МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

ФАКУЛЬТЕТ МЕХАТРОНІКИ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Кваліфікаційна магістерська робота

на тему Розроблення інформаційної системи для контролю за використанням програмних засобів підприємства

Виконав: студент групи МгІТ-1-22
спеціальності
122 комп'ютерні науки
освітньої програми
комп'ютерні науки

Матвій Бунтов

Керівник к.т.н., доц. Володимир Яхно

Рецензент д.т.н., проф. Віктор Чупринка

Київ 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

факультет мехатроніки та комп'ютерних технологій

кафедра комп'ютерних наук

Спеціальність 122 Комп'ютерні науки
Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних наук

_____ Володимир Щербань

"____"_____2023 року

З А В Д А Н Н Я

НА ДИПЛОМНУ МАГІСТЕРСЬКУ РОБОТУ
СТУДЕНТУ

Бунтову Матвію Івановичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення інформаційної системи для контролю за використанням програмних засобів підприємства. Науковий керівник роботи Яхно Володимир. Михайлович, к. т .н., доц.
затверджений наказом вищого навчального закладу 12 . 09.2023 року , № 210-уч.

2. Строк подання студентом роботи 14.11.2023 р.

3. Вихідні дані до роботи Розробка кафедри інформаційних технологій проектування. Математичні методи прийняття керуючих рішень. Задачі керування технологічними ресурсами. Принципи Domain-driven design проектування та побудови програмних засобів.

4. Зміст дипломної роботи (перелік питань, які потрібно розробити) : РОЗДІЛ 1 (Теоретичний. Постановка задачі, огляд літератури і предмет дослідження); РОЗДІЛ 2 (обґрунтування інформаційних, математичних моделей, принципових моделей даних та методів); РОЗДІЛ 3 (алгоритмічне і програмне забезпечення системи).

5. Консультанти розділів дипломної магістерської роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ1	Володимир Яхно, к.т.н., доц		
Розділ 2	Володимир Яхно, к.т.н., доц.		
Розділ3	Володимир Яхно, к.т.н., доц.		
Висновки	Володимир Яхно, к.т.н., доц		

6. Дата видачі завдання 10.9.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної магістерської роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	5.10.2023	
2	Розділ 1 (Постановка задачі та методи дослідження)	5.10.2023	
3	Розділ 2 (Обґрунтування моделей та методів)	5.10.2023	
4	Розділ 3 (Алгоритмічне та програмне забезпечення)	10.10.2023	
5	Висновки	25.10.2023	
6	Оформлення дипломної магістерської роботи (чистовий варіант)	30.10.2023	
7	Здача дипломної магістерської роботи на кафедру для рецензування (за 14 днів до захисту)	4.11.2023	
8	Перевірка дипломної магістерської роботи на наявність ознак плагіату (за 10 днів до захисту)	8.11.2023	
9	Подання дипломної магістерської роботи на затвердження завідувачу кафедри (з 7 днів до захисту)	14.11.2023.	

Студент _____ **Матвій БУНТОВ**
(підпис)

Науковий керівник роботи _____ **Володимир Яхно**
(підпис)

Директор НМЦУПФ _____ **Олена ГРИГОРЕВСЬКА**
(підпис)

АНОТАЦІЯ

Бунтов М. І. Розроблення інформаційної системи для контролю за використанням програмних засобів підприємства. .
– Рукопис.

Дипломна магістерська робота за спеціальністю 122 комп'ютерні науки та інформаційні технології – Київський національний університет технологій та дизайну, Київ, 2023 рік.

Розглянуті теоретичні аспекти та питання практичної реалізації прийняття рішень для контролю за використанням програмних засобів підприємства і розв'язання задач оновлення специфічного виду обладнання – програмних засобів. Інформаційна система для контролю за використанням програмних засобів підприємства є основою для формування моделей прийняття рішень параметри моделей є не чіткими. Інформаційна система містить інформацію для методів узгодження рішень та координації між особами, які приймають рішення оновлення програмних засобів. Кожний керівник відповідає за деяку (важливу) частину загальної проблеми рішень оновлення та обслуговування програмних засобів. Вірне рішення має вирішальне значення для досягнення загальної продуктивності підприємства. В роботі наведені принципи формування та функції програмних засобів системи прийняття рішень для реалізації задач оновлення програмного забезпечення на принципах використання нечітких уявлень про оцінки якості та ефективність програмного забезпечення.

Ключові слова: оновлення програмного забезпечення, керування запасами, моделі даних, автоматизована система

ANNOTATION

untov M. I. Development of an information system for monitoring the use of enterprise software tools. .

- Manuscript.

Master's thesis in the specialty 122 computer science and information technologies - Kyiv National University of Technology and Design, Kyiv, 2023.

Considered theoretical aspects and issues of practical implementation of decision-making to control the use of software tools of the enterprise and solving the problems of updating a specific type of equipment - software tools. The information system for monitoring the use of software tools of the enterprise is the basis for the formation of decision-making models, the parameters of the models are not clear. The information system contains information for the methods of agreement of decisions and coordination between the persons who make the decision to update the software. Each manager is responsible for some (important) part of the overall problem of updating and maintaining software solutions. The right decision is critical to achieving the overall productivity of the enterprise. The paper presents the principles of formation and functions of software tools of the decision-making system for the implementation of software update tasks based on the principles of using vague ideas about software quality and efficiency assessments.

Keywords: software update, inventory management, data models, automated system

Зміст

Вступ

Актуальність теми. Проблема обробки інформації набуває особливого значення для ефективності менеджменту та виробничої діяльності сучасного конкурентного середовища. Проблема обробки інформації потребує розроблення та удосконалення систем аналітичного забезпечення [7; 11]. В більшості випадків найбільш важливою частиною цього процесу є програмне забезпечення та комп'ютерне обладнання підприємства. Проблема координації формування та прийняття рішень по доцільності використання та оновлення конкретного програмного забезпечення є завданням в умовах невизначеності. Визначення та оновлення необхідного програмного забезпечення та планування модернізації програмного та комп'ютерного забезпечення підприємства майже завжди виконується в умовах розбіжності думок про важливе питання – чи дійсно потрібен новий комп'ютер або новий програмний засіб. Планування періодичного оновлення та придбання нових програмних продуктів та технічних засобів повинно базуватися на реальних економічних можливостях і умовах створення дійсно необхідних змін функціонування комп'ютерного та програмного обладнання.

Вибір програмного забезпечення є важливим завданням для багатьох організацій та осіб. Цей процес може бути дуже складним, оскільки існує велика кількість програмних продуктів з різними функціями, характеристиками та вартістю. Для вирішення цієї задачі можна використовувати різні методи та підходи Дослідити можливість використання нових технологій керування, на принципах, що надає поле теорії прийняття рішень [11] мета роботи. Також актуальним є дослідити і, можливо, обґрунтувати ефективність методів, що використовують принципи поля теорії прийняття рішень, для планування календарних планів оновлення програмних засобів та запропонувати програму для інформаційної підтримки цієї задачі.

Мета дослідження. Мета дослідження – розробити програмний засіб, що забезпечить інформаційну допомогу процесу розробки рекомендацій по визначенню та оновленню програмного забезпечення та технічного обладнання комп'ютерної мережі підприємства.

Для досягнення поставленої мети в роботі необхідно вирішити наступні проблеми:

- Дослідити сучасні стратегії по визначенню та оновленню програмного забезпечення та технічного обладнання комп'ютерної мережі підприємства. Розробити концептуальні та логічні моделі що є необхідними для збереження, відображення, редагування даних;
- реалізувати принципи розподілених процесів та правил домену Domain-driven design для формування рішень задач застосування та оновлення програмного забезпечення;
- для реалізації моделей та алгоритмів обрати засоби, що реалізують найбільш ефективні та зручні технології програмування;
- Обґрунтувати вибір та використати інструментальні та апаратні засоби програмування з локалізацією до української мови.

Завдання дослідження. Реалізувати інформаційну систему для координації між особами, які відповідають за свою частину загальної проблеми по визначенню та оновленню програмного забезпечення та технічного обладнання комп'ютерної мережі підприємства.

Необхідними є звичайні і стандартні вимоги, що корисними (обов'язковими) для всіх програмних засобів:

- мінімальні витрати для використання та нескладність операцій використання та розгортання ;

- інтуїтивна зрозумілість та практична наочність представлення результатів.

Об'єкт дослідження. Задачі та моделі прийняття рішень по визначенню та оновленню програмного забезпечення та технічного обладнання комп'ютерної мережі підприємства. Координація рішень у проблемно-орієнтованих інформаційних системах та системах керування

Предмет дослідження. Предметом дослідження є принципи побудови інформаційних систем для підтримки прийняття узгоджених рішень працівниками, кожен з яких відповідає за персональну частину загальної проблеми прийняття рішень.

Методи дослідження. Основними методами дослідження проблеми по визначенню та оновленню програмного забезпечення та технічного обладнання комп'ютерної мережі підприємства є програмування і моделі та математичні методи дослідження операцій. Для практичної реалізації задачі використані принципи та шаблони програмних технологій стандартних патернів програмування NET.

Практична цінність - дозволяє покращити якість інформації про виконання планів оновлення та комплектації програмних засобів.

Елементи наукової новизни. Результати дозволяють отримати обґрунтований набір вимог до параметрів значного класу алгоритмів. Автору роботи програми, що базуються на подібних принципах і мають наведені характеристики не відомі.

Практична значущість роботи. Описані в даній роботі програмний продукт для реалізації принципів планування по визначенню та оновленню програмного забезпечення та технічного обладнання комп'ютерної мережі підприємства є засобами, що може бути використаними для важливих задач планування виробництва. Програми, що базуються на подібних принципах і мають наведені характеристики не відомі.

Апробація результатів роботи. Результати роботи програми перевірені на прикладах відповідних задач планування..

РОЗДІЛ 1. ЗАДАЧА ТА МЕТОДИ ДОСЛІДЖЕННЯ

1.1 Формулювання проблеми інформаційної підтримки задачі оновлення програмного забезпечення

Вибір програмного забезпечення є важливим завданням для багатьох організацій та осіб. Цей процес може бути дуже складним, оскільки існує велика кількість програмних продуктів з різними функціями, характеристиками та вартістю. Правильний вибір є важливою частиною процесу підвищення якості і ефективності керування (взагалі це проблема обробки інформації [1,2]). Правильний вибір потребує розроблення та удосконалення системи аналітичного підґрунтя менеджменту. Цим підґрунтям є програмне та комп'ютерне забезпечення підприємства. Вибір програмного забезпечення це задача прийняття рішень за багатьма атрибутами, що потребу оцінювати альтернативи на основі атрибутів які не можуть бути просто об'єднані в одну цільову функцію, оскільки вони несумірні. Наприклад, важливий атрибут вартість або швидкість реагування є кількісним і його вимірюють числові одиниці, а атрибут зручність є якісним і не має числові оцінки. Для вирішення цієї задачі можна використовувати різні методи та підходи.

Типова послідовність кроків у виборі програмного забезпечення наступна[4]:

1.Необхідно визначити потреби та цілі: З'ясуйте, які конкретні завдання ви хочете вирішити за допомогою програмного забезпечення. Необхідно визначити основні вимоги та функції, які повинні бути у програмі. В більшості випадків це відбувається в умовах неузгодженості думок осіб приймаючих рішення з питання.

2.Аналіз ринку: Дослідження ринку програмного забезпечення, -різні продукти, що відповідають потребам. Необхідно розглянути як популярні

продукти, так і менш відомі, але можливо, що вони мають відмінні функції або вартість.

3. Порівняння функції та вартість: Порівняння функціональні можливості продуктів і їх вартість. Необхідно обрати продукт, який найкраще відповідає вашим потребам і бюджету. В більшості випадків це також відбувається в умовах неузгодженості думок осіб приймаючих рішення з питання. Рішення може бути оптимальним.

Наприклад. Якщо $J \in J_M = \{1, ..., M\}$ роботи j-го виду. I для роботи визначена упорядкованість $УС_{ij}$ – оцінка ефективності виконання роботи j технічно - програмним засобом і, та витрати на впровадження $ВПС_{ij}$ (це теж може бути упорядкована ієрархія). X_{IJ} визначає вибір програмного засобу для роботи

Функція мети матиме вигляд

$$F = \sum_{I=1}^{N}\sum_{J=1}^{N} F(УС_{IJ})X_{IJ} \rightarrow MAX . \qquad 1.1$$

Повинні виконуватись такі умови:

– на кожну роботу (програмну операцію) призначається тільки один виконавець

$$\sum_{J=1}^{N} ВПС\ (X_{IJ}Ï = 1\ \forall I \in I_N\ .$$

$$– X_{ij} \in \{0;1\} ..$$

Всі наведені функціональні залежності моделі (1.1) можуть бути визначені лише методом експертних оцінок.

4.Оцінка користувацького досвіду: Користувацький досвід може бути критично важливим для продуктивності та зручності використання програми. В більшості випадків це теж відбувається в умовах неузгодженості думок осіб приймаючих рішення з питання.

5.Підтримка та оновлення: Важливо переконатися, що вибране програмне забезпечення має надійну технічну підтримку та регулярні

оновлення для виправлення помилок і покращення функціоналу. В більшості випадків це теж відбувається в умовах неузгодженості думок осіб приймаючих рішення з питання

Методи визначення необхідного програмного забезпечення та планування оновлення комп'ютерного забезпечення підприємства завжди використовують в умовах неузгодженості думок приймаючих рішення з питання – чи потрібен новий комп'ютер (і який?) або новий програмний засіб(і який?).

Кожний етап потребує узгодження з усіма зацікавленими сторонами: Якщо ви обираєте програмне забезпечення для бізнесу чи організації, важливо взяти до уваги думку всіх зацікавлених сторін. Узгодьте рішення з іншими користувачами. Для цього використовують методи узгодження рішень.

Методи узгодження рішень - це набір технік, які допомагають вирішувати проблеми, що виникають при прийнятті рішень. Ці методи можуть бути використані для зменшення конфліктів між різними сторонами, забезпечення більш ефективного прийняття рішень та покращення комунікації між учасниками процесу. Методи узгодження рішень декларують стратегію поступового зменшення відстані між позиціями учасників. В більшості випадків застосовується дискретна відстань (Hamming distance) або відстань Махаланобіса.

Дискретна відстань (Hamming distance): Використовується для вимірювання відстані між двома рядками однакової довжини, які складаються з символів з обмеженого алфавіту.

Відстань Махаланобіса: Цей метод використовується для вимірювання відстані між точкою та центроїдам у мультимодальних або ганусових розподілах даних.

Обираючи метод визначення відстані, враховують природу даних і вимоги конкретної задачі. Розуміння та правильний вибір метрики відстані є ключовими для успішного вирішення багатьох аналітичних завдань.

Модель якості програмного забезпечення має чотири рівні подання [1]

Перший рівень відповідає визначенню характеристик (показників) якості програмного забезпечення (ПЗ), кожна з яких відображає окрему точку зору користувача на його якість. Згідно з наявними стандартами в модель якості входить шість характеристик/показників якості ПЗ, функціональність (Functionality), надійність (Realibility), зручність використання (Usability), супроводжуваність (Maitainnability), ефективність (Efficiency), переносність (Portability).

На другому рівні визначають атрибути якості ПЗ для кожної конкретної його характеристики, які деталізують різні її особливості. Набір цих атрибутів потім використовують в метричному аналізі якості ПЗ.

Третій рівень призначений для вимірювання якості

за допомогою метрик, кожну з яких, згідно з стандартом ISO/IEC 9126, визначають як комбінацію методів

вимірювання атрибута і шкали встановлення його значень. Для оцінювання атрибутів якості ПЗ на всіх етапах його розроблення (при перегляді документації та

продуктів проекту, а також за результатами їхнього тестування) використовують метрики з заданою їх вагомістю для нівелювання результатів метричного аналізу сукупних атрибутів конкретного критерію чи показника

якості ПЗ. Атрибут якості визначають за допомогою однієї або декількох методик якості ПЗ на різних етапах

його розроблення і на завершальному етапі його тестування.

На четвертому рівні для оцінювання кількісного

або якісного значення окремих атрибутів якості ПЗ використовують оцінний елемент метрики – її пріоритет.

Залежно від призначення, особливостей експлуатації та

умов супроводу ПЗ вибирають найбільш важливі характеристики і атрибути його якості (рис. 3). Вибрані атрибути і їх пріоритети відображають у вимогах до процесу розроблення ПЗ, або використовують відповідні пріоритети еталона класу ПЗ, до якого воно має належати.

Програмне забезпечення можна класифікувати за різними критеріями. Основні види програмного забезпечення включають [7]:

Системне програмне забезпечення: це базовий рівень програмного забезпечення, який включає драйвери, операційні системи та інші програми, які забезпечують взаємодію інших програм з базовими програмами та апаратними засобами.

Прикладне програмне забезпечення: це програмне забезпечення, яке забезпечує виконання конкретних завдань на комп'ютері, таких як текстові та графічні редактори, диспетчери файлів, WEB редактори, архіватори даних, WEB браузери та інші.

Інструментальне / сервісне програмне забезпечення: це програмне забезпечення, призначене для використання в ході створення архітектури, розробки, оновлення та інсталяції програм. Прикладом є середовища розробки.

Під час оновлення та модернізації програмного забезпечення враховувати такі фактори, як функціональність, сумісність, ціна, легкість використання, підтримка та безпека [11]. Найкращий варіант для потреб організації забезпечить розв'язок задачі (1).

Рисунок. 1.1.1 Формальна модель вибору.

Для ефективного прийняття рішень необхідно визначитися з тим що потрібно одержати в результаті: один єдиний варіант або безліч найкращих варіантів, тому що від цього буде залежати яким критерієм, векторним або скалярним треба користуватись.

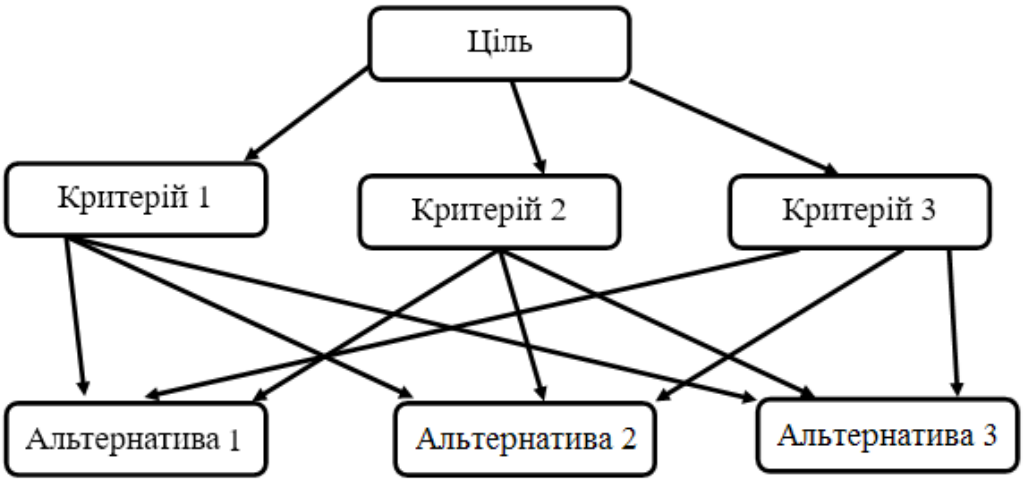


Рисунок. 1.1.2 Формальна модель вибору з багатьох альтернатив.

Метод аналізу ієрархій (MAI) - це математичний метод прийняття рішень, розроблений вченим Томасом Сааті. Він дозволяє упорядковувати та порівнювати альтернативи вибору на основі їх важливості та відносної

переваги відносно один одного. Це єдиний метод що може бути застосований для формулювання задачі (1).

Яскраво виражений для більшості користувачів індивідуальний характер використання та оцінки програмних продуктів (адресність) —дуже важлива характеристика програм. Кожний користувач має своє індивідуальне уявлення про предметну область. Інформація з оцінки програмних продуктів має для нього виключно «індивідуальну цінність». Отже, завдяки адресності та старінню (втраті актуальності з часом) задовольнити потреби в ІПП один раз і назавжди неможливо.

Порівняльна оцінка програмних продуктів дуже ускладнена через відсутність для користувача інструментарію для реалізації вибору того або іншого продукту, який задовольняє його запити. Необхідно розробити спеціальні методи оцінювання програмних продуктів.

1.2 Існуючі технології дослідження

Вибір програмного забезпечення - це важлива стратегічна рішення, яке може вплинути на продуктивність та результативність вашої роботи чи бізнесу. Вибір програмного забезпечення також повинен бути обґрунтованим аналізом використання програмного забезпечення. Аналіз використання програмного забезпечення - це процес вивчення та оцінки того, як саме програмне забезпечення використовується в організації або проекті. Цей аналіз може надати цінну інформацію про те, як програмне забезпечення впливає на бізнес-процеси, продуктивність, задоволення користувачів та інші аспекти роботи

Аналіз використання програмного забезпечення відбувається відповідно до наступних етапів[7]:

Інвентаризація програмного забезпечення: Складіть список всього програмного забезпечення, яке використовується в вашій організації. Це

може включати операційні системи, офісні пакети, програми для управління проектами, обліку, CRM-системи тощо.

Визначення використання: Для кожного програмного продукту необхідно визначити, як саме він використовується в організації. Це може бути управління даними, виробничі процеси, обслуговування клієнтів, аналіз даних тощо.

Оцінка ефективності: Необхідно визначити, наскільки ефективно програмне забезпечення використовується для досягнення мети. Це може включати аналіз продуктивності, якість виведених результатів, співвідношення якості та витрат.

Аналіз задоволеності користувачів: Провидять опитування або визначають зворотний зв'язок від користувачів програмного забезпечення, щоб визначити, наскільки користувачі задоволені його використанням. Це може включати оцінку інтерфейсу, швидкості роботи, доступності функцій тощо.

Оцінка вартості та вигоди: Розгляньте вартість програмного забезпечення у порівнянні із здобутими вигодами. Це може включати аналіз рентабельності, вартості утримання, ризики та можливості зекономити кошти.

Визначення потреб у навчанні: Необхідно визначити, чи потрібне навчання користувачам для ефективного використання програмного забезпечення. Це може включати навчання та підтримку користувачів для оптимального використання функцій продукту.

Оцінка безпеки та відповідності: Переконайтеся, що програмне забезпечення відповідає стандартам безпеки та вимогам з законодавства щодо конфіденційності даних.

Рекомендації та вдосконалення: На основі результатів аналізу розробіть рекомендації для вдосконалення використання програмного забезпечення в організації. Це може включати оптимізацію процесів, навчання користувачів,

або навіть заміну програмного забезпечення на більш відповідний або ефективний продукт.

Аналіз використання програмного забезпечення може допомогти організації оптимізувати витрати, підвищити продуктивність та задоволеність користувачів, а також підвищити загальну ефективність роботи.

Кожний етап потребує узгодження рішень.

Методи узгодження рішень - це набір технік, які допомагають вирішувати проблеми, що виникають при прийнятті рішень. Ці методи можуть бути використані для зменшення конфліктів між різними сторонами, забезпечення більш ефективного прийняття рішень та покращення комунікації між учасниками процесу. Узгодження рішень - це процес досягнення консенсусу або спільного рішення в групі людей чи організацій, які можуть мати різні погляди, інтереси та цілі. Математичні моделі узгодження рішень використовуються для вивчення цього процесу та розробки стратегій для досягнення оптимальних результатів. Деякі з найпоширеніших методів узгодження рішень включають:

1. Метод аналогій: цей метод полягає в тому, щоб знайти аналогію між проблемою, яку необхідно вирішити, та іншою проблемою, яка вже була успішно вирішена. Це може допомогти знайти новий погляд на проблему та знайти ефективний спосіб її вирішення. Цей метод дає можливість залучати в обіг на основі подібності і відповідності новий, ще невідомий матеріал, закріплюючи його на фоні відомого, отже, розширює межі нашого пізнання світу поза можливостями конкретного бачення, а за відомими зразками, аналогами.

2. Метод групового прийняття рішень: цей метод полягає в тому, щоб залучити до процесу прийняття рішень групу людей. Кожен учасник групи може запропонувати свої ідеї та думки щодо проблеми, після чого група може спробувати узгодити свої погляди та прийняти спільне рішення.

Існує безліч методів ефективного групового прийняття рішень. Деякі з цих групових процесів прийняття рішень включають:

Правило більшості: якщо більшість членів групи згодні з певним курсом дій, меншість у групі повинна прийняти його. Цей спосіб не вимагає одностайності.

Техніка Дельфі: Метод Дельфі вимагає, щоб група згенерувала список можливих рішень. Потім лідер групи або фасилітатор групи скорочує список і представляє менший список групі, яка потім робить вибір на основі меншого списку. Може бути легше досягти консенсусу, коли варіантів менше.

Зважена оцінка: кожен член групи виконує ранжування можливі рішення кількісно (числа). Лідер групи підраховує результати, і вибір з найбільшою кількістю балів стає рішенням групи.

3. Метод SWOT-аналізу: цей метод полягає в тому, щоб проаналізувати сильні та слабкі сторони організації або проекту, а також можливості та загрози, якими вони можуть зустрітися. Це може допомогти знайти ефективний спосіб вирішення проблеми.

4. Метод дерева розриву: цей метод полягає в тому, щоб створити дерево розриву для проблеми, яку необхідно вирішити. Це може допомогти виділити ключові фактори, яким необхідно приділити увагу для успішного вирішення проблеми. Мета полягає в тому, щоб мінімізувати суму відмінностей між уподобаннями, і середньою значенням уподобань програмних продуктів. Використовується дерево мінімального охоплення та різні алгоритми, які використовуються для його побудови. Дерево, що охоплює — це підмножина графа G, така, що всі вершини з'єднані за допомогою мінімально можливої кількості ребр. Отже, Дерево, що охоплює не має циклів, а граф може мати більше одного дерева, що охоплює.

Приклад.

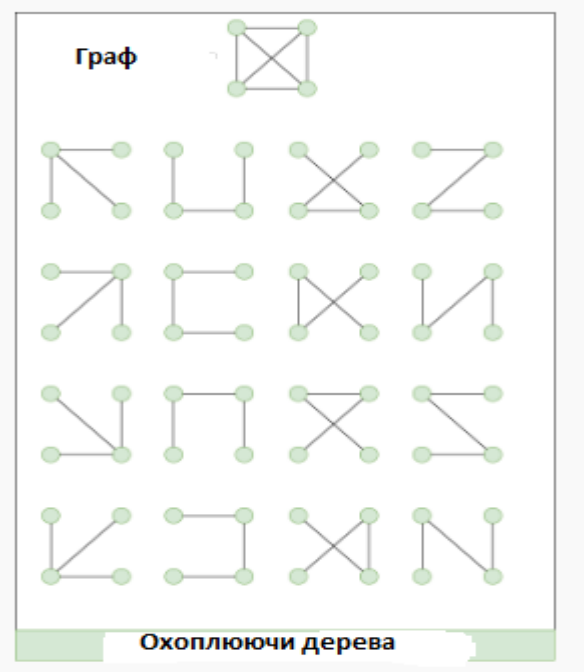


Рисунок 1.2.1

5. Метод Парето-аналізу: цей метод полягає в тому, щоб ідентифікувати 20% факторів, яким необхідно приділити увагу для успішного вирішення проблеми. Це може допомогти сконцентруватися на найбільш важливих факторах та досягти кращих результатів.

Наведені методи дозволяють упорядкувати основні вимоги до програмного забезпечення, що формулюють такими пунктами:

- функціональні вимоги;
- характеристики якості;
- вимоги до документації;
- вимоги по розвитку програмного забезпечення, обслуговуванню і навчанню;
- вимоги по цінах;
- маркетинг програмного забезпечення.

Базовими критеріями для порівняння програмних продуктів можна виділити наступні:

Ефективність: програмні продукти дозволяють автоматизувати багато процесів і завдань, що дозволяє збільшити продуктивність та ефективність роботи.

Економія часу: використання програмних продуктів дозволяє зменшити час, необхідний для виконання рутинних завдань, що дозволяє працювати більше ефективно.

Зручність: програмні продукти надають можливість працювати з даними та інформацією в зручний для користувача спосіб, що полегшує роботу та підвищує комфорт користувача.

Легка модифікація: програмні продукти можуть бути легко модифіковані та вдосконалені відповідно до потреб користувача або змін у бізнес-процесах.

Масштабованість: програмні продукти можуть бути легко масштабовані для використання в різних масштабах – від особистого користування до великих корпоративних систем.

Аналітика та звітність: багато програмних продуктів надають можливість аналізу даних та створення звітів, що дозволяє виробляти обґрунтовані рішення на основі даних.

Забезпечення безпеки: програмні продукти можуть надавати різні засоби захисту даних та інформації від несанкціонованого доступу.

Інтеграція з іншими системами: багато програмних продуктів можуть бути легко інтегровані з іншими системами, що дозволяє створювати комплексні рішення для бізнесу.

Вибір програмного забезпечення це задача прийняття рішень за багатьма атрибутами, що потребу оцінювати альтернативи на основі атрибутів які не можуть бути просто об'єднані в одну цільову функцію, оскільки вони несумірні. Наприклад, важливий атрибут вартість або швидкість реагування є кількісним і вимірюють числові одиниці, а атрибут зручність є якісним і не має числові оцінки. Теорія корисності [7] також надає засоби для оцінки

упорядкованості атрибутів них наслідків і, таким чином є засобом, що полегшує прийняття рішень. Для взаємно незалежних атрибутів функція корисності виражається як зважене підсумовування функцій корисності атрибутів.

Функція корисності це економічна модель для визначення переваг економічних параметрів. Основною концепту функції корисності є раціональна поведінка споживача програмного забезпечення. Перша похідна функції за кількістю певної переваги має назву граничною корисністю цього блага. Гранична корисність визначає кількість збільшення додаткової корисності. Гранична корисність (перша похідна функції), що дорівнює 0, означає досягнення насиченості -максимуму.

.

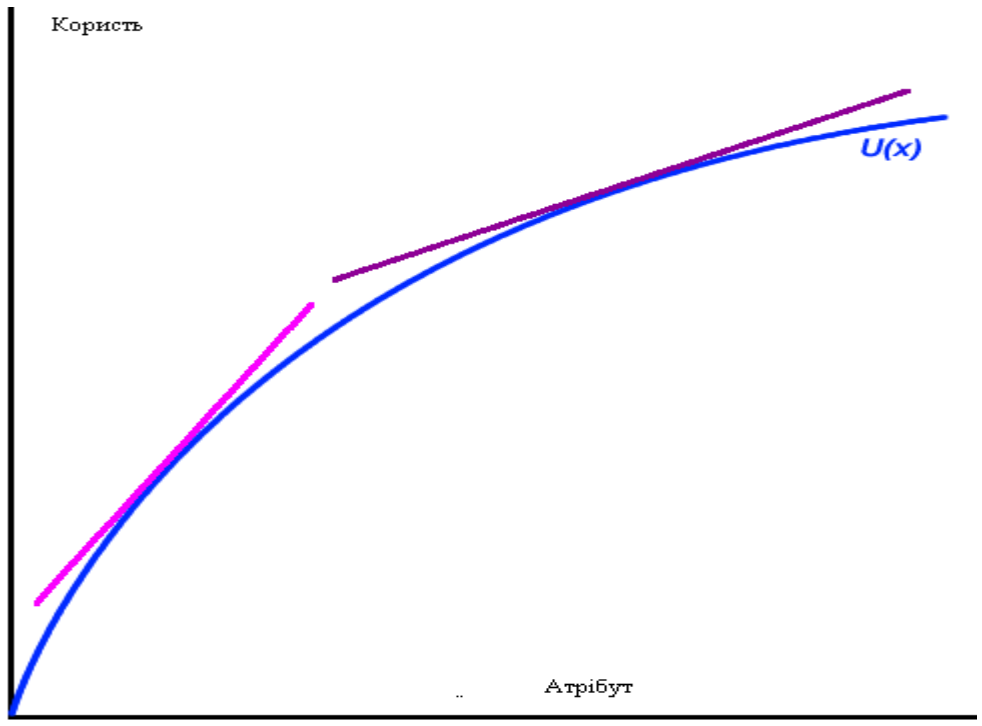


Рисунок 1.2.4 Функція корисності.

Головним обмеженням цієї функції корисності є те, що всі можливі параметри повинні бути перераховані та оцінені. Вимога забороняє використання функції корисності для вибору програмного забезпечення. Але можна застосувати неточну функцію корисності [18], яку можна визначити

на основі підмножини альтернатив. Неточно визначена функція корисності формулює частковий порядок на множині альтернатив.

1.3. Висновки до розділу

Задача оновлення та придбання програмного забезпечення базується на нечіткій інформації про якість та функціональність програм але повинна гарантувати (або бажано) покращувати параметри обробки інформації при обмежених витратах.

Яскраво виражений для більшості користувачів індивідуальний характер використання та оцінки програмних продуктів (адресність) —дуже важлива характеристика програм. Кожний користувач має своє індивідуальне уявлення про предметну область. Інформація з оцінки програмних продуктів має для нього виключно «індивідуальну цінність». Отже, завдяки адресності та старінню (втраті актуальності з часом) задовольнити потреби в ІПП один раз і назавжди неможливо.

Порівняльна оцінка програмних продуктів дуже ускладнена через відсутність для користувача інструментарію для реалізації вибору того або іншого продукту, який задовольняє його запити. Необхідно розробити спеціальні методи оцінювання програмних продуктів.

Розглянуті проблеми вибору програмного забезпечення, що є важливим завданням для багатьох організацій та осіб. Цей процес може бути дуже складним, оскільки існує велика кількість програмних продуктів з різними функціями, характеристиками та вартістю. Правильний вибір є важливою частиною процесу підвищення якості і ефективності керування (взагалі це проблема обробки інформації [1,2]. Правильний вибір потребує розроблення та удосконалення системи аналітичного підґрунтя менеджменту. Цим підґрунтям є програмне та комп’ютерне забезпечення підприємства. Вибір програмного забезпечення це задача прийняття рішень за багатьма атрибутами, що потребу оцінювати альтернативи на основі атрибутів які не

можуть бути просто об'єднані в одну цільову функцію, оскільки вони несумірні. Наприклад, важливий атрибут вартість або швидкість реагування є кількісним і його вимірюють числові одиниці, а атрибут зручність є якісним і не має числові оцінки. Для вирішення цієї задачі можна використовувати різні методи та підходи. Визначена типова послідовність кроків у виборі програмного забезпечення.

Рішення може бути оптимальним (задача 1.1). Наприклад. Якщо $J \in J_M = \{1, ..., M\}$ роботи j-го виду. І для роботи визначена упорядкованість $УС_{ij}$ – оцінка ефективності виконання роботи j технічно - програмним засобом і, та витрати на впровадження $ВПС_{ij}$ (це теж може бути упорядкована ієрархія). X_{IJ} визначає вибір програмного засобу для роботи

Функція мети матиме вигляд

$$F = \sum_{I=1}^{N}\sum_{J=1}^{N} F(УС_{IJ})X_{IJ} \rightarrow MAX .$$

Повинні виконуватись такі умови:

– на кожну роботу (програмну операцію) призначається тільки один виконавець

$$\sum_{J=1}^{N} ВПС\ (X_{IJ}Ï = 1\ \forall I \in I_N\ .$$

$$– X_{ij} \in \{0;1\} ..$$

Але всі наведені функціональні залежності моделі можуть бути визначені лише методом експертних оцінок. Порівняльна оцінка програмних продуктів ускладнена - необхідно надати спеціальні методи оцінювання програмних продуктів

Кожний етап (функціональні залежності) потребує узгодження з усіма зацікавленими сторонами. Тому є необхідність класифікувати програмне забезпечення за різними критеріями.

Під час оновлення та модернізації програмного забезпечення враховувати такі фактори, як функціональність, сумісність, ціна, легкість використання, підтримка та безпека [11]. Найкращий варіант для потреб організації забезпечить розв'язок задачі (1).

Вибір програмного забезпечення також повинен бути обґрунтованим аналізом використання програмного забезпечення. Аналіз використання програмного забезпечення - це процес вивчення та оцінки того, як саме програмне забезпечення використовується в організації або проекті.

Кожний етап потребує узгодження рішень. Кожний користувач має своє індивідуальне уявлення про предметну область. Інформація з оцінки програмних продуктів має для нього виключно «індивідуальну цінність».

В розділі розглянуті базові методи узгодження рішень - це набір технік, які допомагають вирішувати проблеми, що виникають при прийнятті рішень. Ці методи можуть бути використані для узгодження оцінок (зменшення конфліктів) між різними сторонами, забезпечення більш ефективного прийняття рішень та покращення комунікації між учасниками процесу. Узгодження рішень - це процес досягнення консенсусу або спільного рішення в групі людей чи організацій, які можуть мати різні погляди, інтереси та цілі. Розглянуті математичні моделі узгодження рішень використовуються для вивчення цього процесу та розробки стратегій для досягнення оптимальних результатів.

Розділ 2. ОБҐРУНТУВАННЯ МОДЕЛЕЙ ТА МЕТОДІВ

2.1. **Методи побудови формальних моделей задачі оновлення та придбання програмного забезпечення.**

В розділі 1 сформульована модель задачі про вибір програмних та технічних засобів, що базується на типовій задачі про призначення. Задача оновлення та придбання програмного забезпечення базується на нечіткій інформації про якість та функціональність програм але повинна гарантувати (або бажано) покращувати параметри обробки інформації при обмежених витратах.

Аналіз задачі про вибір програмних та технічних засобів вимагають від керівника будь-якого рангу уміння швидко і правильно приймати рішення. При цьому необхідно враховувати вимоги зростання обсягів обробки інформації, подальший розвиток техніки, підвищення вимог до якості обробки інформації та інші фактори.

У цих умовах рішення, що приймаються на основі особистого досвіду і інженерної інтуїції, можуть виявитися малоефективними, оскільки не враховують цілого ряду протидіючих умов. Така ситуація вимагає застосування технології прийняття рішень, яка базується на кількісних оцінках варіантів, і виключає або зменшує значення суб'єктивних факторів і при цьому враховує вплив різних неточно або невизначено описаних параметрів.

Існують різні підходи до прийняття рішень, залежно від того які поняття вважають за основні при аналізі процесу прийняття рішень.

В рамках теоретико-ігрової концепції прийняття рішень є вибором найбільш переважної альтернативи з множини наявних альтернатив. Тоді процес прийняття рішень описується парою (Ω, C), де Ω – множина можливих альтернатив, C – принцип оптимальності.

При статистичному підході ситуація прийняття рішень описується трійкою (Ф,Ѳ,F) де Ф – множина можливих рішень особи, що приймає рішення, Ѳ – множина станів середовища, F – оціночний функціонал.

Всі функціональні залежності що визначають модель (1.1.1) не визначені чітко. Кількісні параметри цієї моделі можуть з допомогою технологій розділу 1 - визначені лише методами узгодження рішень або експертних оцінок.

Нижче наведемо еквівалентну модель що суттєво полегшує визначення функціональних залежності або навіть ієрархій. Задача оновлення та придбання програмного забезпечення, що має необхідну функціональність та дозволяє спростити визначення параметрів може бути сформульованою подібно задачі 1. Необхідно визначити n видів програмного та технічного комплексів з визначеними стандартними функціями, Необхідно визначити як пакети відрізняються вартістю і якісними показниками.

Повинні бути сформульовані множини

$$G_1, G_2, G_3 \ldots G_i$$ -

Це множини програм, які визначають тип програмного та технічного забезпечення *i*.

$$G_1, G_2, G_3 \ldots G_i$$

$$G_i = \{1, 2 \ldots n_i\}$$

. Для кожного типу програмного та технічного забезпечення необхідно визначити такий номер m_i для якого досягається максимальне значення оцінок якості:

$$\max \sum_{i=1}^{n} f_i(m_i)$$

при виконанні умови з обмеженням витрат

$$\sum_{i=1}^{n} S_G(m_i) \le v$$

v – максимальні можливі витрати, що пов’язані з купівлею та налагодженням програмного та технічного забезпечення, $fi(m_i)$ - функція якості, кожного програмного продукту з множини

$$G_i = \{1,2...n_i\}$$

Це формулювання задачі будемо називати Задачею 2.

Функції $fi(m_i)$ можуть бути визначені методом аналізу ієрархій. Метод аналізу ієрархій (MAI) - це математичний метод прийняття рішень, розроблений вченим Томасом Сааті. Він дозволяє упорядковувати та порівнювати альтернативи вибору на основі їх важливості та відносної переваги відносно один одного.

MAI використовується для розв'язання складних завдань прийняття рішень, коли необхідно враховувати багато критеріїв та альтернатив. Метод передбачає побудову ієрархії критеріїв та альтернатив, визначення їх вагомості та відносної важливості, а потім застосування математичних операцій для визначення оптимального рішення.

Основні кроки методу аналізу ієрархій включають ілюструє наступний малюнок.

Рисунок 2.1.1. Побудова ієрархії критеріїв та альтернатив.

Наступі етапи методу аналізу ієрархій.

Визначення відносної важливості кожного критерію відносно іншого.

Порівняння альтернатив за кожним критерієм та визначення їх відносної переваги.

Розрахунок загальних балів для кожної альтернативи та вибір оптимального варіанту.

Метод аналізу ієрархій дозволяє систематизувати та формалізувати процес прийняття рішень, що робить його корисним для багатьох галузей, включаючи управління проектами, виробництво, фінанси та багато інших.

Реалізуючи алгоритм аналітичного ієрархічного процесу (AHP) спеціалісти порівнюють важливість критеріїв, по два за один раз, через попарне порівняння. Наприклад, чи більше ви дбаєте про переваги швидкості реакції чи функціональності, і наскільки більше? AHP перетворює ці оцінки в числа, які можна порівняти з усіма можливими критеріями. Ця кількісна здатність відрізняє AHP від інших методів прийняття рішень.

МАІ використовується для розв'язання складних завдань прийняття рішень, коли необхідно враховувати багато критеріїв та альтернатив. Метод передбачає побудову ієрархії критеріїв та альтернатив, визначення їх вагомості та відносної важливості, а потім застосування математичних операцій для визначення оптимального рішення.

Основні кроки методу аналізу ієрархій включають:

Побудова ієрархії критеріїв та альтернатив.

Визначення відносної важливості кожного критерію відносно іншого.

Порівняння альтернатив за кожним критерієм та визначення їх відносної переваги.

Розрахунок загальних балів для кожної альтернативи та вибір оптимального варіанту.

АІП складається з чотирьох основних кроків:

1. Створення ієрархії проблеми, яка включає ціль, критерії, під критерії та альтернативи.

2. Оцінка відносної важливості критеріїв та під критеріїв за допомогою попарних порівнянь.

3. Оцінка відносної привабливості альтернатив за кожним критерієм та під критерієм за допомогою попарних порівнянь.

4. Обчислення загального пріоритету кожної альтернативи та вибір найкращої.

АІП використовує матриці попарних порівнянь, які відображають суб’єктивні оцінки експертів або приймаючих рішення. Для перевірки узгодженості оцінок використовується індекс узгодженості (ІУ), який має бути меншим за 0,1 [3].

Яскраво виражений для більшості користувачів індивідуальний характер використання та оцінки програмних продуктів (адресність) —дуже важлива характеристика програм. Кожний користувач має своє індивідуальне

уявлення про предметну область. Інформація з оцінки програмних продуктів має для нього виключно «індивідуальну цінність». Отже, завдяки адресності та старінню (втраті актуальності з часом) задовольнити потреби в ІПП один раз і назавжди неможливо.

Порівняльна оцінка програмних продуктів дуже ускладнена через відсутність для користувача інструментарію для вибору того або іншого продукту, який задовольняє його запити. Необхідно розробити спеціальні методи оцінювання програмних продуктів.

Ось приклад матриці попарних порівнянь для трьох критеріїв A (функціональні вимоги), B (характеристики якості) та C (вимоги до документації):

	A	**B**	**C**
A	1	3	5
B	1/3	1	2
C	1/5	1/2	1

Таблиця 2.1.1. Таблиця ранжування параметрів якості програмних продуктів

Ця матриця означає, що критерій A важливіший за критерій B в три рази, а за критерій C - в п’ять разів. Аналогічно, критерій B важливіший за критерій C в два рази.

Для обчислення пріоритетів критеріїв можна використати метод власного вектору, який полягає в знаходженні нормалізованого вектору, що відповідає найбільшому власному значенню матриці. Наприклад, для даної матриці власне значення дорівнює 3,03, а власний вектор дорівнює (0,74; 0,19; 0,07). Це означає, що пріоритети критеріїв A, B та C становлять 74%, 19% та 7% відповідно.

Наведений метод дозволяє упорядкувати основні вимоги до програмного забезпечення, що формулюють такими пунктами:

• функціональні вимоги (п1);

• характеристики якості(п2);;

• вимоги до документації(п3);;

• вимоги по розвитку програмного забезпечення, обслуговуванню і навчанню(п4);;

• вимоги по цінах(п5);

2.2. **Реалізація методів дослідження**

Враховуючі результати розділу 2.1 функції $fi(m_i)$ критерію

$$\max \sum_{i=1}^{n} f_i(m_i)$$

можуть бути визначені методом аналізу ієрархій а Задача 2 досліджена з допомогою рекурентних співвідношень динамічного програмування.

Визначимо допоміжні функції

$F_k(y),\ k<=n,\ y<=b$

$F_0(y)=max\ f_1(m_1) \mid \quad V_1(m_1) <=y$

$$F_k(y)=\max_{J=1,..к} \sum f_j(m_j) \mid \quad \sum_{J=1,..к} V_j(m_j) <=y$$

З принципу оптимальності витікає наступне рекурентне співвідношення

$$F_k(y)=\max_{m_к} (f_к(m_к) + F_{k-1}(y- V_k(m_k)))$$

Це співвідношення призначено послідовно будувати множину функцій F_k. Якщо ці функції побудовані, то також з рекурентних співвідношень будується вектор m. Текст програми для оптимізації Задачі 2 наведено нижче

```
var table = document.getElementById("tbl");  //таблиця

(function () {
    document.body.onload = function (e) {
        var t = screen.width * 70 / 100 + "px";
        document.getElementById("inputs").style.width = t;
        var calc1 = document.getElementById("calcFirst");
        var calc2 = document.getElementById("calcSecond");
        //calc.style = "float:right;margin-right:5%;";
        calc1.onclick = execute1;
        calc2.onclick = execute2;
        table.style.width = t;
console.log('cmd');
    };
})();

var defaultStyle = table.style;
```

Рисунок 2.2.1. Лістинг програми оптимального вибору обладнання (частина 1)

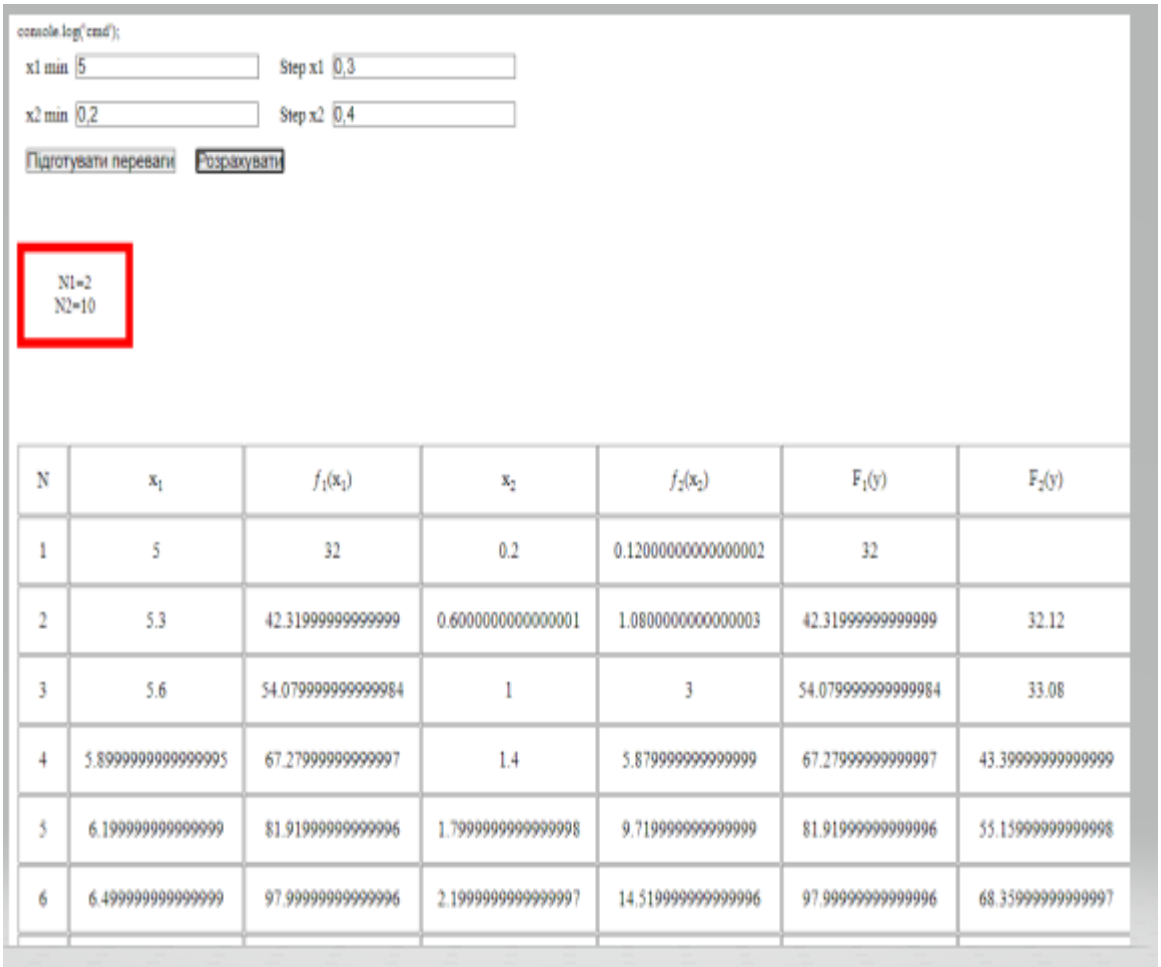
```
var table = document.getElementById("tbl");   //таблиця

(function () {
    document.body.onload = function (e) {
        var t = screen.width * 70 / 100 + "px";
        document.getElementById("inputs").style.width = t;
        var calc1 = document.getElementById("calcFirst");
        var calc2 = document.getElementById("calcSecond");
        //calc.style = "float:right;margin-right:5%;";
        calc1.onclick = execute1;
        calc2.onclick = execute2;
        table.style.width = t;
console.log('cmd');
    };
})();

var defaultStyle = table.style;

const count = 12,
    infinite = -99999999,
    colCount = 7, //кількість стовпчиків у таблиці
k1 = 8,
    k2 = 3,
    k3 = 0,
    k4 = 0,
    k5 = 3,
    k6 = 0;

var rowIndex;

var headers = ["N",
    "x<sub>1</sub>",
    "&#402;<sub>1</sub>(x<sub>1</sub>)",
    "x<sub>2</sub>",
    "&#402;<sub>2</sub>(x<sub>2</sub>)",
    "F<sub>1</sub>(y)",
    "F<sub>2</sub>(y)"];//заголовки для таблиці
function getInputs() {
    var res = {};

    var inpts = document.getElementsByClassName("inpt");
    for (var i = 0; i < inpts.length; i++)
        res[inpts[i].id] = inpts[i].value * 1;

    return res;
}

function GetF1X1(x) {
    var temp = (x - k2);
    var sin = Math.sin(temp);
    return k1 * Math.pow(temp, 2) + k3 * temp * sin + k4 *
Math.pow(sin, 2);
}

function GetF2X2(x) {
    return k5 * Math.pow((x - k6), 2);
}
                                                                }
```

```
function createEmptyCells(row) {
    for (var i = 0; i < colCount; i++) {
        var cell = document.createElement("td");
        cell.id = (rowIndex - 1) + "_" + i;
        row.appendChild(cell);
    }
}

function GetValue(col, row) {
    return document.getElementById(row + "_" + col).textContent * 1;
}

function SetValue(col, row, value) {
    document.getElementById(row + "_" + col).textContent = value;
}

function createRow(index, values) {



        var row = document.createElement("tr"); //пустая строка
        if (index != undefined || index != null) row.id = index;
        if (values != undefined || values != null) {
            var length = 0;
            if (values.length > colCount) length = colCount;   //если
    количество значений больше чем количество столбцов
            else length = values.length;
            for (var i = 0; i < length; i++) {
                var cell = document.createElement("td");
                cell.innerHTML = values[i];
                row.appendChild(cell);
            }
        } else createEmptyCells(row);
        tbl.appendChild(row);
    }

    function Fill_B_F1() {
        for (var N = 0; N < count; N++) {
            SetValue(5, N, infinite);
            for (var i = 0; i <= N; i++) {
                if (GetValue(5, N) < GetValue(2, i)) {
                    SetValue(5, N, GetValue(2, i));
                }
                else continue;
            }
        }
    }

    function Fill_B_F2() {
        document.getElementById("N2_res").style = "display:block;";
        var op, or = 0, max = 0.0;
        for (var N = 1; N < count - 1; N++) {
            op = N; or = 0;
            SetValue(6, 0, infinite);
            do {
                do {
                    if (GetValue(4, op) + GetValue(5, or) > max) {
                        max = GetValue(4, op) + GetValue(5, or);
                        document.getElementById("N1_val").textContent = op
    + 1;
                        document.getElementById("N2_val").textContent = or
    + 1;
                    }
                    or++;
                } while (op + or <= N);
                op--;
            } while (op >= 1);
            SetValue(6, N + 1, max);
            SetValue(6, 1, GetValue(4, 0) + GetValue(5, 0));
            SetValue(6, 0, "");
        }
    }

    function createHeaders() {
        createRow(null, headers);
    }
```

```
function addRow() {
    rowIndex++;
    createRow(rowIndex);
}

function clearTable() {   //очистка таблицы
    rowIndex = 0;   //индекс в таблице

    table.innerHTML = "";
}
function execute1() {
    var v = getInputs(); //данные пользователя

    clearTable();

    createHeaders();

    for (var i = 0; i < count; i++) {

            if (i === 0) {
                addRow();
                SetValue(0, i, rowIndex);
                SetValue(1, i, v.minX1);
                SetValue(2, i, GetF1X1(v.minX1));//f(X1)
                SetValue(3, i, v.minX2);
                SetValue(4, i, GetF2X2(v.minX2));//f(X2)
            }
            else {
                addRow();
                SetValue(0, i, rowIndex);
                SetValue(1, i, v.minX1 += v.step_X1);
                SetValue(2, i, GetF1X1(v.minX1));//f(X1)
                SetValue(3, i, v.minX2 += v.step_X2);
                SetValue(4, i, GetF2X2(v.minX2));//f(X2)
            }
        }
    }

    function execute2() {
        Fill_B_F1();
        Fill_B_F2();
    }
```

Рисунок 2.2.2. Лістинг програми оптимального вибору обладнання

В наведеній нижче формі відображено результат роботи програми оптимального вибору обладнання.



N	x_1	$f_1(x_1)$	x_2	$f_2(x_2)$	$F_1(y)$	$F_2(y)$
1	5	32	0.2	0.12000000000000002	32	
2	5.3	42.31999999999999	0.6000000000000001	1.0800000000000003	42.31999999999999	32.12
3	5.6	54.079999999999984	1	3	54.079999999999984	33.08
4	5.899999999999995	67.27999999999997	1.4	5.879999999999999	67.27999999999997	43.39999999999999
5	6.199999999999999	81.91999999999996	1.7999999999999998	9.719999999999999	81.91999999999996	55.15999999999998
6	6.499999999999999	97.99999999999996	2.1999999999999997	14.519999999999996	97.99999999999996	68.35999999999997

Рисунок 2.1.3. Результат роботи програми оптимального вибору обладнання

Алгоритм дозволяє послідовно будувати множину функцій F_k., що наведені в таблицях. Це надає додаткові можливості для аналізу.

2.2. **Інформаційні моделі задачі**

Базова компонента технології прийняття рішень по оновленню програмних засобів узгодження рішень для оновлення програмного забезпечення є комплекс програм інформаційної підтримки.

Основою інформаційної системи є модель даних — фіксована система понять і правил для представлення структури даних, реального стану і динаміки проблемної області в базі даних

Для побудови інформаційних моделей комплексу програмних засобів узгодження рішень по оновленню програмного забезпечення використано 4 рівня моделювання.

1. Рівень документів вхідних та результативних форм для формування моделей та даних та аналізу результатів. Наприклад форма для задання експертних рішень і звіт

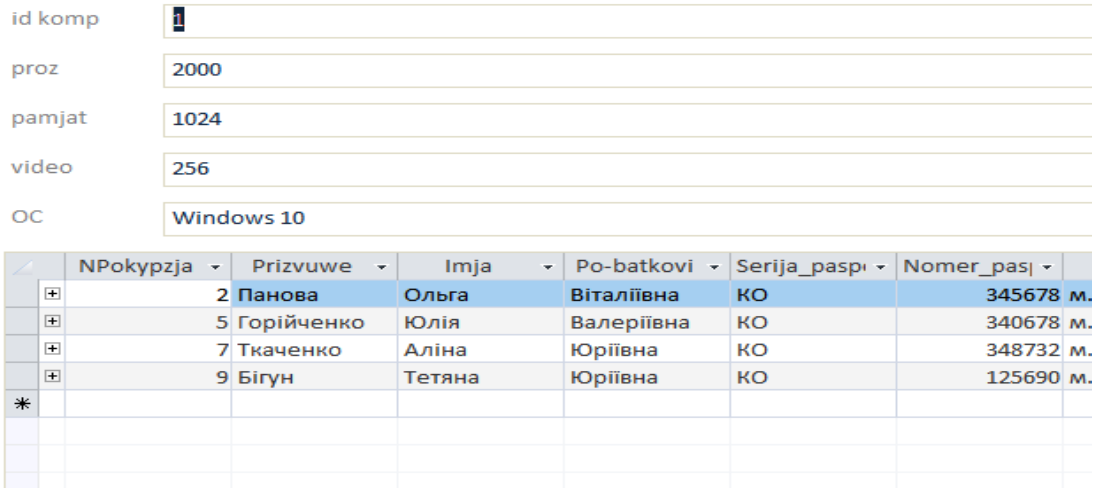


Рисунок 2.2.1. Форма визначення експертних оцінок(частина)

Рисунок 2.2.2. Форма визначення експертних оцінок(частина)

2. Концептуальна - реляційна модель, що зберігає інформацію та підтримується ядром обраної бази даних.

3. Зовнішня або програмна модель сутностей даних , що керується доменами та базується на об’єктних множинах.

4. Модель прийняття узгоджених рішень.

Моделі зберігання даних - це фіксовані системи понять і правил для представлення даних структури, стану і динаміки проблемної області в базі даних. У різний час послідовне застосування одержували ієрархічна, мережева і реляційна моделі даних. У наш час усе більшого поширення набуває об’єктно-орієнтований підхід до організації баз даних.

Ієрархічна модель даних - прикладом є інтернет. Модель будується у вигляді ієрархічної деревоподібної структури, у якій для кожного головного об’єкта (сторінки) існує кілька підлеглих (посилання), а для кожного підлеглого об’єкта може бути тільки один головний. Мережева модель даних (прикладом є інтернет) будується у вигляді графа, в якому вузлами є записи (сторінки), а дугами - зв’язки між записами (посилання).

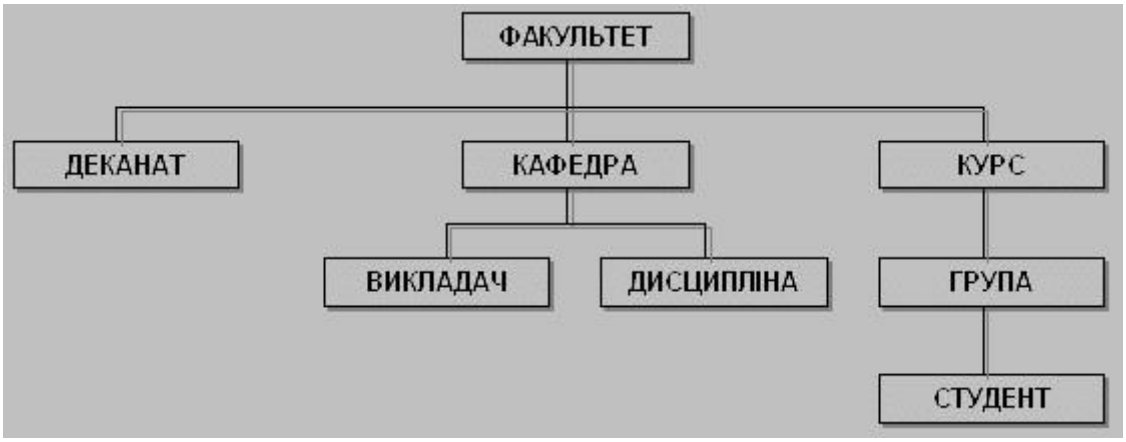


Рисунок 2.2.3 Схема відношень між об'єктами в ієрархічній базі даних

Реляційна модель даних базується на теорії множин і логіки першого порядку. Вона використовує таблиці для зберігання інформації, де кожен рядок таблиці відповідає запису, а кожний стовпець – атрибуту.

Проектування бази даних — це сукупність процесів, які полегшують проектування, розробку, впровадження та підтримку корпоративних систем керування даними. Правильно розроблена база даних проста в обслуговуванні, покращує узгодженість даних і економічно ефективна з точки зору дискового простору. Розробник бази даних вирішує, як співвідносяться елементи даних і які дані потрібно зберігати.

Основними цілями проектування бази даних у СУБД є створення логічних і фізичних моделей проектування запропонованої системи баз даних.

Логічна модель зосереджується на вимогах до даних і даних, які мають зберігатися незалежно від фізичних факторів. Його не цікавить те, як зберігатимуться дані або де вони фізично зберігатимуться.

Модель проектування фізичних даних передбачає переклад логічної конструкції БД бази даних на фізичні носії з використанням апаратних ресурсів і систем програмного забезпечення, таких як системи керування базами даних (СУБД).

Модель зв’язку сутностей (ER Modeling) — це графічний підхід до проектування бази даних. Це модель даних високого рівня, яка визначає елементи даних та їхній зв’язок для певної програмної системи. Модель ER використовується для представлення об’єктів реального світу.

Сутність - це річ або об'єкт у реальному світі, який можна відрізнити від навколишнього середовища. Наприклад, кожен працівник (експерт) організації є окремою сутністю. Нижче наведено деякі основні характеристики сутностей.

Сутність має набір властивостей.

Властивості сутності можуть мати значення.

Кожен атрибут може мати значення. У більшості випадків один атрибут має одне значення. Але атрибути також можуть мати кілька значень . Наприклад, «номер експерту» має єдине значення. Але його властивість «телефонні номери» може мати кілька значень.

Суб'єкти можуть мати відносини один з одним. Розглянемо найпростіший приклад. Припустимо, що кожному програмісту дається комп’ютер. Зрозуміло, що комп’ютер також є сутністю. Програміст використовує цей комп’ютер, і той самий комп’ютер використовує програміст. Іншими словами, між програмістом і його комп’ютером існує взаємозв’язок.

Entity Relationship Modeling моделює сутності, їхні атрибути та зв’язки між сутностями.

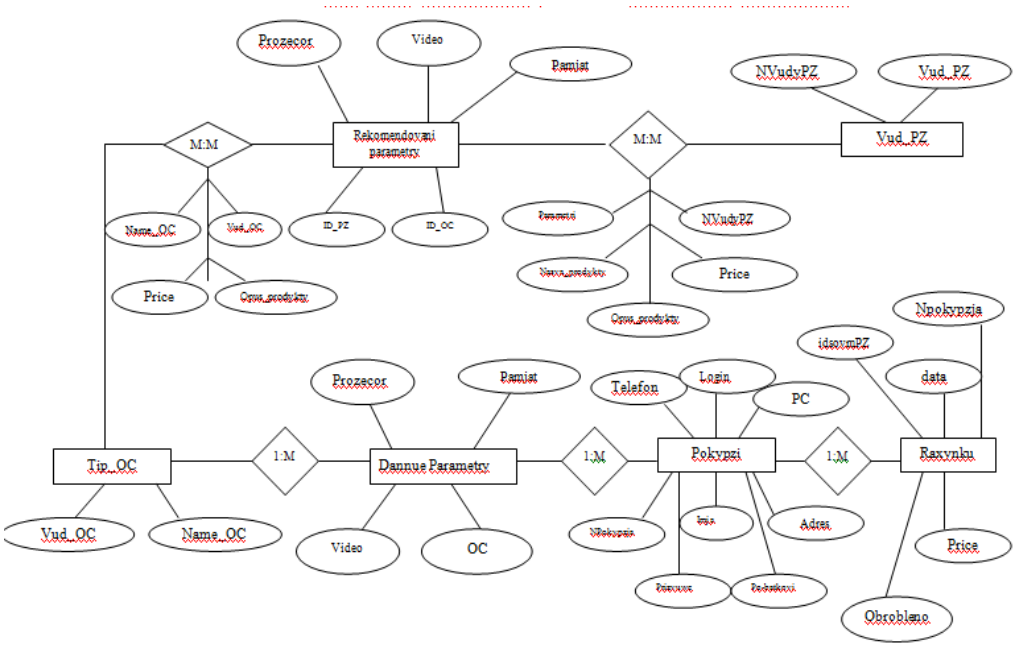


Рисунок 2.2.4. ЕР діаграма концептуальної бази даних.

Схема об’єктів. Цю схему часто використовують для повторної перевірки схеми класу на точність. Іншими словами, вона допомагає перевірити, чи працюватиме схема класу. Вона відображає об’єкти системи та їхні зв’язки, а також допомагає визначити потенційні недоліки структури, які слід усунути.

Концептуальна модель, описана попереднім малюнком, може бути перетвореною на реляційну на підставі наступних простих принципів.

На першому етапі виділяємо об’єкти - сутності сильного типу (або сутності, що існують незалежно від інших об’єктів та у зв'язках беруть участь лише на стороні одиниці) та підпорядковані об'єктні множини - слабкого (або сутності, не можуть існувати незалежно від інших).

Об'єктні множини сильного типу (або сутності, що існують незалежно від інших) визначають таблиці (перетворюються на таблиці) з такими ж іменами та полями, що відповідають атрибутам множини. У нашому випадку це множини "експертів", "програмних засобів" і "типів".

Об'єктні множини - слабкого типу перетворюються на таблиці з такими ж іменами. Поля таблиць відповідають лише тим атрибутам множини, які є

характерними для конкретизації та не властиві узагальнення. Кожна множина отримує зовнішній ключ - посилання на первинний ключ головної таблиці. (Таблиці "експертів" та "програмних засобів")

Складові об'єктні множини - стають таблицею перетинів, і таблиці множин, що перетинаються, поповнюються новими атрибутами - якщо немає атрибутів характерних саме для перетину (деталі виробу, операції - деталі).

відношення перетворяться так:

- таблиці успадковують відносини відповідних об'єктних множин;

- відношення друг до друга зазвичай перетворюється шляхом злиття двох таблиць у один рядок, якого містять атрибути обох об'єктів;

- для кожного відношення один до багатьох в таблицю об'єкта, для якого потужність відношення дорівнює "багато" включається стовпець, є зовнішнім ключем, що вказує на інший об'єкт для якого потужність відношення дорівнює 1;

- Для створення відношення багато хто до багатьох створюється таблиця перерізу.

- рекурсивні відносини призводять до появи рекурсивних зовнішніх ключів, і можливо нових полів, якщо потрібна цілісність даних

Відповідно до цієї технології зв'язок між замовленнями та товарами змінюється таблицею перетину з необхідними атрибутами. Реляційна схема даних набуває вигляду.

Побудована відповідно до формальних механізмів перетворення ЕР діаграми схема бази даних, що відповідає EDM моделі наведена нижче.

.

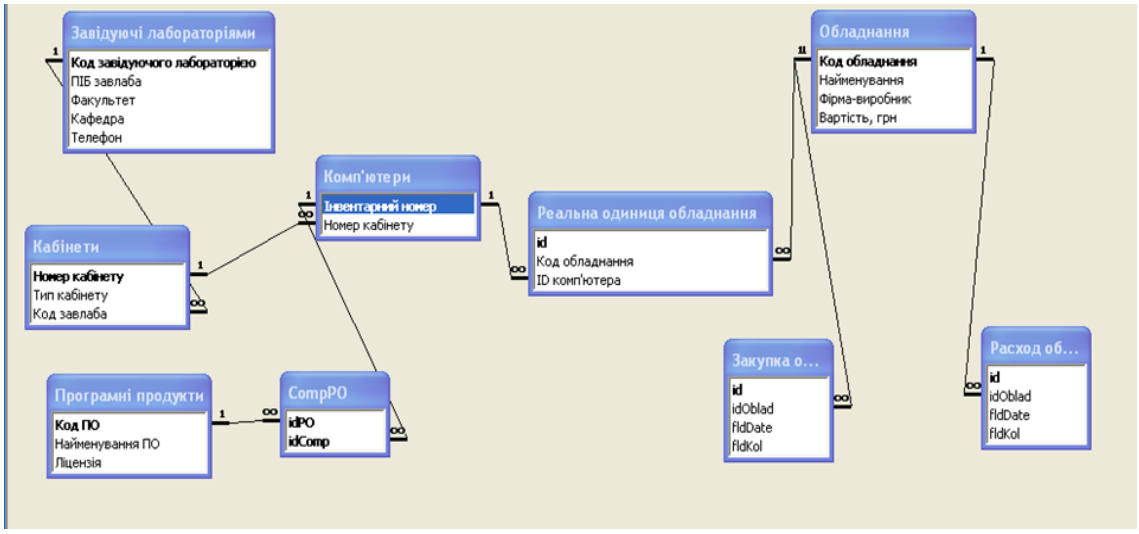


Рисунок 2.2.5. Схема реляційної бази даних.

Кожний зв’язок між таблицями відповідає необхідному правилу логічної цілісності, що базується на посиланнях. З метою спрощення технології редагування даних застосовуються механізми підтримки логічної цілісності даних – каскадне оновлення та каскадне видалення

Спроектована база даних забезпечує цілісність даних.

Цілісність даних означає точність, повноту та узгодженість даних у вашій базі даних.

Цілісність даних включає три конкретні технічні аспекти структури реляційної бази даних:

Цілісність сутності (або цілісність на рівні таблиці) гарантує, що таблиця не має повторюваних записів і що всі значення первинних ключів таблиці є унікальними, а не нульовими.

Цілісність домену (або цілісність на рівні поля) гарантує, що мета кожного поля є чіткою та ідентифікованою, а значення в кожному полі дійсні, послідовні та точні.

Посилальна цілісність (або цілісність на рівні зв’язку) гарантує надійність зв’язків між парами таблиць, тож записи в таблицях синхронізуються щоразу, коли дані вводяться, оновлюються чи видаляються з будь-якої таблиці.

Добре спроектована база даних підтримує цілісність даних шляхом впровадження процесів і стандартів, запропонованих на етапі проектування

2.3. Висновки до розділу

Задача оновлення та придбання програмного забезпечення базується на нечіткій інформації про якість та функціональність програм але повинна гарантувати (або бажано) покращувати параметри обробки інформації при обмежених витратах.

Аналіз задачі про вибір програмних та технічних засобів вимагають від керівника будь-якого рангу уміння швидко і правильно приймати рішення. При цьому необхідно враховувати вимоги зростання обсягів обробки інформації, подальший розвиток техніки, підвищення вимог до якості обробки інформації та інші фактори.

У цих умовах рішення, що приймаються на основі особистого досвіду і інженерної інтуїції, можуть виявитися малоефективними, оскільки не враховують цілого ряду протидіючих умов. Така ситуація вимагає застосування технології прийняття рішень, яка базується на кількісних оцінках варіантів, і виключає або зменшує значення суб'єктивних факторів і при цьому враховує вплив різних неточно або невизначено описаних параметрів.

Розглянуту дві моделі формування оптимальних рішень задачі про вибір програмних та технічних засобів. Обидві моделі базується на нечіткій інформації про якість та функціональність програм. Чітких уявлень не існує. Обрана модель що надає зручні (порівняно) умови для визначення якісних параметрів.

Для кожного типу програмного та технічного забезпечення необхідно визначити такий номер m_i для якого досягається максимальне значення оцінок якості:

$$\max \sum_{i=1}^{n} f_i(m_i)$$

при виконанні умови з обмеженням витрат

$$\sum_{i=1}^{n} S_G(m_i) \le v$$

Враховуючі результати розділу 2.1 функції $fi(m_i)$ критерію

$$\max \sum_{i=1}^{n} f_i(m_i)$$

можуть бути визначені методом аналізу ієрархій а Задача 2 досліджена з допомогою методу Беллмана рекурентних співвідношень динамічного програмування. Текст програми наведений в розділі 2.1.

Функції $fi(m_i)$ можуть бути визначені методом аналізу ієрархій. Метод аналізу ієрархій дозволяє упорядковувати та порівнювати альтернативи вибору на основі їх важливості та відносної переваги відносно один одного.

Метод передбачає побудову ієрархії критеріїв та альтернатив, визначення їх вагомості та відносної важливості, а потім застосування математичних операцій для визначення оптимального рішення. Розглянуті принципи реалізації методу для задачі.

Базою програмного засобу для обґрунтування рішень, що пов’язані із прийняттям рішень для визначення необхідних та оновлення програмних засобів підприємства є інформаційна система програмних засобів, функцій програмних засобів та переваг. Інформація, що визначає інформаційна система є основою для формування моделей прийняття рішень. Параметри моделей є не чіткими та формуються з допомогою методів узгодження рішень [1].

Використано 4 рівня моделювання для побудови інформаційних моделей комплексу програмних засобів узгодження рішень по оновленню програмного забезпечення. Обґрунтоване проектне рішення для кожного рівня моделювання.

РОЗДІЛ 3. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Обґрунтування принципів програмної реалізації

Програмний засіб розроблений інструментми Microsoft Visual Studio.

Microsoft Visual Studio має багато переваг для розробників, які працюють з базами даних. Він має вбудовані інструменти, які допомагають розробникам створювати, змінювати та керувати базами даних.

Зручний інтерфейс: Visual Studio має дружній інтерфейс, який дозволяє розробникам легко створювати та керувати базами даних.

Підтримка багатьох типів баз даних: Visual Studio підтримує багато типів баз даних, включаючи SQL Server, MySQL, Oracle та інші.

Інструменти для моделювання даних: Visual Studio має вбудовані інструменти для моделювання даних, які дозволяють розробникам створювати моделі даних та візуалізувати їх.

Підтримка мов програмування: Visual Studio підтримує багато мов програмування, включаючи C#, Visual Basic та інші.

Підтримка розробки веб-додатків: Visual Studio має вбудовані інструменти для розробки веб-додатків, які дозволяють розробникам створювати веб-сайти та веб-додатки.

Підтримка розробки мобільних додатків: Visual Studio має вбудовані інструменти для розробки мобільних додатків, які дозволяють розробникам створювати додатки для Android, iOS та Windows Phone.

Підтримка роботи з Git: Visual Studio має вбудовану підтримку Git, що дозволяє розробникам легко керувати версіями свого коду.

Підтримка тестування: Visual Studio має вбудовані інструменти для тестування коду, що дозволяє розробникам перевіряти правильність свого коду.

Підтримка роботи з Azure: Visual Studio має вбудовану підтримку Azure, що дозволяє розробникам легко розгортати свої додатки в хмарі.

Підтримка роботи з Docker: Visual Studio має вбудовану підтримку Docker, що дозволяє розробникам легко розгортати свої додатки в контейнерах.

Підтримка роботи з Kubernetes: Visual Studio має вбудовану підтримку Kubernetes, що дозволяє розробникам легко керувати своїми додатками в кластері Kubernetes.

Програмний інтерфейс системи побудований відповідно до принципів діаграми варіантів використання. Діаграма прецедентів (або діаграма варіантів використання — в UML), діаграма, на якій зображено відношення між акторами та прецедентами в системі.[41]

Діаграма прецедентів показує різні варіанти використання та різні типи користувачів системи і часто супроводжується іншими типами діаграм. Варіанти використання представлені колами або еліпсами. Актори (дійові особи) часто зображуються у вигляді паличок.

Для програмного засобу вона має вигляд

Рисунок 3.1.1. Діаграма варіантів використання

Призначення діаграм класів полягає у тому, щоб надати огляд архітектури високого рівня, взаємодії між її частинами та розподілу відповідальносте між класами. Це може бути корисно при проектуванні програмного забезпечення та в розвитку проектів для зрозуміння їхньої структури та взаємодії.

Діаграми класів призначення або діаграми класів використовуються для відображення взаємодії між класами та їхніми взаємозв'язками в системі програмного забезпечення. Діаграми класів призначення спрямовані на відображення того, як система взаємодіє з різними користувацькими або зовнішніми елементами.

Основні елементи діаграм класів призначення включають класи, інтерфейси, зв'язки та компоненти. Тут коротко розглянемо ці елементи:

Класи (Class): Представляють об'єкти або сутності, які взаємодіють в системі. Класи зображуються у верхній частині діаграми та можуть включати методи, атрибути та інші характеристики.

Інтерфейси (Interface): Вказують, як клас взаємодіє з іншими класами чи системами. Їх можна зображати у вигляді стереотипу (наприклад, "<<interface>>").

Зв'язки (Association): Вказують на зв'язки між різними класами. Зв'язки можуть бути одно зв’язковими (наприклад, один до одного) або багато зв’язковими (наприклад, один до багатьох). Вони відображаються лініями, що з'єднує класи.

Компоненти (Component): Представляють окремі частини системи, які можуть бути розглянуті як відокремлені модулі чи підсистеми.

Пакети (Package): Групують класи та компоненти в більші логічні структури. Пакети можуть вказувати на рівні абстракції в системі.

Рисунок 3.1.2 Діаграма класів

Використання діаграм класів має декілька переваг у процесі розробки програмного забезпечення:

Візуалізація структури системи: Діаграми класів надають візуальне відображення структури системи, що дозволяє розробникам та архітекторам легше розуміти, як класи взаємодіють між собою та як вони організовані в системі.

Архітектурна ясність: Діаграми класів допомагають у визначенні архітектурних аспектів системи, таких як розподіл відповідальностей між класами, модульність та інші аспекти, що сприяють чіткій організації коду.

Доцільність взаємодії: З допомогою зв'язків та взаємодій між класами можна краще розуміти, як класи співпрацюють між собою та взаємодіють з іншими елементами системи.

Підтримка прийняття рішень: Діаграми класів можуть бути використані для аналізу та прийняття рішень на ранніх етапах розробки. Вони дозволяють зрозуміти наслідки змін в структурі системи та допомагають у визначенні оптимальних шляхів розвитку.

Підтримка генерації коду: Деякі інтегровані середовища розробки дозволяють автоматично генерувати код на основі діаграм класів, що полегшує та прискорює процес розробки.

Зрозумілість для команди: Діаграми класів є ефективним інструментом для передачі інформації між членами команди розробників, а також для спілкування з іншими учасниками проекту, такими як менеджери чи замовники.

Підтримка документації: Діаграми класів можуть бути включені до технічної документації проекту, що полегшує розуміння системи для нових членів команди або для тих, хто взаємодіє з системою ззовні.

Загалом, діаграми класів є потужним інструментом, який допомагає в розумінні, проектуванні та впровадженні систем програмного забезпечення.

Структура проекту .NET відповідно до Domain Driven Design наступна

```
/App
 /ProjectName.Web.Public
 /ProjectName.Web.Admin
 /ProjectName.Web.SomeOtherStuff
/Domain
 /ProjectName.Domain.Core
 /ProjectName.Domain.BoundedContext1
 /ProjectName.Domain.BoundedContext1.Services
 /ProjectName.Domain.BoundedContext2
 /ProjectName.Domain.BoundedContext2.Command
 /ProjectName.Domain.BoundedContext2.Query
 /ProjectName.Domain.BoundedContext3
/Data
 /ProjectName.Data
/Libs
 /Problem1Resolver
 /Problem2Resolver
```

Рисунок 3.1.3

Структура (Рисунок 3.1.3) відповідає наступним принципам Domain Driven Design. Проекти з папки Libs не залежать від домену. Вони вирішують лише своє локальне завдання, формування звітів, і далі..

3.2. Керівництво користувача

Відповідно діаграми 3.1.1 Існує два типу користувачів – адміністратор і менеджер. Права визначають правила входу до системи.

Рисунок 3.2.1Вхід в систему.

Базовою для адміністраторів є інформація про агентів та їх права, що визначаються належністю до різних груп.

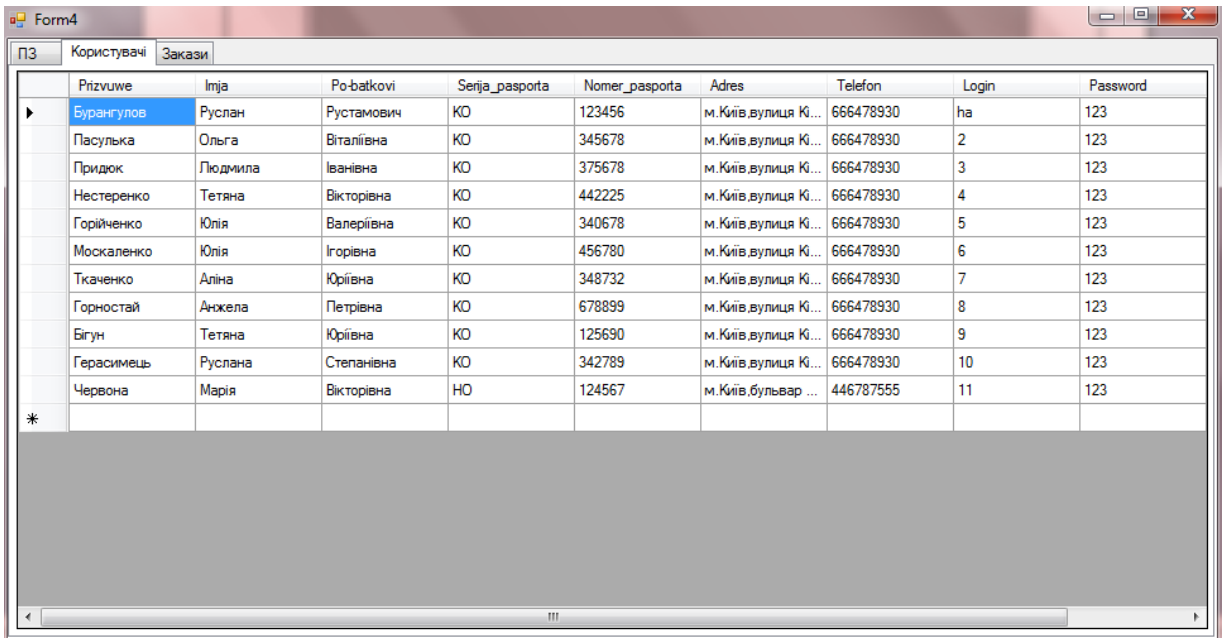


Рисунок 3.2.2 Визначається перелік зареєстрованих агентів.

На формі визначається перелік зареєстрованих агентів і їх можливостей у вигляді спадаючого списку. Для визначення прав потрібно вибрати з списку потрібне прізвище, потім необхідну інформацію.

Наступна форма Рисунок 3.2.2 призначена для експертів – менеджерів.

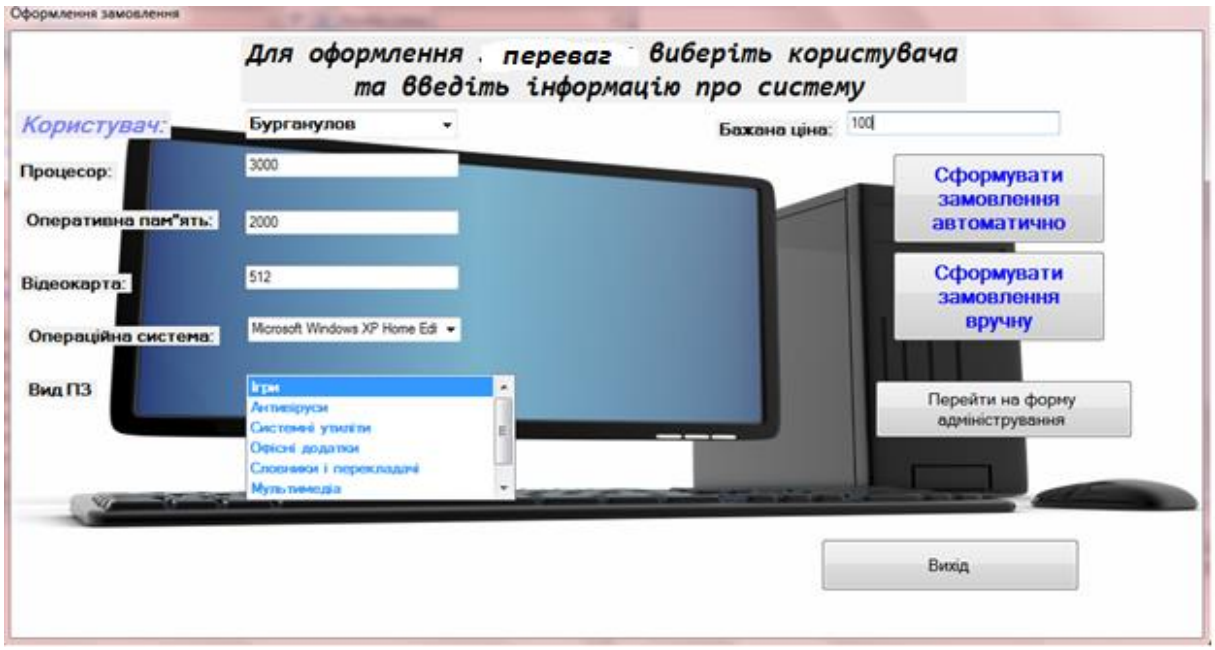


Рисунок 3.2.3 Формування потреб

Наступна форма Рисунок 3.2.3 дозволяє реалізувати механізм бінарних порівнянь.

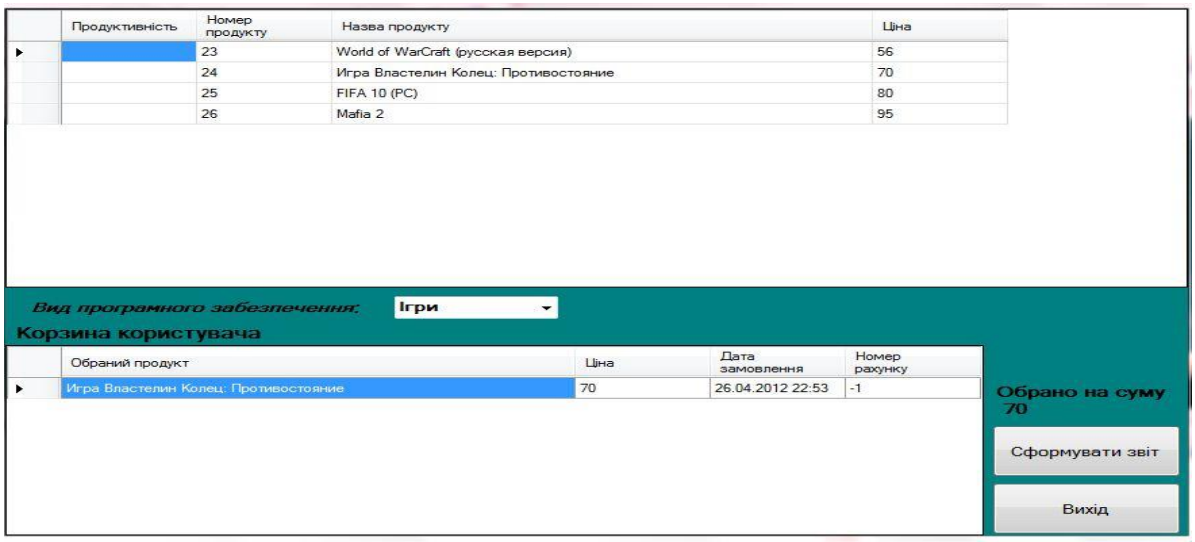


Рисунок 3.2.4

Наступна форма Рисунок 3.2.4 надає можливість реалізувати вибір найбільш узгодженого засобу для всіх уподобань

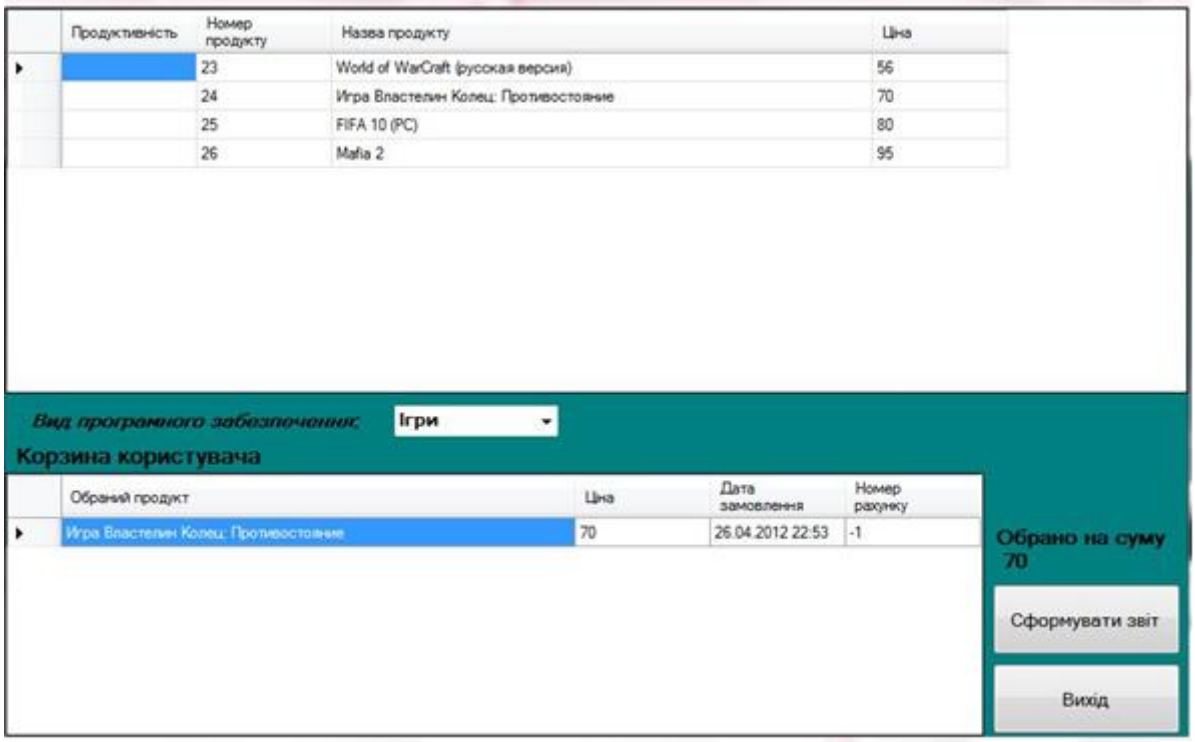


Рисунок 3.2.5 вибір найбільш узгодженого засобу

Уточнений вибір найбільш узгодженого засобу для всіх уподобань надає наступна форма.

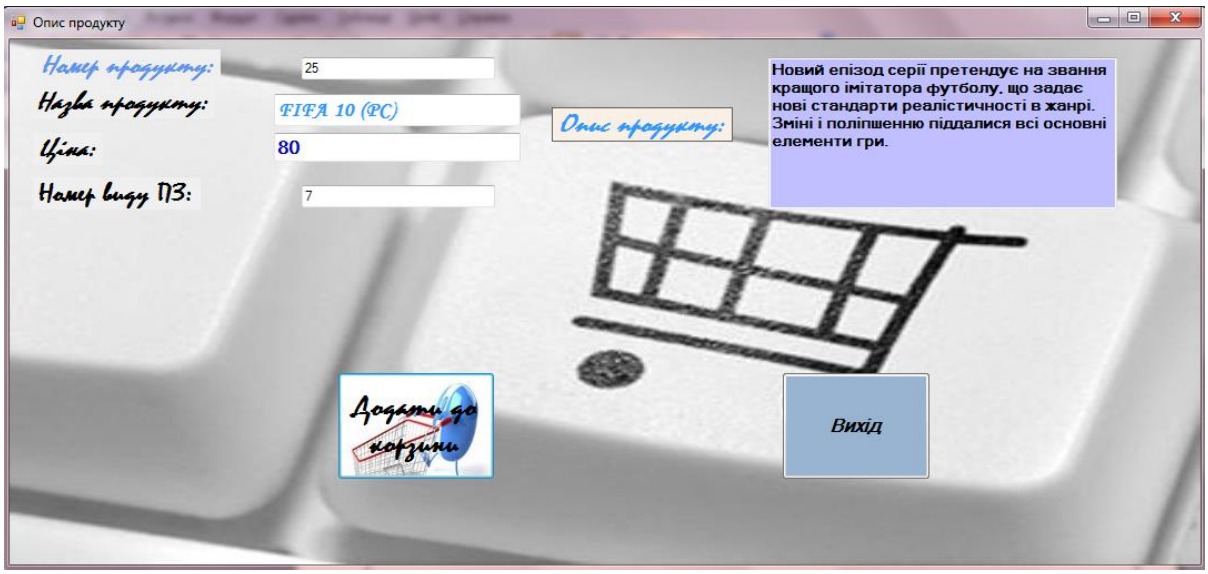


Рисунок 3.2.6 Опис продукту

3.3 Висновки до розділу

Під час розробки інформаційних моделей програми узгодження рішень по оновленню програмного забезпечення було використано 4 рівня програмного моделювання.

1. Рівень документів.
2. Концептуальна модель даних,
3. Модель сутностей даних. Керується доменами і є основою об’єктних множин.
4. Модель формування узгоджених рішень.

Програма реалізована засобами Microsoft Visual Studio.

Висновки

Розглянуті практичні та теоретичні аспекти технології прийняття рішень проблеми оновлення та обслуговування специфічного інформаційного виду обладнання – програмних засобів. Розглянуті теоретичні аспекти та питання практичної реалізації прийняття рішень для контролю за використанням програмних засобів підприємства і розв'язання задач оновлення специфічного виду обладнання – програмних засобів. Інформаційна система для контролю за використанням програмних засобів підприємства є основою для формування моделей прийняття рішень параметри моделей є не чіткими. Інформаційна система містить інформацію для методів узгодження рішень та координації між особами, які приймають рішення оновлення програмних засобів

В роботі наведені принципи формування та функції програмних засобів системи прийняття рішень для реалізації задач оновлення програмного забезпечення на принципах використання нечітких уявлень про оцінки якості та ефективність програмного забезпечення.

СПИСОК ЛІТЕРАТУРИ

1. Щербань В.Ю., Краснитський С. М. Астістова Т. І. , Яхно В. М. Методи представлення, збереження та аналізу даних інформаційних систем –К. “Фастбінд Україна”, 2023, 480 сторінок.

2. Боровик О. Л. Дослідження операцій в економіці : навч. посіб./ О. Л. Боровик, Л. В. Боровик. – Київ : Центр учбової літератури, 2007. –424 с.

3. Боровська Т. М. Основи теорії управління та дослідження операцій :навч. посіб. / Т. М. Боровська, І. С. Колеснік, В. А. Северілов. – Вінниця :УНІВЕРСУМ-Вінниця, 2008. – 242 с.

4. Голіков А. П. Економіко-математичне моделювання світогосподарських процесів : навч. посіб. / А. П. Голіков. – 3-тє вид., перероб. і допов. – Київ : Знання, 2009. – 222 с.

5. Економіко-математичне моделювання : навч. посіб. / Т. С. Клебанова, О. В. Раєвнєва, С. В. Прокопович та ін. – Харків : ВД "ІНЖЕК", 2010. – 352 с.

6. Єгоршин О. О. Математичне програмування : підручник / О. О. Єгоршин, Л. М. Малярець. – Харків : ВД "ІНЖЕК", 2006. – 384 с.

7. Рогоза М. Є. Нелінійні моделі та аналіз складних систем: навч. посіб. в 2 ч. Ч. 1 / М. Є. Рогоза, С. К. Рамазанов, М. К. Мусаєва. – Полтава : РВВ ПУЕТ, 2011. – 300 с.

8. Роман Л. Л. Дослідження операцій. Курс лекцій / Л. Л. Роман. –Львів : Видавництво Тараса Сороки, 2008. – 272 с.

9. Кутковецький В. Я. Дослідження операцій: [навч. посіб.] / В. Я. Кутковецький. – [2-ге видання, виправлене]. – К.: ВД «Професіонал», 2005. – 264 с.

10. Боровик О. В., Боровик Л. В. Дослідження операцій в економіці : навч. посіб. Київ : Центр учбової літератури, 2007. 424 с.

11. Вітлінський В. В., Наконечний С. І., Терещенко Т. О. Математичне програмування : навч.-метод. посіб. для самост.вивч. дисц. Київ : КНЕУ, 2001. 248 с.

12. Вітлінський В. В., Терещенко Т. О., Савіна С. С.Економіко-математичні методи та моделі: оптимізація : навч.посіб. Київ : КНЕУ, 2016. 303 с.
13. Воронков О. О. Оптимізаційні методи і моделі : конспект лекцій з курсу. Харків : ХНУМГ ім. О. М. Бекетова, 2016. 110 с.
14. Дослідження операцій в економіці : підруч. / О. І. Черняк та ін. ; ред. О. І. Черняка. Миколаїв : МНАУ, 2020. 398 с.
15. Дослідження операцій : метод рекоменд. для самост.роботи студентів ден. та заоч. форм навчання напряму підготов. /О. В. Шебаніна та ін. Миколаїв : МНАУ, 2014. 98 с.
16. Дослідження операцій : курс лекцій / О. В. Шебаніна та ін. Миколаїв : МНАУ, 2015. 248 с.
17. Дослідження операцій в економіці : підруч. / І. К Федоренко. та ін. ; за ред. І. К. Федоренка, О. І. Черняка. Київ : Знання, 2007. Київ : Знання, 2017. 558 с.
18. Дослідження операцій. Практичний курс : навч. посіб. / В. Є. Березовський та ін. Умань : Видавець «Сочінський», 2011.238 с.
19. Зайченко О. Ю., Зайченко Ю. П. Дослідження операцій : збірник задач. Київ : Видавничий дім «Слово», 2007. 472 с.
20. Зайченко Ю. П. Дослідження операцій : підруч. Для вузів. 5-е вид., перероб. та допов. Київ : 2001. 688 с.
21. Економіко-математичне моделювання : навч. посіб. / В. В. Вітлінський та ін. ; за заг. ред. В. В. Вітлінського. Київ : КНЕУ, 2008. 536 с.
22. Економіко-математичне моделювання : навч. посіб. / за ред. О. Т. Іващука. Тернопіль : ВПЦ «Економічна думка ТНЕУ», 2008. 704 с.
23. Економіко-математичне моделювання : навч. посіб. / Т. С. Клебанова та ін. Харків : ВД «Інжек», 2012. 352 с.
24. Івченко І. Ю. Математичне програмування : навч. посіб. Київ : Центр учбової літератури, 2007. 232 с.
25. Катренко А. В. Дослідження операцій : підруч. Львів : Магнолія Плюс, 2015. 352 с.

17. Карагодова О. О., Кігель В. Р., Рожок В. Д. Дослідження операцій : навч. посіб. Київ : Центр учбової літератури, 2007. 256 с.

18. Колодінська О. В., Медведєв М. Г. Дослідження операцій : навч. посіб. Київ : Видавництво Європейського університету, 2006. 158 с.

19. Корольов М. Є., Павленко В. І., Савіна О. В., Тимошенко А. Г. Дослідження операцій і методи оптимізації : навч. посіб. Київ : Університет «Україна», 2007. 177 с.

20. Кутковецький В. Я. Дослідження операцій : навч. посіб. Київ : ТОВ “Видавничий дім “Професіонал”, 2004. 350 с.

21. Леснікова І. Ю., Халіпова Н. В. Дослідження операцій у середовищі електронних таблиць Excel : навч. посіб. Київ : Центр учбової літератури, 2007. 186 с.

22. Математичне програмування. Дослідження операцій : навч. посіб. / А. Ф. Барвінський та ін. Львів : «Інтелект-Захід»,

2008. 468 с.

23. Математичне програмування : контр. індивід. завд. Та метод. реком. для сам. роб. студ. / О. В. Шебаніна та ін. Миколаїв : МНАУ, 2015. 80 с.

24. Математичне програмування : метод. реком. з вивч.дисципліни та виконання контрольних робіт здобувачами вищ.освіти / О. В. Шебаніна та ін. Миколаїв : МНАУ, 2020. 132 с.

25. Наконечний С. І., Савіна С. С. Математичне програмування : навч. посіб. Київ : КНЕУ, 2016. 452 с.

26. Охріменко М. Г., Дзюбан І. Ю. Дослідження операцій : навч. посіб. Київ : Центр навчальної літератури, 2006. 184 с.

27. Ржевський С. В., Александрова В. М. Дослідження операцій : підруч. Київ : Академвидав, 2006. 560 с.

28. M. T. Hannan and J. Freeman, “Structural inertia and organizational change,” American Sociological Review, vol. 49, no. 2, pp. 149–164, 1984.

29. A. Basaure, H. Suomi, and H. Hämmäinen, “Transaction vs. switching costs-comparison of three core mechanisms for mobile markets,” Telecommunications Policy, vol. 40, no. 6, pp. 545–566, 2016.

30. T. A. Burnham, J. K. Frels, and V. Mahajan, “Consumer switching costs: a typology, antecedents, and consequences,” Journal of the Academy of Marketing Science, vol. 31, no. 2, pp. 109–126, 2003.

31. N. Agmon, S. Kraus, and G. A. Kaminka, “Multi-robot perimeter patrol in adversarial settings,” in Proeedings of the 2008 IEEE International Conference on Robotics and Automation, pp. 2339–2345, Pasadena, CA, USA, May 2008.

32. J. Le Ny, M. Dahleh, and E. Feron, “Multi-agent task assignment in the bandit framework,” in Proceedings of the 2006 45th IEEE Conference on Decision and Control, pp. 5281–5286, IEEE, San Diego, CA, USA, December 2006.

33. Y.-H. Lyu and D. Balkcom, “Optimal trajectories for kinematic planar rigid bodies with switching costs,” The International Journal of Robotics Research, vol. 35, no. 5, pp. 454–475, 2015.

34. K. I. J. Ho and J. Sum, “Scheduling jobs with multitasking and asymmetric switching costs,” in Proceedings of the 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 2927–2932, Miyazaki, Japan, May 2017.

35. Andon, F. I., Koval, G. I., Korotun, T. M., & Suslov, V. Iu. (2002).
Osnovy inzhenerii kachestva programmnykh sistem, (Sergienko, I.
V. Scientific Ed.). Kyiv: Akademperiodika. 504 p .