моделей / А. Н. Борисов, О. А. Крумберг, И. П. Федоров. – Рига: Зинатне, 1990. – 184 с.

14. Андрейчиков, А. В. Анализ, синтез, планирование решений в экономике / А. В. Андрейчиков, О. Н. Андрейчикова. – М.: Финансы и статистика, 2000. – 205 с.

15. Подиновский, В. В. Парето-оптимальные решения многокритериальных задач / В. В. Подиновский, В. Д. Ногин. – М.: Наука, 1982. – 256 с.

16. Беляев, Л. С. Решение сложных оптимизационных задач в условиях неопределенности. – Новосибирск: Наука, 1978. – 126 с.

17. Фишберн, П. С. Теория полезности для принятия решений. – М.: Наука, 1977. – 352 с.

18. Ларичев, О. И. Анализ процессов принятия человеком решений при альтернативах, имеющих оценки по многим критериям / О. И. Ларичев // Автоматика и телемеханика, – 1981. – №8. – С. 131 – 141.

19. Евланов, Л. Г. Теория и практика принятия решений. – М.: Экономика, 1984. – 176 с.

20. Интерактивный метод решения задачи оптимального

проектирования машин / [И. И. Артоболевский, С. В. Емельянов, В. И. Сергеев и др.] // Докл. АН СССР, 1977. Т. 237. – № 4. – С. 793 – 795.

21. Борисов, А. Н. Методы интерактивной оценки решений / А. Н. Борисов, А. С. Левченко. – Рига: Зинатне, 1982. – 139 с.

22. Борисов, А. Н. Диалоговые системы принятия решений на базе мини-ЭВМ / А. Н. Борисов, Э. Р. Виллюмс, Л. Я. Сукур. – Рига: Зинатне, 1986. – 195 с.

23. Саати, Т. Принятие решений. Метод анализа иеархий: Пер.с англ. – М.: Радио и связь, 1989. – 316 с.

24. Brans, J.P. Promethee Methods // Multiple Criteria Decision Analysis: State of the Art Surveys [Text] / J.P.Brans, B.Mareschal. – 2005. – vol 78. – P. 163-195.

25. Hwang, C. L. Multiple Attribute Decision Making: Methods and Application [Text] / C. L. Hwang, K. Yoon. – Springer, New York, 1981. – 259 p.

26. Milani, A. S. Using different ELECTRE methods in strategic planning in the presence of human behavioral resistance / A. S. Milani, A. Shanian, C. El-Lahham // Journal of Applied Mathematics and Decision Sciences. – 2006. – № 1. – P. 1-19.

27. Leyva-Lopez, J. C. A new method for group decision support based on ELECTRE III methodology / J. C. Leyva-Lopez, E. Fernandez-Gonzalez // European Journal of Operational Research. – 2003. – № 148. – P. 14-27.

28. Olson, D. L. Comparison of weights in TOPSIS models / D. L. Olson // Mathematical and Computer Modelling. – 2004. –vol 40.– №7-8. – P. 721-727.

29. Кравченко, Т. К. Развитие систем поддержки принятия решений с помощью метода Promethee / Т. К. Кравченко, Ю. В. Авдеев // Актуальные проблемы гуманитарных и естественных наук. – 2010. – № 9. – С. 73–76.

30. Романченко, І. С. Метод TOPSIS-ядро та його використання для багатокритеріального порівняння альтернатив / І. С. Романченко, М. М. Потьомкін // Обробка інформації в складних організаційних системах. – 2016. – № 1. – С. 103–106.

31. Миронова, Н. О. Методи та інформаційна технологія багатокритеріального колективного експертного оцінювання з використанням

нечітких оцінок: дис. … канд. техн. наук : 05.13.06 / Миронова Наталя Олексіївна. – Запоріжжя, 2016. – 159 с.

32. Software Engineering Product Quality - Part 1: Quality Model: ISO/IEC, ISO/IEC 9126-1:2001. – [Effective from 2001-12-15]. – Geneve: ISO, 2001. – 26 p.

33. Панкратова, Н. Д. Методология обработки нечеткой экспертной информации в задачах предвидения. Ч. 1 [Текст] / Н. Д. Панкратова, Н. И. Недашковская // Проблемы управления и информатики. – 2007. – № 2. – С. 40-55.

34. Ногин, В. Д. Упрощенный вариант МАИ на основе нелинейной свертки критериев / В. Д. Ногин // Журнал вычислительной математики и математической физики. – 2004. – Т. 44. – № 7. – С. 1259-1268.

35. Saaty, T. L. Decision making with Dependence and Feedback. The Analityc Network Process / T. L. Saaty; PWS Publications. – Pittsburgh: 2001. – 386 p.

36. Трофимец, В. Я. К вопросу разработки основных вычислительных процедур МАИ: [Электрон. ресурс]. – Режим доступа: http://zhurnal.ape.relarn.ru/articles/2004/078.pdf.

37. Сайко, В. В. Модификация метода парных сравнений для случаев с большим количеством оцениваемых параметров / В. В. Сайко // Интеллектуальные системы принятия решений и проблемы вычислительного интеллекта: Материалы международной научной конференции. – Том 1. – Херсон: ХНТУ, 2009. – С. 210-214.

38. Schneiderman, S. Designing the User Interface. Strategies for Effective Human-Computer Interaction / S. Schneiderman. – Boston: Addisson-Wessley, third edition, 1998. – 639 p.

39. Bevan, Nigel. Classifying and selecting UX and usability measures / Nigel Bevan // Proceedings of Meaningful Measures: Valid Useful User Experience Measurement (VUUM), 5th COST294-MAUSE Open Workshop, 18th June 2008: proceedings. – Reykjavik: IEEE, 2008. – P. 241–245.

40. Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability:ISO 9241-11:1998. – [Effective from 1998-12-15]. – Geneve: ISO, 1998. – 29 p.

41. Kazuhiro, E. System Quality Requirement and Evaluation. Importance of application of the ISO/IEC25000 series. / E. Kazuhiro // Global Perspectives on Engineering Management. – 2013. - Vol. 2 Iss. 2, - P. 52-59.

42. Bias, G., Mayhew, D. Cost-Justifying Usability / G. Bias, D. Mayhew: Academic Press, 1994. – 255 p.

43. Бродецкий, Г. Л. Методы оптимизации многокритериальных решений в логистике / Г. Л. Бродецкий. – М.: Academia, 2010. – 314 с.

44. Методичні вказівки до лабораторних робіт з дисципліни “Людино-машинна взаємодія” для студентів спеціальності 7.05010301“Програмне забезпечення систем” та 7.05010302 “Інженерія програмного забезпечення” денної форми навчання. Частина 1 / Укл.: С. М. Сердюк, Ж. К. Камінська, О.О. Степаненко. – Запоріжжя: ЗНТУ, 2014. – 58с.

45. Остапенко, В. В. Методичні вказівки по економічному обґрунтуванню дипломних проєктів для студентів спеціальностей 7.080403 «Програмне забезпечення автоматизованих систем», 7.080402 «Інформаційні технології проєктування» і 7.091501 «Комп’ютерні та інтелектуальні системи і мережі» всіх форм навчання [Текст] / В. В. Остапенко, Е. М. Касьян, Є. М. Сердюк. – Запоріжжя : ЗНТУ, 2009. – 22 с.

46. Атаманчук П. Охорона праці в галузі. Навчальний посібник / П. Атаманчук, В. Мендерецкий, О. Панчук, Р. Билык. К: Центр навчальної літератури, 2017. – 322 с.

47. Правила улаштування електроустановок [Текст]. Міненерговугілля України, 2017. – 617 с.

48. Березуцький В. В. Небезпечні виробничі ризики та надійність: навч. посібник / В. В. Березуцький, М. І. Адаменко; Харківський політехнічний ін-т, нац. техн. ун-т. – Харків: ФОП Панов А. М., 2016. – 385 с.

49. Березуцький В. В. Небезпечні виробничі ризики та надійність: навч.

посібник / В. В. Березуцький, М. І. Адаменко; Харківський політехнічний ін-т, нац. техн. ун-т. – Харків: ФОП Панов А. М., 2016. – 385 с.

50. Охорона праці: навч. посіб. / З. М. Яремко, С. В. Тимошук, О. І. Третяк, Р. М. Ковтун; за ред. проф. З. М. Яремка. - Львів: Видавничий центр ЛНУ імені Івана Франка, 2010. - 374 с.

51. Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями. НПАОП 0.00-7.15-18. Державні санітарні правила і норми. / Харків: Форт, 2018 – 48 с.

52. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень. / Харків: Форт, 2000 – 24 с.

53. ДБН В.2.5-67:2013 «Опалення, вентиляція та кондиціонування». / Київ: Мінрегіон України, 2013. – 139 с.

54. ДБН В.2.5-28:2018. Природне та штучне освітлення. Посібник: проблеми природного і штучного освітлення. / К: Етін, 2019. – 180 с.

55. Правила пожежної безпеки в Україні. 2015 / Харків : Форт, 2015 – 124 с.

56. ДБН В.1.1-7:2016 «Пожежна безпека об’єктів будівництва. Загальні вимоги». / Київ: Міністерство регіонального розвитку, будівництва та житлово-комунального господарства України, 2017. – 39 с.

57. Стеблюк М. І. Цивільна оборона та цивільний захист [Текст] : навч. посіб. для вузів / М. І. Стеблюк. – К. : Знання, 2013, - 487 с.

ДОДАТОК А

Текст програми

А.1 Текст розроблених застосунків

А.1.1 Текст файлу Figures.cs

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace AHP2._0.Class
{
    public class Figures
    {
        private float width;
        private float height;
        private float wall;
        public Figures(float x, float y, float width, float height,
float wall)
        {
            this.width = width;
            this.height = height;
            this.wall = wall;
        }

        public float Width { get => width; set => width = value; }
        public float Height { get => height; set => height = value;
}
        public float Wall { get => wall; set => wall = value; }
    }
}
```

А.1.2 Текст файлу InfoCriteria.cs

```
using System;
namespace AHP2._0.Class
{
    public class InfoCriteria
    {
        static private int countCriteria = 0;
```

```
                static private int countAlternative = 0;
                private int id;                          // Ідентифікатор таблиці.
                private string type = String.Empty;      // Тип шкали.
                private string title = null;             // Назва критерія.
                private string[,] infoTable;                  // Значення таблиці.

        //private RatingTable infoTable;
                private ResultRatingTable infoTableResult;
                public InfoCriteria() { }
                public InfoCriteria(InfoCriteria o)
                {
                    Id = o.Id;
                    InfoTable = o.InfoTable;
                    infoTableResult = o.infoTableResult;
                    Title = o.Title;
                }
                public InfoCriteria(int id, ResultRatingTable infoTableResult,
string title, string type, string[,] arr)
                {
                    Id = id;
                    InfoTableResult = infoTableResult;
                    Title = title;
                    Type = type;
                    InfoTable = arr;
                }
                public int Id { get => id; set => id = value; }
                public    ResultRatingTable    InfoTableResult    {    get    =>
infoTableResult; set => infoTableResult = value; }
                public string Title { get => title; set => title = value; }
                public static int CountCriteria { get => countCriteria; set =>
countCriteria = value; }
                public static int CountAlternative { get => countAlternative; set
=> countAlternative = value; }
                public string Type { get => type; set => type = value; }
                public string[,] InfoTable { get => infoTable; set => infoTable
= value; }
            }
        }
```

A.1.3 Текст файлу IValidation.cs

```
using System;
using System.Collections.Generic;
namespace AHP2._0.Class
{
    public class IValidation
    {
        private static string messageError = "";
        public static string MessageError { get => messageError; set =>
messageError = value; }
        public static bool CheckComboBoxValue(Object o)
        {
            if (o != null)
                return true;
            else
                return false;
        }
        public static bool CheckDataGridTextBox(String s)
        {
            bool check = false;
            if (s.Length >= 3 && s != "")
                check = true;

            return check;
        }
        public static bool CheckType(int num, string type)
        {
            bool check = true;
            if (type == "Sati")
                if (num > 9)
                    check = false;
            return check;
        }
        public static bool CheckIsLetter(String s)
        {
            bool check = true;
            foreach (char c in s)
                if (Char.IsLetter(c) || c == '.' || c == '\\' || c == '
')
                    check = false;
```

```
            return check;
        }

        public static List<string> ValidationArray(string[,] array, int
size)
        {
            List<string> wrongType = new List<string>();
            for (int i = 0; i < size; i++)
            {
                for (int j = i; j < size; j++)
                {
                    string tempS = array[i, j];
                    if (tempS != "" && tempS != null)
                    {
                        if (CheckIsLetter(tempS))
                        {
                            int number = 0;
                            if (Int32.TryParse(tempS, out number))
                            {
                                if (!CheckType(number, "Sati"))
                                    wrongType.Add(number.ToString());
                            }
                            else
                            {
                                int first =
Convert.ToInt32(tempS.Split('/')[1]);
                                int second =
Convert.ToInt32(tempS.Split('/')[0]);
                                if (first > 9 || second > 9)
                                    wrongType.Add(tempS);
                            }
                        }
                    }
                    tempS ="";
                }
            }
            return wrongType;
        }
    }
```

```
}
```

A.1.4 Текст файлу ResultRatingTable.cs

```
using System;
using System.Data;
using System.Threading.Tasks;
namespace AHP2._0.Class
{
    public class ResultRatingTable
    {
        // МПП - матриця парних порівнянь.
        private List<double> localResult;   // Список локальних
пріоритетів.
        private double dim;                 // Розмір МПП.
        private double lam;                 // Максимальне власне
значення МПП.
        private double ci;                  // Індекс узгодженності МПП;
        private double cr;                  // Відношення узгодженності
МПП.
        public ResultRatingTable(List<double> localResult, double dim,
double lam, double ci, double cr)
        {
            this.localResult = localResult;
            this.dim = dim;
            this.lam = lam;
            this.ci = ci;
            this.cr = cr;
        }
        public List<double> LocalResult { get => localResult; set =>
localResult = value; }
        public double DIM { get => dim; set => dim = value; }
        public double LAM { get => lam; set => lam = value; }
        public double CI { get => ci; set => ci = value; }
        public double CR { get => cr; set => cr = value; }
        public void CalculatedLocalPriority(string[,] array, int size)
        {
            if (this.LocalResult.Count > 0)
```

```
                LocalResult.Clear();
            double srgeom = 0;
            string tempS = "";
            double tempMiddleGeometry = 1;
            List<double> listMiddleGeometry = new List<double>();
            DataTable a = new DataTable();
            for (int i = 0; i < size; i++)
            {
                for (int j = 0; j < size; j++)
                {
                    srgeom = 0;
                    tempS = array[i, j];
                    if (tempS.Length > 1)
                        tempMiddleGeometry *=
Convert.ToDouble(a.Compute(tempS, ""));
                    else
                        tempMiddleGeometry *= Convert.ToDouble(tempS);
                }
                srgeom = Math.Pow(tempMiddleGeometry, 1.0 / size);
                listMiddleGeometry.Add(srgeom);
0;
            for (int i = 0; i < listMiddleGeometry.Count; i++)
                tempSumMidleGeometryc += listMiddleGeometry[i];
            for (int i = 0; i < size; i++)
                this.LocalResult.Add((listMiddleGeometry[i] /
tempSumMidleGeometryc));
            dim = size;
            CalculatedLAM(array, size);
            CalculatedCI(size);
            CalculatedCR(size);
        }
        public void CalculatedLAM(string[,] array, int size)
        {
            double llam = 0;
            double sum = 0;
            DataTable a = new DataTable();
            string tempS = "";
            for (int i = 0; i < size; i++)
            {
```

```
                for (int j = 0; j < size; j++)
                {
                    tempS = array[j, i];
                    if (tempS.Length > 1)
                        sum += Convert.ToDouble(a.Compute(tempS, ""));
                    else
                        sum += Convert.ToDouble(tempS);
                }
                llam += sum * this.LocalResult[i];
                sum = 0;
            }
            lam = llam;
        }
        public void CalculatedCI(int size)
        {
            ci = (lam - size) / (size - 1);
        }
        private static Dictionary<int, double> dictionaryP = new
Dictionary<int, double>()
        {
            {2, 0}, {3, 0.58}, {4, 0.90}, {5, 1.12}, {6, 1.24} , {7,
1.32}
        };
        public void CalculatedCR(int size)
        {
            double P = dictionaryP[size];
            if (P!=0)
                cr = (ci / P);
            else
                cr = 0;
        }
        public void ClearListAfterChange()
        {
            LocalResult.Clear();
        }
    }
}
```

A.1.5 Текст файлу ResultDataDesigner.cs

```
namespace AHP2._0
{
    partial class ResultData
    {
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }
        #region Windows Form Designer generated code
        private void InitializeComponent()
        {
            System.Windows.Forms.DataGridViewCellStyle
dataGridViewCellStyle4 = new
System.Windows.Forms.DataGridViewCellStyle();
            this.btnBack = new System.Windows.Forms.Button();
            this.btnBuild = new System.Windows.Forms.Button();
            this.CColumn8 = new
System.Windows.Forms.DataGridViewTextBoxColumn();
            this.CColumn6 = new
System.Windows.Forms.DataGridViewTextBoxColumn();
            this.CColumn5 = new
System.Windows.Forms.DataGridViewTextBoxColumn();
            this.CColumn4 = new
System.Windows.Forms.DataGridViewTextBoxColumn();
            this.CColumn3 = new
System.Windows.Forms.DataGridViewTextBoxColumn();
            this.CColumn2 = new
System.Windows.Forms.DataGridViewTextBoxColumn();
            this.CColumn1 = new
System.Windows.Forms.DataGridViewTextBoxColumn();
            this.TitleColumn = new
System.Windows.Forms.DataGridViewTextBoxColumn();
```

```
            this.dataGridResult = new
System.Windows.Forms.DataGridView();
            this.CColumn7 = new
System.Windows.Forms.DataGridViewTextBoxColumn();
            this.label1 = new System.Windows.Forms.Label();

((System.ComponentModel.ISupportInitialize)(this.dataGridResult)).BeginI
nit();
            this.SuspendLayout();
            // btnBack
            //
            this.btnBack.Font = new System.Drawing.Font("Microsoft Sans
Serif", 9.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.btnBack.Location = new System.Drawing.Point(395, 340);
            this.btnBack.Name = "btnBack";
            this.btnBack.Size = new System.Drawing.Size(100, 25);
            this.btnBack.TabIndex = 32;
            this.btnBack.Text = "Вернуться";
            this.btnBack.UseVisualStyleBackColor = true;
            this.btnBack.Click += new
System.EventHandler(this.btnBack_Click);
            // btnBuild
            //
            this.btnBuild.Font = new System.Drawing.Font("Microsoft Sans
Serif", 9.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.btnBuild.Location = new System.Drawing.Point(510, 340);
            this.btnBuild.Name = "btnBuild";
            this.btnBuild.Size = new System.Drawing.Size(100, 25);
            this.btnBuild.TabIndex = 29;
            this.btnBuild.Text = "Построить";
            this.btnBuild.UseVisualStyleBackColor = true;
            this.btnBuild.Click += new
System.EventHandler(this.btnBuild_Click);
            // CColumn8
            //
            this.CColumn8.HeaderText = "";
            this.CColumn8.Name = "CColumn8";
```

```
            this.CColumn8.SortMode =
System.Windows.Forms.DataGridViewColumnSortMode.NotSortable;
            this.CColumn8.Width = 55;
            // CColumn6
            //
            this.CColumn6.HeaderText = "";
            this.CColumn6.Name = "CColumn6";
            this.CColumn6.SortMode =
System.Windows.Forms.DataGridViewColumnSortMode.NotSortable;
            this.CColumn6.Width = 55;
            // CColumn5
            //
            this.CColumn5.HeaderText = "";
            this.CColumn5.Name = "CColumn5";
            this.CColumn5.SortMode =
System.Windows.Forms.DataGridViewColumnSortMode.NotSortable;
            this.CColumn5.Width = 55;
            // CColumn4
            //
            this.CColumn4.HeaderText = "";
            this.CColumn4.Name = "CColumn4";
            this.CColumn4.SortMode =
System.Windows.Forms.DataGridViewColumnSortMode.NotSortable;
            this.CColumn4.Width = 55;
            // CColumn3
            //
            this.CColumn3.HeaderText = "";
            this.CColumn3.Name = "CColumn3";
            this.CColumn3.SortMode =
System.Windows.Forms.DataGridViewColumnSortMode.NotSortable;
            this.CColumn3.Width = 55;
            // CColumn2
            //
            this.CColumn2.HeaderText = "";
            this.CColumn2.Name = "CColumn2";
            this.CColumn2.SortMode =
System.Windows.Forms.DataGridViewColumnSortMode.NotSortable;
            this.CColumn2.Width = 55;
            // CColumn1
```

```
            //
            this.CColumn1.HeaderText = "";
            this.CColumn1.Name = "CColumn1";
            this.CColumn1.SortMode =
System.Windows.Forms.DataGridViewColumnSortMode.NotSortable;
            this.CColumn1.Width = 55;
            // TitleColumn
            //
            dataGridViewCellStyle4.Font = new
System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(204)));
            this.TitleColumn.DefaultCellStyle = dataGridViewCellStyle4;
            this.TitleColumn.HeaderText = "";
            this.TitleColumn.Name = "TitleColumn";
            this.TitleColumn.ReadOnly = true;
            this.TitleColumn.Resizable =
System.Windows.Forms.DataGridViewTriState.False;
            this.TitleColumn.SortMode =
System.Windows.Forms.DataGridViewColumnSortMode.NotSortable;
            this.TitleColumn.Width = 110;
            // dataGridResult
            //
            this.dataGridResult.AllowUserToAddRows = false;
            this.dataGridResult.AllowUserToDeleteRows = false;
            this.dataGridResult.AllowUserToResizeColumns = false;
            this.dataGridResult.AllowUserToResizeRows = false;
            this.dataGridResult.BackgroundColor =
System.Drawing.SystemColors.ButtonHighlight;
            this.dataGridResult.ColumnHeadersHeightSizeMode =
System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;
            this.dataGridResult.Columns.AddRange(new
System.Windows.Forms.DataGridViewColumn[] {
            this.TitleColumn,
            this.CColumn1,
            this.CColumn2,
            this.CColumn3,
            this.CColumn4,
            this.CColumn5,
```

```
            this.CColumn6,
            this.CColumn7,
            this.CColumn8});
            this.dataGridResult.Enabled = false;
            this.dataGridResult.Location = new System.Drawing.Point(52,
68);
            this.dataGridResult.Margin = new
System.Windows.Forms.Padding(4);
            this.dataGridResult.MultiSelect = false;
            this.dataGridResult.Name = "dataGridResult";
            this.dataGridResult.RightToLeft =
System.Windows.Forms.RightToLeft.No;
            this.dataGridResult.RowHeadersVisible = false;
            this.dataGridResult.RowHeadersWidth = 40;
            this.dataGridResult.Size = new System.Drawing.Size(558,
250);
            this.dataGridResult.TabIndex = 26;
            // CColumn7
            //
            this.CColumn7.HeaderText = "";
            this.CColumn7.Name = "CColumn7";
            this.CColumn7.SortMode =
System.Windows.Forms.DataGridViewColumnSortMode.NotSortable;
            this.CColumn7.Width = 55;
            // ResultData
            //
            this.ClientSize = new System.Drawing.Size(642, 377);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.btnBack);
            this.Controls.Add(this.btnBuild);
            this.Controls.Add(this.dataGridResult);
            this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedSingle;
            this.MaximizeBox = false;
            this.Name = "ResultData";
            this.Text = "ResultData";
            this.FormClosing += new
System.Windows.Forms.FormClosingEventHandler(this.ResultData_FormClosing
);
```

```
            this.Load += new System.EventHandler(this.ResultData_Load);
((System.ComponentModel.ISupportInitialize)(this.dataGridResult)).EndIni
t();
            this.ResumeLayout(false);
            this.PerformLayout();
        }
        #endregion
        private System.Windows.Forms.Button btnBack;
        private System.Windows.Forms.Button btnBuild;
        private System.Windows.Forms.DataGridViewTextBoxColumn CColumn8;
        private System.Windows.Forms.DataGridViewTextBoxColumn CColumn6;
        private System.Windows.Forms.DataGridViewTextBoxColumn CColumn5;
        private System.Windows.Forms.DataGridViewTextBoxColumn CColumn4;
        private System.Windows.Forms.DataGridViewTextBoxColumn CColumn3;
        private System.Windows.Forms.DataGridViewTextBoxColumn CColumn2;
        private System.Windows.Forms.DataGridViewTextBoxColumn CColumn1;
        private System.Windows.Forms.DataGridViewTextBoxColumn
TitleColumn;
        private System.Windows.Forms.DataGridView dataGridResult;
        private System.Windows.Forms.DataGridViewTextBoxColumn CColumn7;
        private System.Windows.Forms.Label label1;
    }
}
```

ДОДАТОК Б

Приклад документу «Результати аналізу»

Метод анализа иерархий

Для анализа цели были введены следующие критерии:

Номер критерия	Название критерия
K1	Часоемкость
K2	Использование ресурсов
K3	Правильная распознаваемость
K4	Обучаемость
K5	Легкость использования

В результате анализа иерархий были получены следующие значения глобальных приоритетов:

Альтернатива	Значение глобального приоритета
A1	0,1967
A2	0,3406
A3	0,3398
A4	0,1228

Полученная иерархия критериев и альтернатив изображена на рис.1, в котором оптимальная альтернатива выделена квадратом красного цвета.

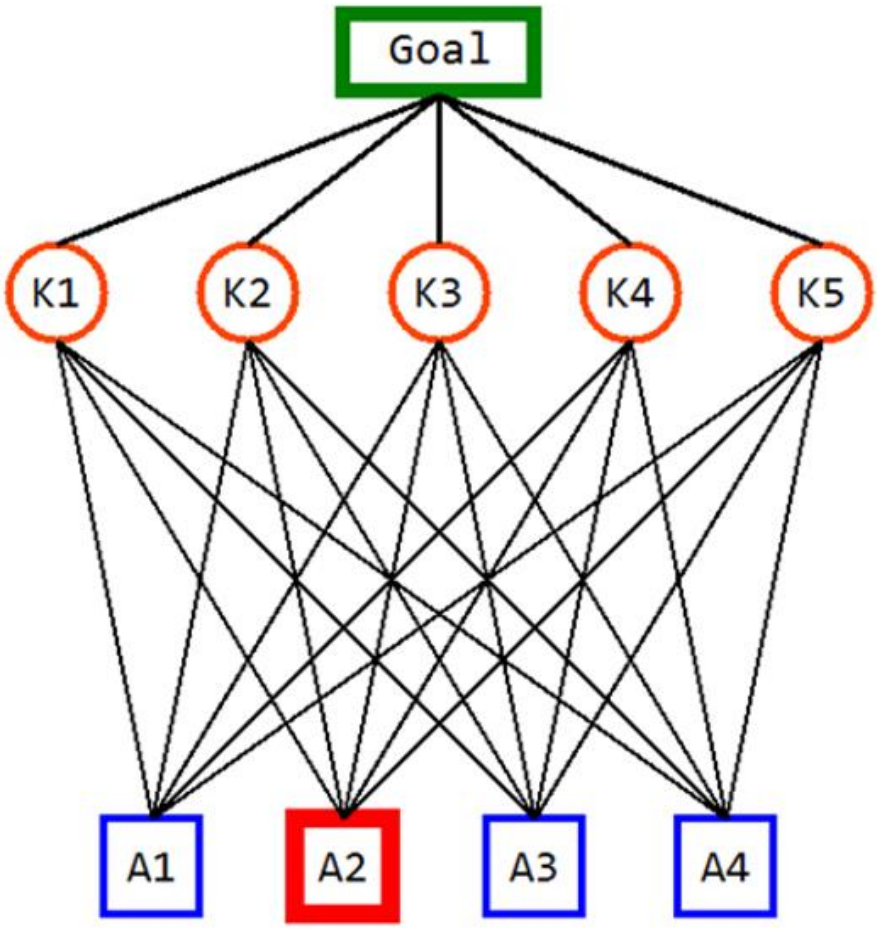


Рисунок 1 – Форма с построенной иерархией альтернатив и критериев

ДОДАТОК В

Копії слайдів презентації

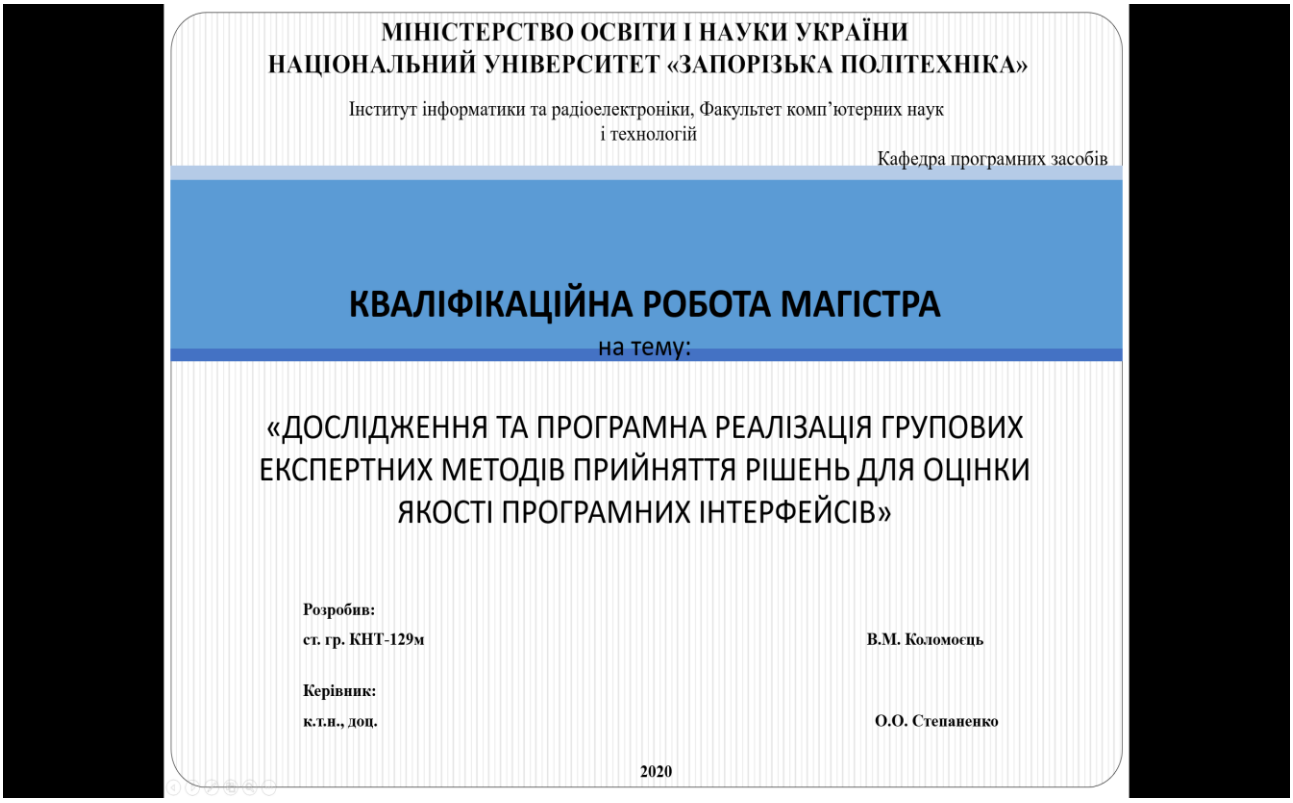


Рисунок В.1 – Слайд №1

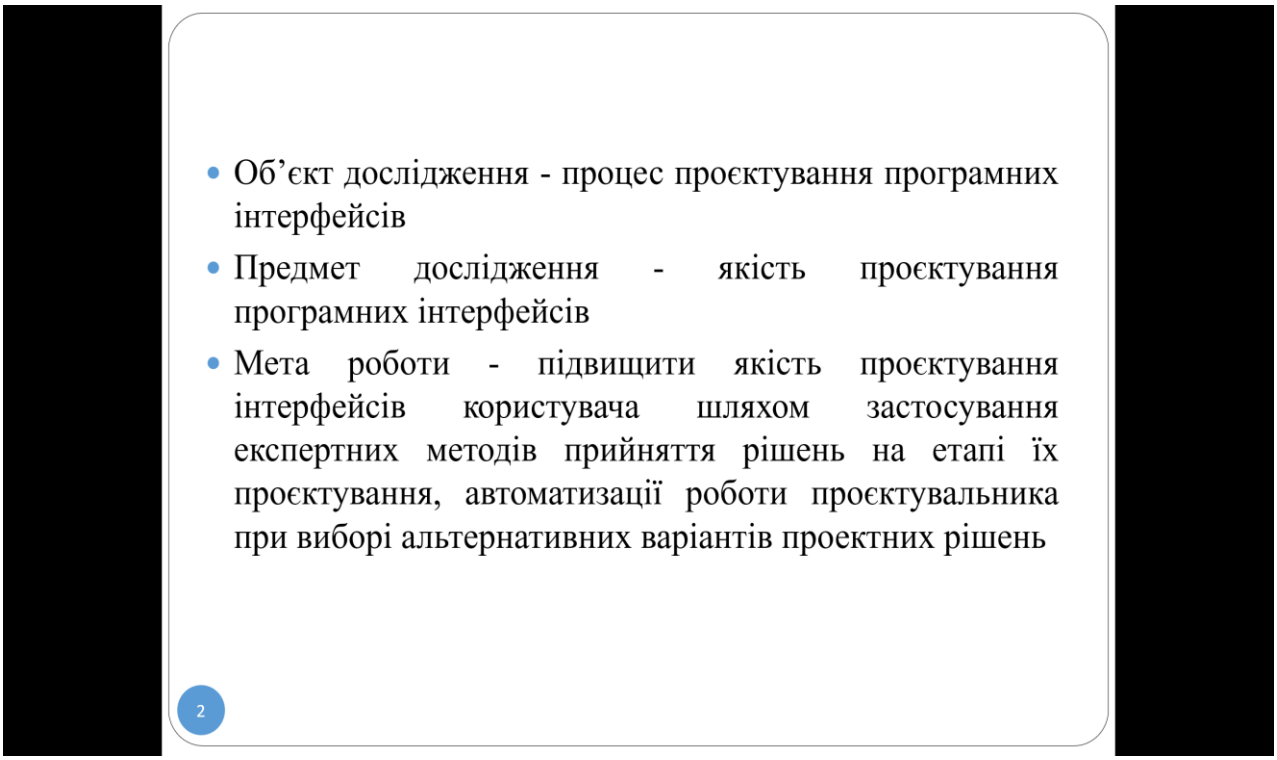


Рисунок В.2 – Слайд №2

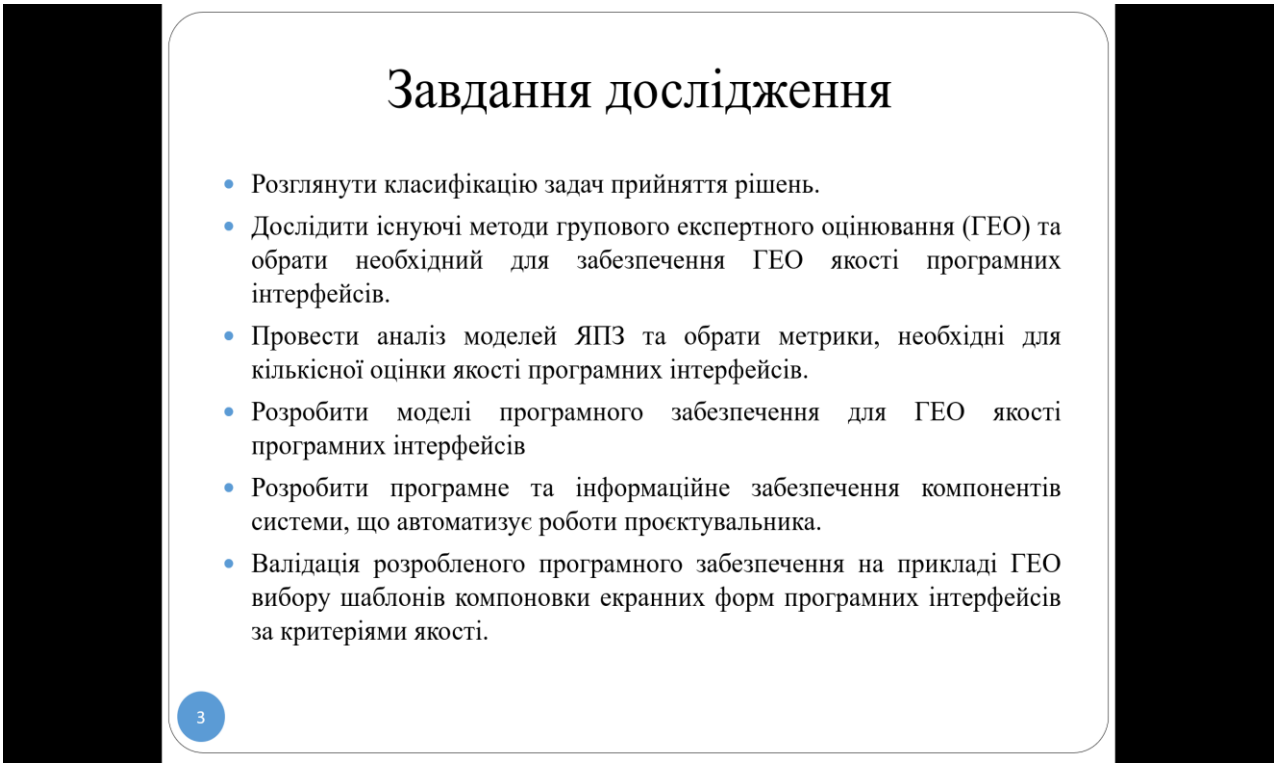


Рисунок В.3 – Слайд №3

Рисунок В.4 – Слайд №4

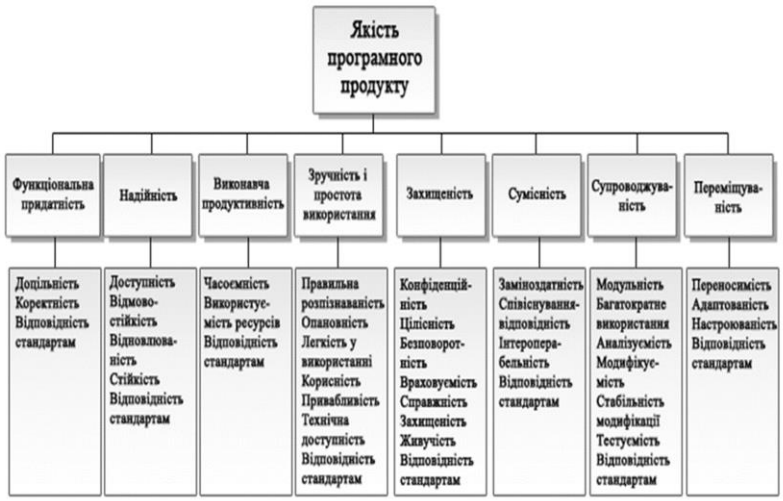
Класифікація методів прийняття рішень

№ п/п	Вміст інформації	Тип інформації	Метод прийняття рішень
1	Експертна інформація не вимагається		Метод домінування Метод на основі глобальних критеріїв
2	Інформація про переваги на множині критеріїв	Якісна інформація	Лексикографічне впорядкування Порівняння різниць критеріальних оцінок Метод «припасовывания» Методи "ефективність-вартість "
		Кількісна оцінка переважності критеріїв	Методи згортки на ієрархії критеріїв Методи порогів
		Кількісна інформація про заміщення	Методи теорії цінності
3	Інформація про переваги на множині критеріїв і про наслідки альтернатив	Відсутність інформації про переваги; кількісна та/або інтервальна інформація про наслідки. Якісна інформація про переваги і кількісна про наслідки. Якісна (порядкова) інформація про переваги та наслідки.	Методи з дискретизацією невизначеності Стохастичне домінування Методи прийняття рішень в умовах ризику і невизначеності на основі глобальних критеріїв Метод аналізу ієрархій Методи теорії нечітких множин Метод практичного прийняття рішень Методи вибору статистично ненадійних рішень
		Кількісна інформація про переваги і наслідки	Методи дерев рішень
4	Інформація про перевагу альтернатив	Оцінка переважності парних порівнянь	Методи математичного програмування Лінійна та нелінійна згортка при інтерактивному способі визначення її параметрів

4

Рисунок В.5 – Слайд №5

Модель якості програмного продукту ISO/IEC 25010.2-2008



6

Рисунок В.6 – Слайд №6

Рисунок В.7 – Слайд №7

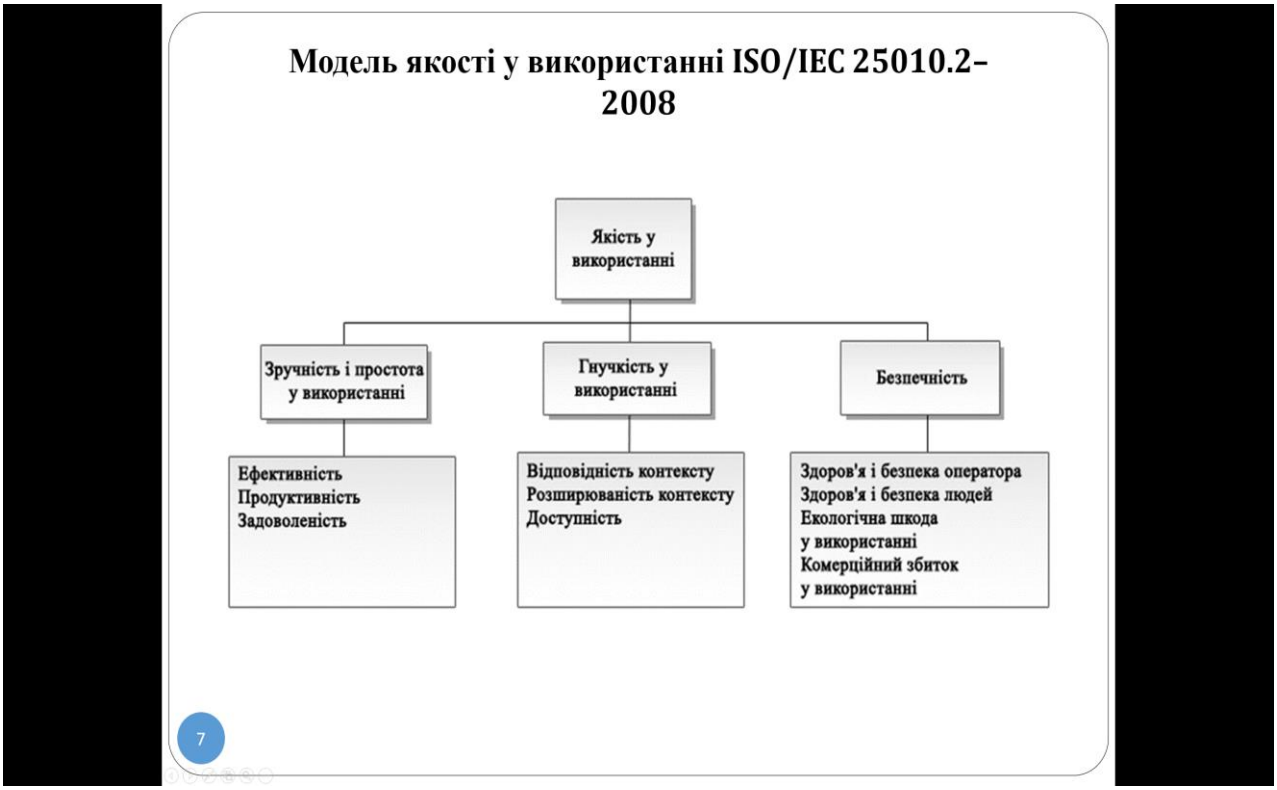


Рисунок В.8 – Слайд №8

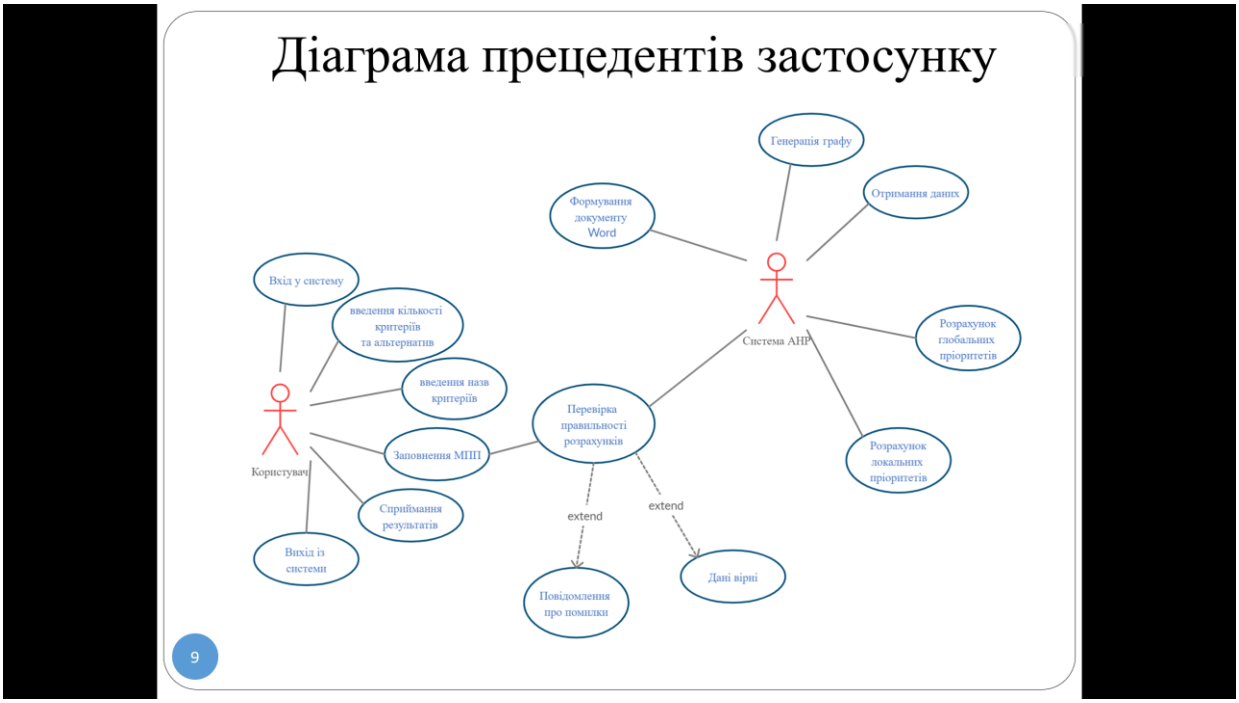


Рисунок В.9 – Слайд №9

Рисунок В.10 – Слайд №10

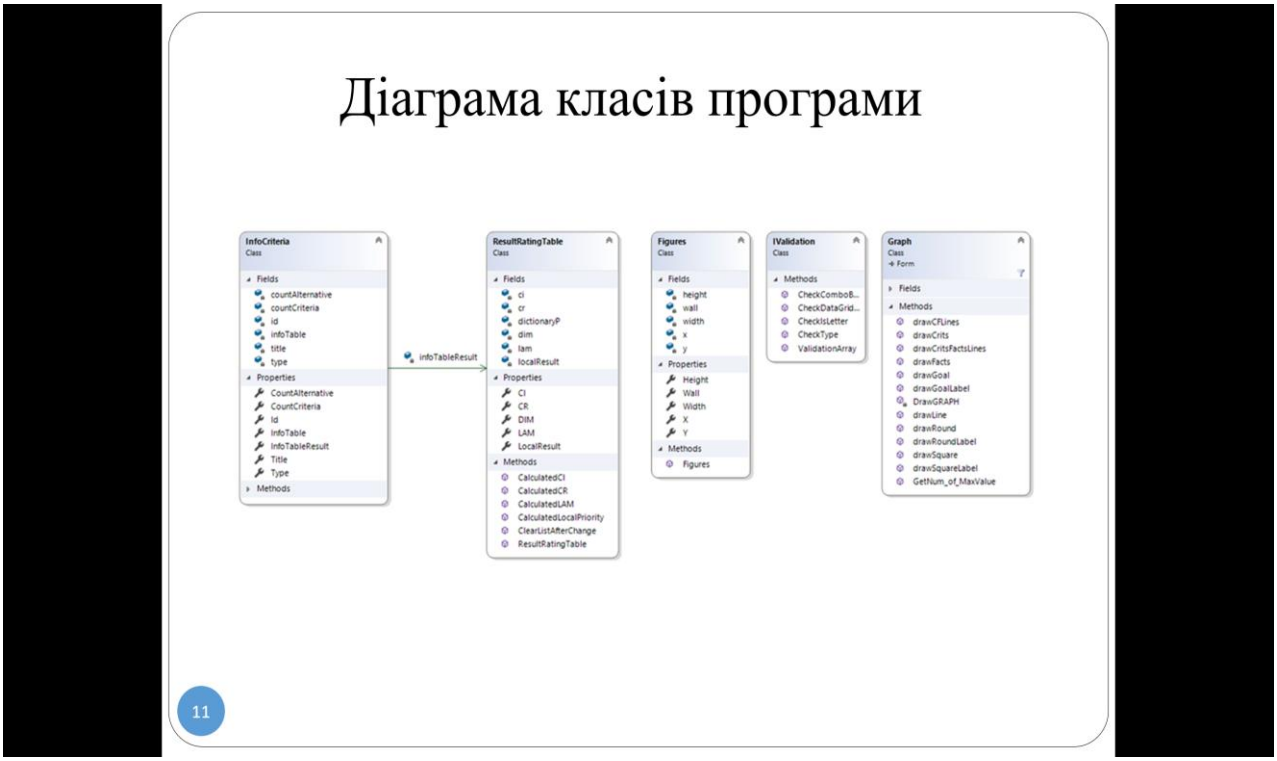


Рисунок В.11 – Слайд №11

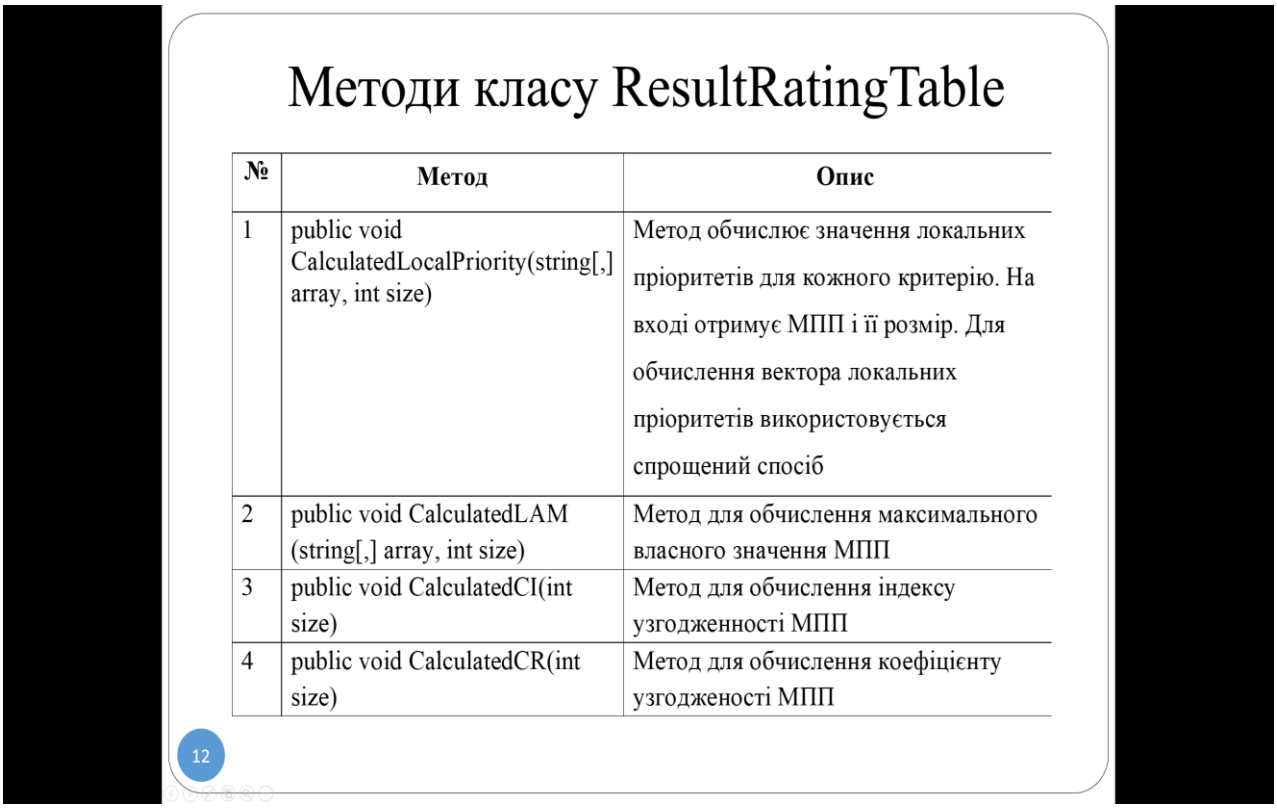
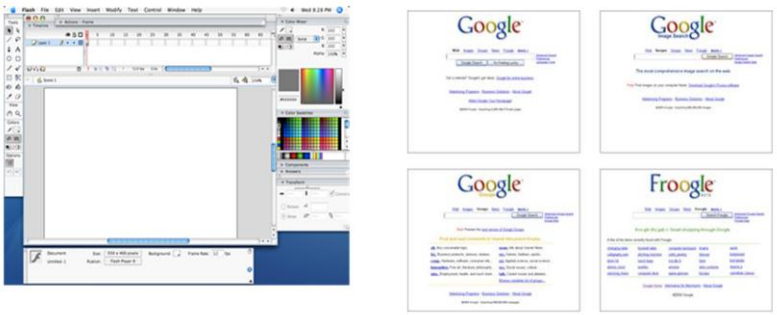


Рисунок В.12 – Слайд №12

Вибрані проєктні альтернативи

- Center Stage (Центральна сцена) - A1;
- Visual Framework (Візуальна схема) - A2;
- Card Stack (Пачка карток) - A3;
- Movable Panels (Панелі, що переміщуються) - A4.



13

Рисунок В.13 – Слайд №13

Вибрані критерії оцінювання

- часоємність (time behaviour);
- використання ресурсів (resource utilisation);
- правильна розпізнаваність (appropriateness recognisability);
- опанованість (learnability);
- легкість використання (ease of use)

14

Рисунок В.14 – Слайд №14

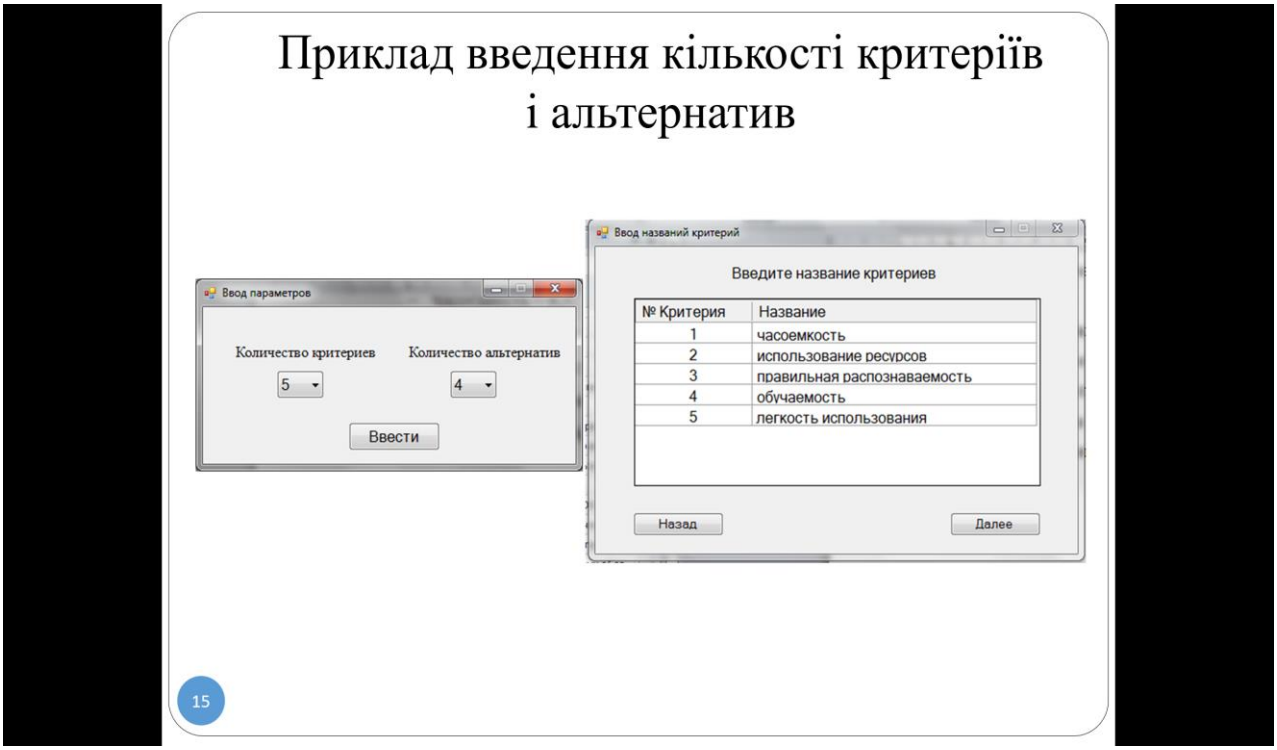


Рисунок В.15 – Слайд №15

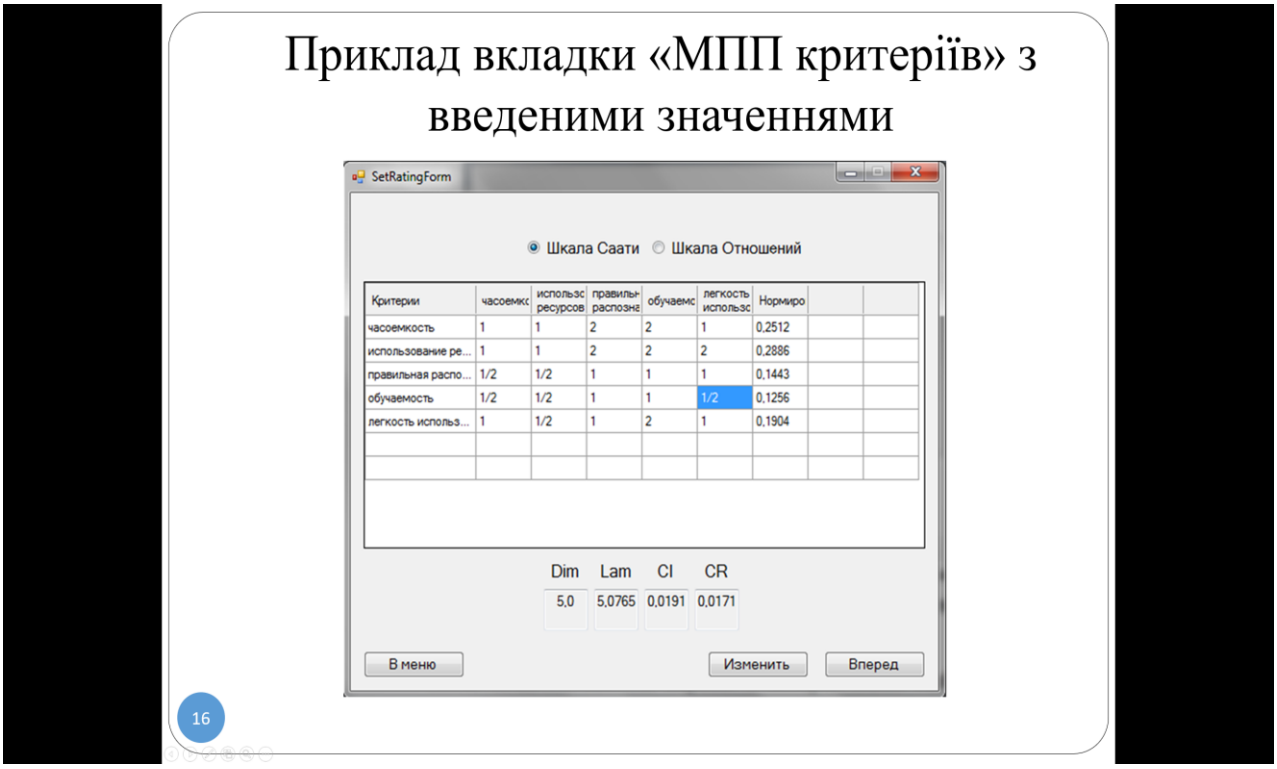


Рисунок В.16 – Слайд №16

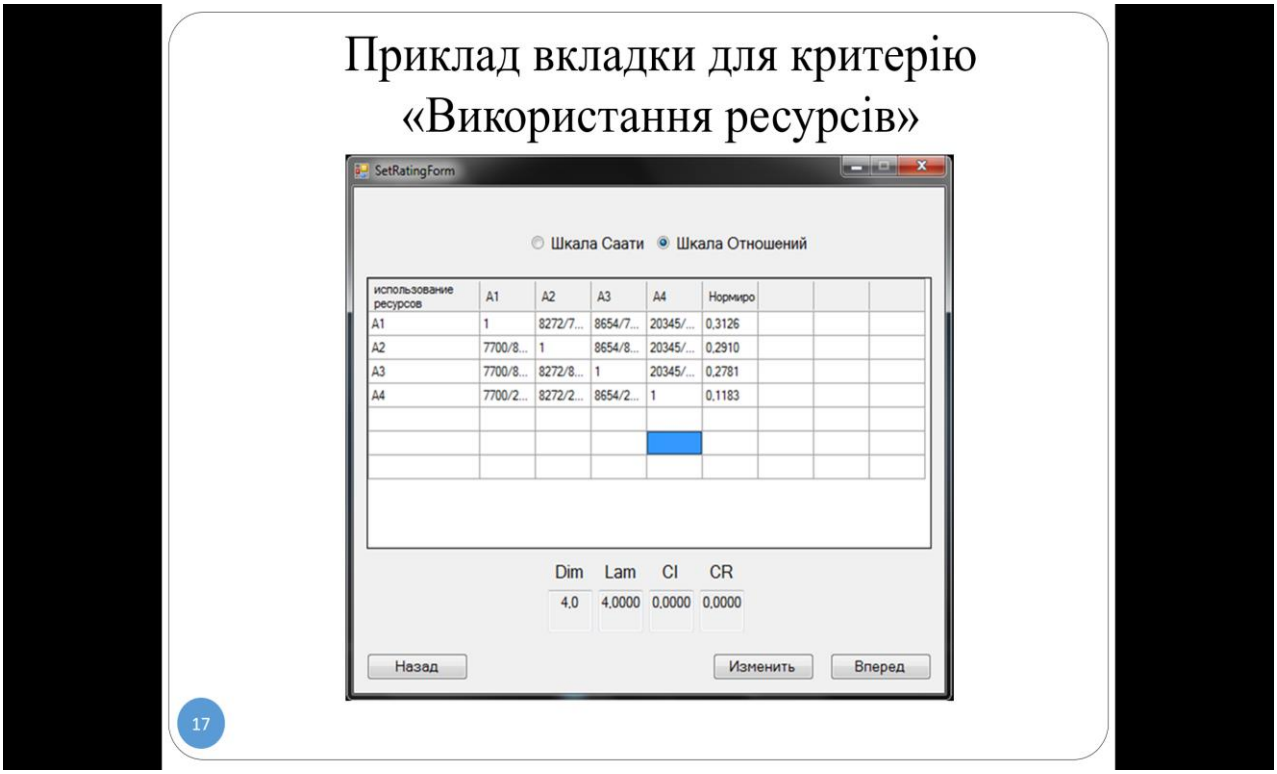


Рисунок В.17 – Слайд №17

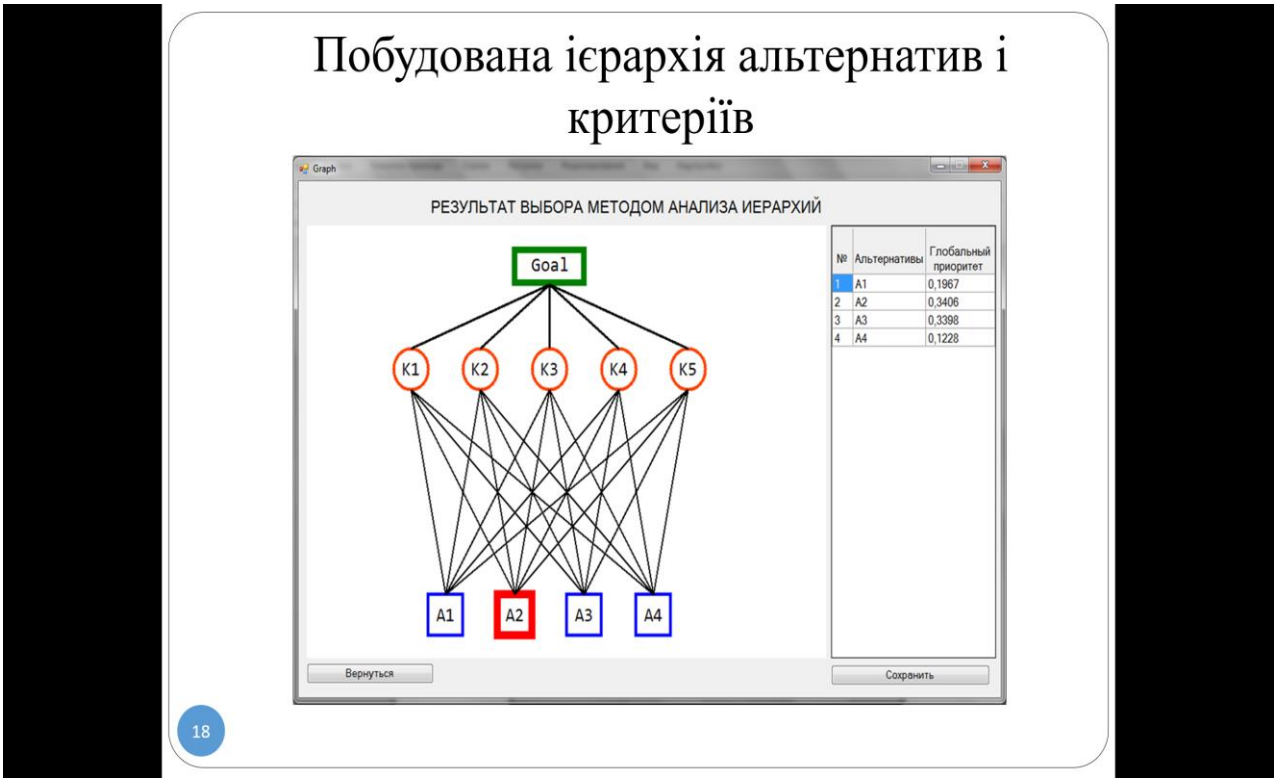


Рисунок В.18 – Слайд №18

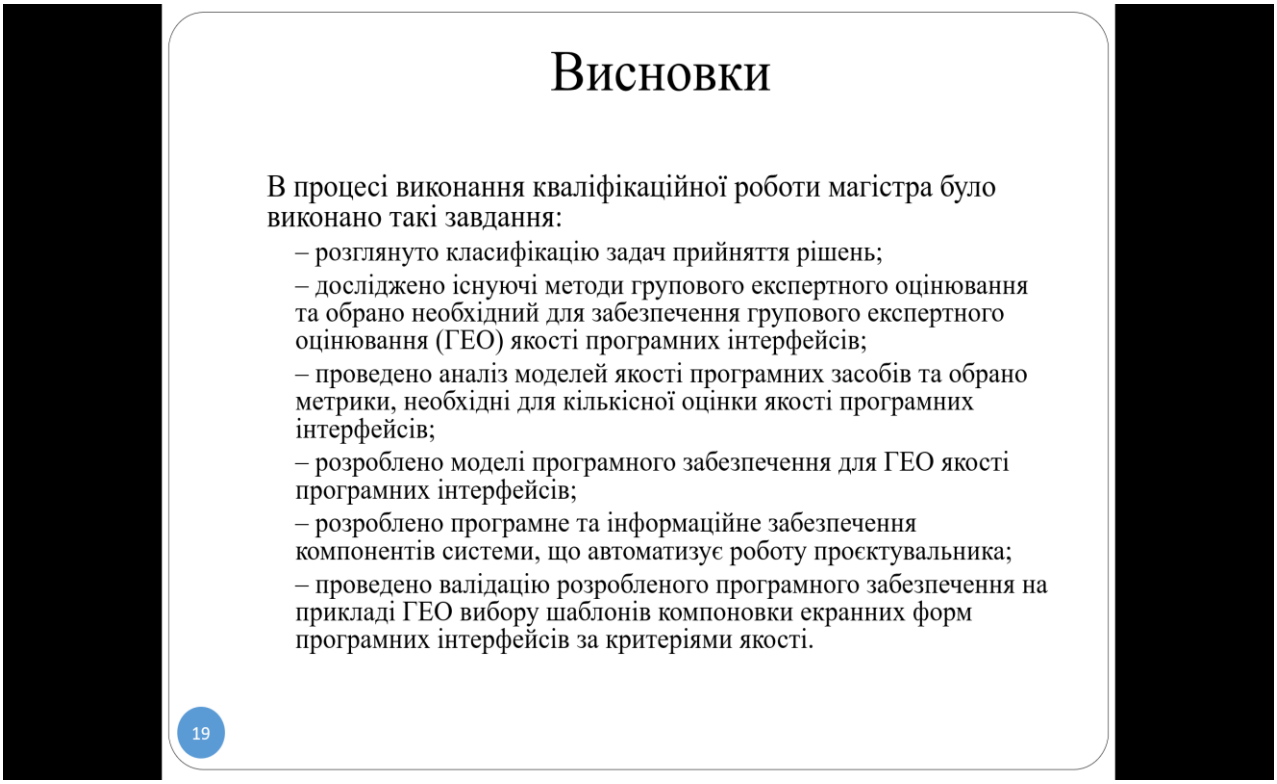


Рисунок В.19 – Слайд №19

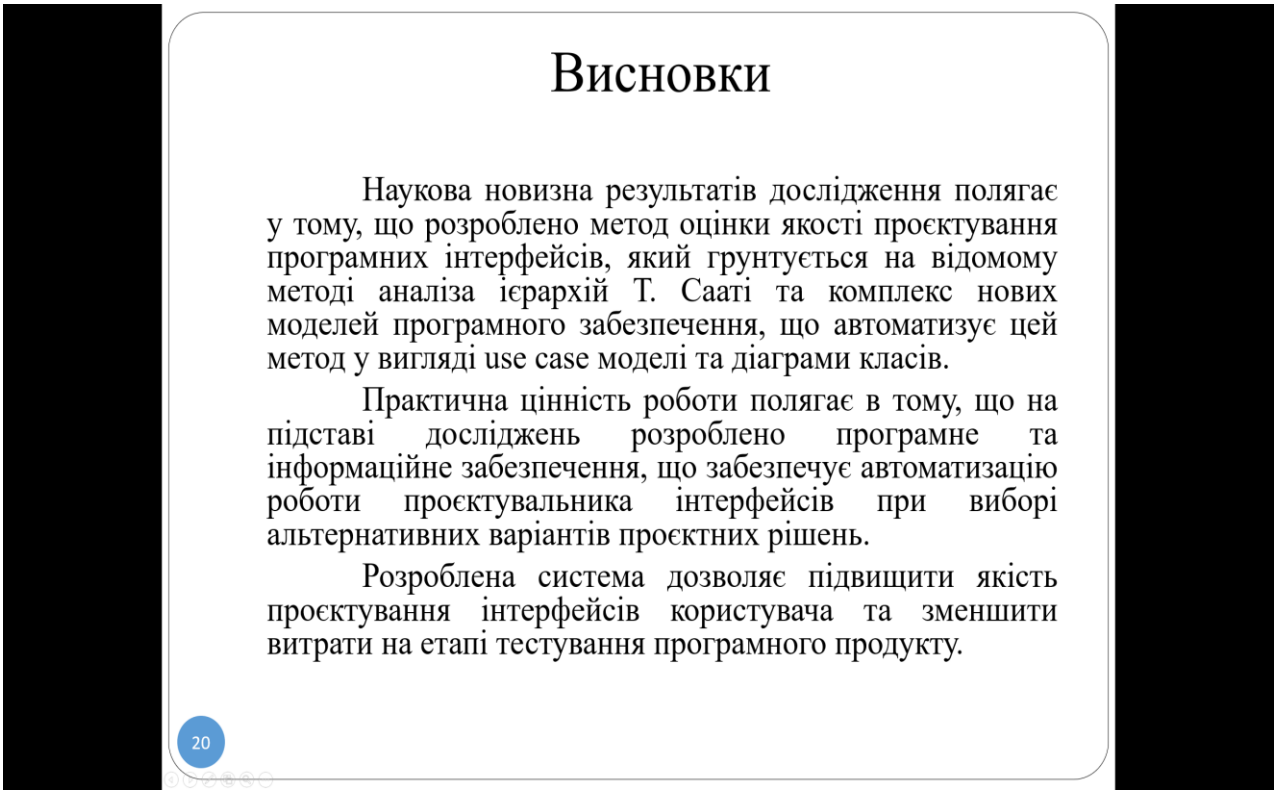


Рисунок В.20 – Слайд №20

Рисунок В.21 – Слайд №21