МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

ФАКУЛЬТЕТ МЕХАТРОНІКИ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Дипломна магістерська робота

на тему РОЗРОБКА АЛГОРИТМІЧНИХ ТА ПРОГРАМНИХ КОМПОНЕНТІВ СИСТЕМИ КЕРУВАННЯ ОНОВЛЕННЯМ ПРОГРАМНИХ ЗАСОБІВ ПІДПРИЄМСТВА

Виконав: студент групи МгІТ-21
спеціальності
122 комп'ютерні науки
освітньої програми
комп'ютерні науки

Свида Олександр Олександрович

Керівник к.т.н., доц. Володимир Яхно

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

Київ 2022

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

"____"_____2022 року

З А В Д А Н Н Я

НА ДИПЛОМНУ МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Свиде Олександру Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка алгоритмічних та програмних компонентів системи керування оновленням програмних засобів підприємства. Науковий керівник роботи Яхно Володимир. Михайлович, к. т .н., доц.
затверджений наказом вищого навчального закладу від " "жовтня 2022 року № .

2. Строк подання студентом роботи 14.11.2022 р.

3. Вихідні дані до роботи Розробка кафедри інформаційних технологій проектування. Математичні методи дослідження операцій. Задачі керування 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 10.9.2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної магістерської роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	5.10.2922	
2	Розділ 1 (Постановка задачі та методи дослідження)	5.10.2022	
3	Розділ 2 (Обґрунтування моделей та методів)	5.10.2022	
4	Розділ 3 (Алгоритмічне та програмне забезпечення)	10.10.2022	
5	Висновки	25.10.2022	
6	Оформлення дипломної магістерської роботи (чистовий варіант)	30.10.2022	
7	Здача дипломної магістерської роботи на кафедру для рецензування (за 14 днів до захисту)	4.11.2022	
8	Перевірка дипломної магістерської роботи на наявність ознак плагіату (за 10 днів до захисту)	8.11.2022	
9	Подання дипломної магістерської роботи на затвердження завідувачу кафедри (з 7 днів до захисту)	14.11.2022р.	

Студент _____ **Олександр СВИДА**
(підпис)

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

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

АНОТАЦІЯ

Свида О.О. Розробка алгоритмічних та програмних компонентів системи керування оновленням програмних засобів підприємства.
– Рукопис.

Дипломна магістерська робота за спеціальністю 122 комп'ютерні науки та інформаційні технології – Київський національний університет технологій та дизайну, Київ, 2022 рік.

Розглянуті практичні та теоретичні аспекти прийняття рішень для розв'язання задач оновлення та обслуговування специфічного виду обладнання – програмних засобів. Основою 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 є основою для формування моделей прийняття рішень. Параметри моделей є не чіткими та формуються з допомогою методів узгодження рішень.

Розглянуто дві моделі для визначення оптимальної програми планування структури програмних та технічних засобів. Для прийняття рішення прийнята найпростіша модель. Параметри моделі формуються з допомогою методів узгодження рішень. Координація між особами, які приймають рішення в організації, кожна з яких відповідає за певну частину загальної проблеми прийняття рішень для розв'язання задач оновлення та обслуговування програмних засобів., має вирішальне значення для досягнення загальної продуктивності. Серед проблем координації в системах розподіленого прийняття рішень (DDMS) є умови навколишнього середовища, такі як, наприклад, складність рішення-проблеми, яку потрібно вирішити, передбачуваність проблеми та її динаміка формують адаптацію механізмів координації. Проблема координації прийняття рішень по

доцільності використання та оновлення програмного забезпечення є проблемою в умовах невизначеності. В роботі описано принципи побудови та функції системи прийняття рішень для підтримки задач оновлення та обслуговування обладнання (програмного забезпечення) на базі нечітких уявлень про якість та ефективність програмного забезпечення.

Ключові слова: дослідження операцій, керування запасами, моделі даних, автоматизована система

ANNOTATION

Svida O.O. Development of algorithmic and software components of the enterprise software update management system.

- Manuscript.

Master's thesis in the specialty 122 computer science and information technologies - Kyiv National University of Technology and Design, Kyiv, 2022.

Considered practical and theoretical aspects of decision-making for solving the problems of updating and maintaining a specific type of equipment - software. The basis of the software tool for justifying decisions related to decision-making for determining the necessary and updating software tools of the enterprise is the information system of software tools, functions of software tools and benefits. Information is the basis for forming decision-making models. The parameters of the models are not clear and are formed with the help of decision matching methods.

Two models are considered for determining the optimal program for planning the structure of software and technical means. The simplest model is used to make a decision. The parameters of the model are formed with the help of decision matching methods. Coordination among decision makers in an organization, each responsible for a specific part of the overall decision-making problem for software update and maintenance tasks, is critical to overall performance.. The solution matching technology is used to solve the problem. Coordination among decision makers in an organization, each responsible for a specific part of the overall decision-making problem for software update and maintenance tasks, is critical to overall performance. Among the problems of coordination in distributed decision-making systems (DDMS) are environmental conditions, such as, for example, the complexity of the solution-problem to be solved, the predictability of the problem and its dynamics shape the adaptation of

coordination mechanisms. The problem of coordination of decision-making on the expediency of using and updating software is a problem in conditions of uncertainty. The work describes the principles of construction and functions of the decision-making system to support the tasks of updating and maintaining equipment (software) based on vague ideas about the quality and efficiency of the software.

Keywords: operations research, inventory management, data models, automated system

Зміст

Вступ…….…… 10

Розділ 1. Задача та методи дослідження……..... 14

1.1. Формулювання проблеми оновлення програмного забезпечення……14

1.2. Існуючі технології дослідження ……...17

1.3. Висновки до розділу…… 20

Розділ 2. Обґрунтування моделей та методів……....22

2.1. Формальні моделі задачі…… ……..25

2.2. Алгоритми узгодження…….30

2.3 Висновки до розділу ……..……..47

Розділ 3. Програмне забезпечення……....38

3.1. Обґрунтування принципів програмної реалізації……..41

3.2. Керівництво користувача ……...45

3.3. Висновки до розділу ……..49

Висновки……..50

Література…….. 51

Додатки…… 55

Вступ

Актуальність теми. У сучасному конкурентному середовищі особливого значення набуває проблема підвищення ефективності менеджменту (в широкому сенсі – проблема обробки інформації), що потребує розроблення та удосконалення системи аналітичного забезпечення [7; 11]. Найбільш важливою частиною цього розв'язання є програмне та комп'ютерне забезпечення підприємства. Визначення необхідного та планування оновлення програмного та комп'ютерного забезпечення підприємства завжди виконується в умовах неузгодженості думок на важливе питання – чи потрібен новий комп'ютер або новий програмний засіб. Наприклад… В навчальному закладі вивчають програмування з допомогою Visual studio 10, 15, 17, 19, 22. Ці засоби дозволяють проілюструвати всі необхідні технології, чи ні. Комп'ютери різні. Найшвидше виконує необхідні операції (реагує) Visual studio 10. Найкращій (здається) засіб Visual studio 22 у якого на більшості комп'ютерів час реагування дуже значний (можливо неприпустимо). Ситуація з якістю і вартістю цих засобів для навчання невизначена. Планування оновлення та придбання нових програмних та технічних засобів базуватися на реальних економічних можливостях і умовах змін функціонування комп'ютерного та програмного обладнання.

Дослідити можливість використання нових стратегій керування, що базуються на принципах, що надає поле теорії прийняття рішень [11] яке визначає інструменти для раціонального прийняття рішень мета роботи. Також актуальним є обґрунтувати ефективність методів, що базуються на принципах, які надає поле теорії прийняття рішень, для планування визначення програмних засобів та запропонувати програму, що робить вибір програмних засобів наочним.

Мета дослідження. Мета дослідження – програмний засіб для розробка рекомендацій прийняття рішень по визначенню та оновленню

комп’ютерного та програмного забезпечення, засіб, що дозволяє дослідити, і що дуже важливо обґрунтувати, принципи узгодженого формування основних проектних рішень. Важливим компонентом є програмні засоби, що повинні реалізувати інформаційну підтримку - важливі задачі пошуку і оновлення інформації.

Для досягнення поставленої мети в роботі необхідно вирішити наступні проблеми:

- Дослідити сучасні стратегії для проблем прийняття рішень та сформулювати принципи узгодження рішень достатньо адекватні узгодженню рішень для оновлення програмного забезпечення. Розробити моделі що є необхідними для збереження, відображення, редагування даних;
- реалізувати принципи процесів та правил домену Domain-driven design для відображення інформаційних моделей задач застосування та оновлення програмного забезпечення;
- для реалізації комплексу задач узгодження рішень обрати середовища програмування, що реалізують найбільш ефективні та зручні технології, а це вимагає виконання дослідження та порівняльного аналізу засобів розробки та видів програмної архітектури;
- Обґрунтувати вибір та використати інструментальні та апаратні засоби програмування з локалізацією до української мови.

Завдання дослідження. Реалізувати підхід до координації між особами, які приймають рішення в організації, кожна з яких відповідає за певну частину загальної проблеми прийняття рішень. Дослідити принципи, що мають вирішальне значення для досягнення загальної продуктивності. Методи координації в системах розподіленого прийняття рішень (DDMS) де складність рішення-проблеми, яку потрібно вирішити, передбачуваність

проблеми та її динаміка формують адаптацію механізмів координації для оновлення програмного та комп'ютерного забезпечення підприємства. Принципи, що засновані на відокремленні рішень щодо здійсненності проекту від тих, що стосуються економічної цінності конкретних дизайнів. Для цього необхідна розробка інформаційних, математичних та програмних моделей. Для прийняття рішень потрібно мати повний набір моделей для задач дослідження. Програмна реалізація деяких алгориттмів та моделей дозволить експериментально перевірити ефективність обраних стратегій.

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 двох класів задач прийняття рішень: прийняття управлінських рішень в організаційних системах; керування технічними системами. Узгодження рішень у проблемно-орієнтованих інформаційних системах та системах керування

Предмет дослідження. Предметом дослідження є принципи узгодження прийняття рішень серед набору працівників, кожен з яких відповідає за певну частину загальної проблеми прийняття рішень. Ці принципи є фундаментальним питанням у розробці розподілених систем прийняття рішень (DDMS), наприклад, для визначення необхідного та планування оновлення програмного та комп'ютерного забезпечення підприємства, що завжди виконується в умовах неузгодженості думок на важливе питання – чи потрібен новий комп'ютер або новий програмний засіб. Координація в рамках DDMS вивчається в різних областях організаційного мислення. Наприклад, головним питанням теорії організації є механізми управління взаємозалежностями між діяльністю всередині

організації [7–9], у сфері управлінського контролю так звані системи управлінського контролю (MCS) призначені для забезпечення відповідності прийняття рішень цілям і стратегіям фірми шляхом використання множини механізмів і методів для координації вибору менеджерів [10 – 12].

Методи дослідження. Основними методами дослідження, що застосовані, і які потребує сформульована задача є програмування і математичні методи дослідження операцій. Для практичної реалізації задачі використані принципи до проектування програмного забезпечення Domain-driven design (DDD) адаптовані до, програмних технологій стандартних патернів програмування NET.

Практична цінність - дозволяє покращити якість 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 принципи планування та програмний продукт є засобами, що може бути використаними для важливих задач планування виробництва. Засоби дозволяють отримати обґрунтований набір програмних технічних параметрів комп'ютерних мереж. 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 оновлення програмного забезпечення

У сучасному конкурентному середовищі особливого значення набуває проблема підвищення ефективності менеджменту (в широкому сенсі – проблема обробки інформації), що потребує розроблення та удосконалення системи аналітичного забезпечення [1,2]. Найбільш важливою частиною технології підвищення ефективності менеджменту (в широкому сенсі – проблема обробки інформації), що потребує розроблення та удосконалення системи аналітичного забезпечення є програмне та комп'ютерне забезпечення підприємства. Визначення необхідного та планування оновлення програмного та комп'ютерного забезпечення підприємства завжди виконується в умовах неузгодженості думок на важливе питання – чи потрібен новий комп'ютер або новий програмний засіб.

Визначення необхідного та планування оновлення програмного та комп'ютерного забезпечення підприємства завжди виконується в умовах неузгодженості думок на важливе питання – чи потрібен новий комп'ютер або новий програмний засіб. Наприклад… В навчальному закладі вивчають програмування з допомогою Visual studio 10, 15, 17, 19, 22. Ці засоби дозволяють проілюструвати всі необхідні технології, чи ні. Комп'ютери різні. Найшвидше виконує необхідні операції (реагує) Visual studio 10. Найкращій (здається) засіб Visual studio 22 у якого на більшості комп'ютерів час реагування дуже значний (можливо неприпустимо). Ситуація з якістю і вартістю цих засобів для навчання невизначена. Планування оновлення та придбання нових програмних та технічних засобів базуватися на реальних економічних можливостях і умовах змін функціонування комп'ютерного та програмного обладнання. Для побудови ієрархічної системи вподобань та переваг для програмного забезпечення використовується технологія

узгодження рішень. Координація між особами, які приймають рішення в організації, кожна з яких відповідає за певну частину загальної проблеми прийняття рішень для розв'язання задач оновлення та обслуговування програмних засобів, має вирішальне значення для досягнення загальної продуктивності. У сучасних системах управління в результаті поділу праці склалося положення, при якому готують, розробляють рішення одні працівники організації, приймають або затверджують інші, а виконують треті. Інакше кажучи, керівник часто затверджує і несе відповідальність за рішення, якого не розробляв, фахівці, які готували та аналізували рішення, не беруть участь у його реалізації, а виконавці не беруть участі у підготовці та обговоренні рішень, що готуються.

Для таких випадків [1], складність загальної проблеми прийняття рішення і, отже, потреба в координації формується взаємодією між її компонентами: якщо загальна проблема прийняття рішення (майже) розкладна (може бути поділена частини), вона можна бути розділена на (майже) часткові проблеми, так що зв'язки між під проблемами є сильнішими, ніж взаємодії між частковими проблемами. Як наслідок, під проблеми можна вирішувати незалежно одна від одної, не беручи до уваги позитивні чи негативні взаємодії щодо загального вирішення проблеми

Навпаки, якщо загальна проблема прийняття рішення нерозкладна, не можна знайти розкладання на під проблеми, яке (майже) зменшує взаємодію між під проблемами

Задача формально може формулюється як специфічна задача дискретного програмування про призначення – це відома модель дослідження операцій. В даному випадку є n типів (тип визначає призначення) програм (видів робіт, визначає тип) та n конкретних програмних та темничних засобів – потенційних кандидатів для виконання кожного типу робіт (виконавців). Вважається, що кожен з програмних і

технічних комплексів кандидатів $I \in J_M = \{1, ..., M\}$ може виконувати будь-яку роботу (вище був наведений приклад Visual studio 10, 15, 17, 19, 22.)

$J \in J_M = \{1, ..., M\}$, при цьому повинна бути визначена упорядкованість $ФС_{ij}$ – ієрархія ефективності виконаної роботи j-го виду і-м технічно - програмним засобом, та витрати на впровадження $ВС_{ij}$ (це теж може бути упорядкована ієрархія). У сферах організаційного пошуку [4] методи поетапного пошуку упорядкування дозволяють визначити ступень змін у термінах дистанції нових знайдених варіантів порівняно зі статус-кво, тобто дослідницькими, експлуатаційними стратегіями пошуку.

Необхідно знайти максимальні значення в упорядкованості, що задовольняють обмеженням на витрати.

Ця модель задачі має декілька важливих проблемних елементів. В реальній ситуації $ФС_{ij}$ невідома але може бути оцінена з допомогою апарату узгодження рішень - експертних оцінок.. На відміну від типових задач про призначення кількість типів операцій та кількість програмних засобів кандидатів на виконання не співпадає. По третє для кожної роботи (завдання) існує свій наперед відомий список кандидатів. Останнє, оновлювати програмне забезпечення потрібно лише у таких випадках коли приріст ефективності виконання робіт (цей параметр теж невідомий) більше ніж затрати на оновлення.

Якщо використати стандартний детермінований підхід для максимізації корисності програм, то функція мети матиме вигляд

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

Повинні виконуватись такі умови:

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

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

– $X_{ij} \in \{0;1\}$. при умовах $g_j \geq 0$.

Всі функціональні залежності цієї моделі можуть бути визначені лише методом експертних оцінок.

Інформаційна система для визначення переваг повинна надавати три можливості для оцінювання функціональності програмно технічних засобів:

- Оцінити швидкість реакції системи.

- Оцінити ступінь функціональності системи : чи забезпечує інтерфейс користувача достатню функціональність і відповідні функції? Чи забагато функціональних можливостей?

- Оцінити вплив на користувача: скільки часу потрібно для використання інтерфейсу користувача? Чи задоволений користувач? Чи правильно відповідає користувач? Чи може користувач отримати доступ до інформації, представленої інтерфейсом користувача?

- Визначити конкретні проблеми зручності використання : Чи є в інтерфейсі користувача помилки? Користувач заплутався?.

1.2 Існуючі технології дослідження.

Узгодження рішень на підприємстві [2] це процедура досягнення згоди членів організації із запропонованими рішеннями проблеми оновлення програмного та технічного забезпечення. **У** сучасних системах управління в результаті поділу праці склалося положення, при якому готують, розробляють рішення одні працівники організації, приймають або затверджують інші, а виконують треті. Інакше кажучи, керівник часто затверджує і несе відповідальність за рішення, якого не розробляв, фахівці, які готували та аналізували рішення, не беруть участь у його реалізації, а виконавці не беруть участі у підготовці та обговоренні рішень, що готуються.

Експертні оцінки - це формальні оцінки, які виконують проектувальники та експерти з зручності використання. Ефективність експертних оцінок полягає в тому, що вони займають менше часу на підготовку та не

потребують збору користувачів. Отже, експертні оцінки можуть відбуватися в будь-який час у процесі проектування. Якщо оцінки роблять проектувальники, вони повинні мати незвичайні навички, щоб оцінити свою роботу. Самооцінка вимагає відокремлення його від роботи та здатності множинного сприйняття проектів. Отже, проектувальник може найняти консультантів для проведення оцінки. Існує два популярних типи експертних оглядів це когнітивне проходження та евристичні оцінки

Поле теорії прийняття рішень [7] надає інструменти для раціонального прийняття рішень. Оптимальність визначається в термінах заяви про переваги, зроблені особою, яка приймає рішення. Визначення економічних переваг між альтернативами забезпечує просте засіб досягнення економічних цілей і добре зрозумілий тим, хто приймає рішення. Крім того, це математично добре заснована.

Розв'язування задачі за допомогою теорії прийняття рішень відбувається наступним чином. Усі альтернативи рішення ідентифікуються, разом із ними відповідні наслідки. Бажаність кожного наслідку визначається за допомогою тверджень про перевагу. Імовірність використовується для вимірювання ймовірності наслідків, а функція корисності використовується для вимірювання бажаності пари альтернатива/наслідок. Використовуючи це формулювання, альтернатива забезпечує найвище очікуване значення.

Прийняття рішень за багатьма атрибутами дозволяє оцінювати альтернативи на основі ряду атрибутів. Загалом,атрибути не можуть бути безпосередньо об'єднані в одну цільову функцію, оскільки вони несумірні. Наприклад, атрибут прибуток або швидкість реагування є кількісним і містить числові одиниці, тоді як атрибут зручність є якісним і не має стандартний набір агрегатів. Теорія корисності (MAUT) [7] надає засоби для оцінки бажаності багато атрибут них наслідків і, таким чином, полегшує прийняття рішень. Для взаємно незалежних атрибутів функція корисності виражається як зважене підсумовування функцій корисності атрибутів.

Функція корисності — економічна модель визначення переваг економічних суб'єктів. Основною умовою концепту функції корисності є раціональна поведінка споживача, що виражається у виборі з численних альтернатив саме тих, які виводять його на високий рівень корисності. У мікроекономіці концепт функції корисності служить пояснення поведінки споживачів і виробників, тоді як макроекономіці їм користуються зображення переваг державних інтересів. Перша похідна функції корисності за кількістю певного блага називається граничною корисністю цього блага. Гранична корисність виражає, що додаткової корисності приносить додаткова одиниця блага і. Гранична корисність, що дорівнює 0, означає досягнення насиченості.

Більшість функцій корисності, що розглядаються в економіці, мають негативну другу похідну — закон спадної граничної корисності.

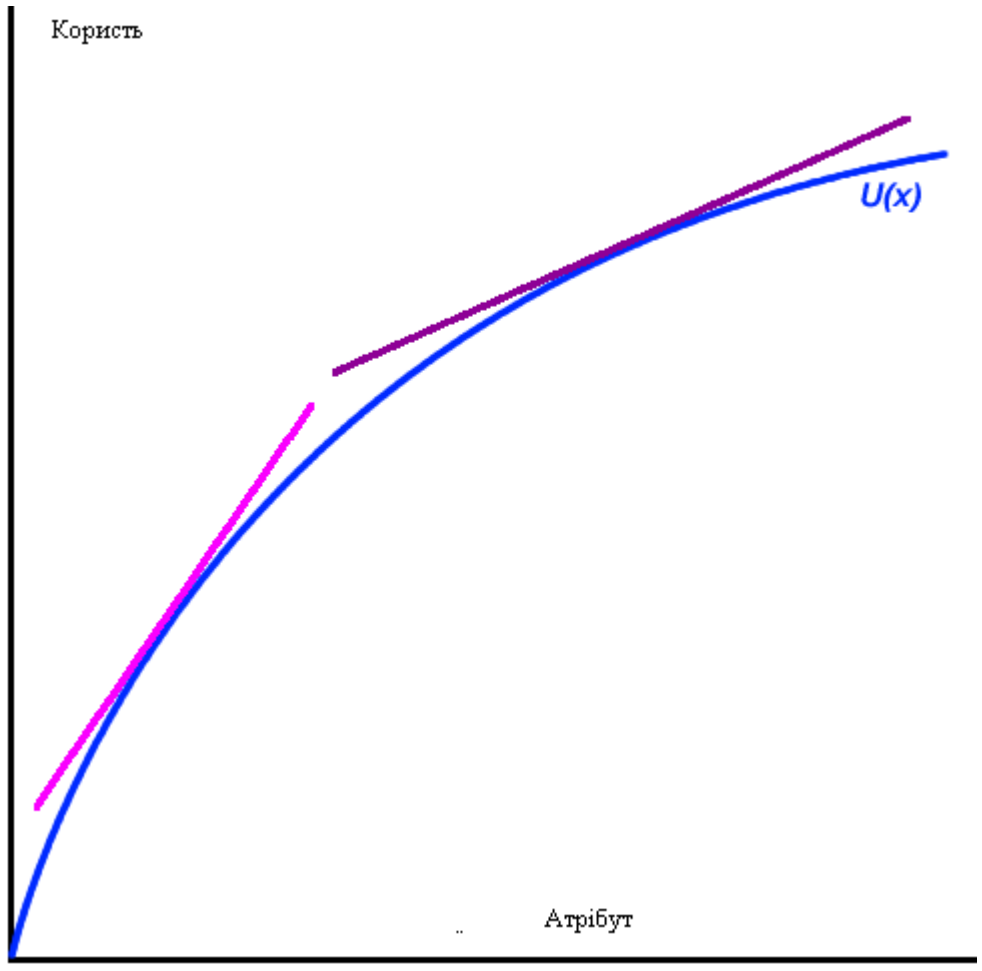


Рисунок 1.2.1

Основним обмеженням функції корисності є те, що всі можливі альтернативи повинні бути перераховані та оцінені, щоб визначити функцію корисності. Хоча ця вимога забороняє використання функції корисності для багатьох проблем, але можна використовувати неточну функцію корисності [18], яку можна вказати на основі підмножини альтернатив. Неточно визначена функція корисності визначає частковий порядок на множині можливих альтернатив і може бути використана для ідентифікації не домінуючого набору альтернатив.

Цей підхід до розподіленого прийняття економічних рішень на основі використання неточної функції корисності для врахування економічних та організаційних переваг. Підхід заснований на ієрархії атрибутів, а також про розподіл переваг координатора щодо підлеглих.

Ієрархія цілей/атрибутів

Для заданої задачі проектування визначається ієрархія атрибутів/цілей. Ця ієрархія подібна до тієї, що бере участь у будь-яка техніка на основі MAUT (Теорія корисності). Цілі уточнюються до точки, коли їх можна безпосередньо виміряти у фізичних термінах.

1.3. Висновки до розділу

Задача вибору програмного забезпечення базується на неповних відомостях про якість програм але повинна гарантувати (або бажано) покращувати якості та умови обробки інформації при обмежених витратах.

Визначення необхідного програмного та комп’ютерного забезпечення та планування оновлення програмного та комп’ютерного забезпечення підприємства завжди виконується в умовах неузгодженості думок на принципово важливе питання – чи потрібен, для виконання текучих функцій, новий комп’ютер або новий програмний засіб. Наприклад… В навчальному закладі вивчають програмування з допомогою Visual studio 10, 15, 17, 19, 22. Ці засоби дозволяють проілюструвати всі необхідні технології, чи ні. Комп’ютери різні. Найшвидше виконує

необхідні операції (реагує) Visual studio 10. Найкращій (здається) засіб Visual studio 22 у якого на більшості комп'ютерів час реагування дуже значний (можливо неприпустимо). Ситуація з якістю і вартістю цих засобів для навчання невизначена. Планування оновлення та придбання нових програмних та технічних засобів базуватися на реальних економічних можливостях і умовах змін функціонування комп'ютерного та програмного обладнання. Для побудови ієрархічної системи вподобань та переваг для програмного забезпечення використовується технологія узгодження рішень[4-6]. Координація між особами, які приймають рішення в організації, кожна з яких відповідає за певну частину загальної проблеми прийняття рішень для розв'язання задач оновлення та обслуговування програмних засобів, має вирішальне значення для досягнення загальної продуктивності.

Для таких випадків [1], складність загальної проблеми прийняття рішення і, отже, потреба в координації формується взаємодією між її компонентами: якщо загальна проблема прийняття рішення (майже) розкладна (може бути поділена частини), вона можна бути розділена на (майже) часткові проблеми, так що зв'язки між під проблемами є сильнішими, ніж взаємодії між частковими проблемами. Як наслідок, під проблеми можна вирішувати незалежно одна від одної, не беручи до уваги позитивні чи негативні взаємодії щодо загального вирішення проблеми.

Навпаки, якщо загальна проблема прийняття рішення нерозкладна, не можна знайти розкладання на під проблеми, яке (майже) зменшує взаємодію між під проблемами

Задача формально може формулюється як специфічна задача дискретного програмування про призначення – це відома модель дослідження операцій. В даному випадку є n типів (тип визначає призначення) програм (видів робіт, визначає тип) та n конкретних програмних та темничних засобів – потенційних кандидатів для виконання

кожного типу робіт (виконавців). Вважається, що кожен з програмних і технічних комплексів кандидатів I ∈ J M = {1, ..., M} може виконувати будь-яку роботу (вище був наведений приклад Visual studio 10, 15, 17, 19, 22.) J ∈ J M = {1, ..., M} , при цьому повинна бути визначена упорядкованість $ФС_{ij}$ – ієрархія ефективності виконаної роботи j-го виду і-м технічно - програмним засобом, та витрати на впровадження $ВС_{ij}$ (це теж може бути упорядкована ієрархія). У сферах організаційного пошуку [1] методи поетапного пошуку упорядкування дозволяють визначити ступень змін у термінах дистанції нових знайдених варіантів порівняно зі статус-кво, тобто дослідницькими, експлуатаційними стратегіями пошуку.

Необхідно знайти максимальні значення в упорядкованості, що задовольняють обмеженням на витрати.

Ця модель задачі має декілька важливих проблемних елементів. В реальній ситуації $ФС_{ij}$ невідома але може бути оцінена з допомогою апарату узгодження рішень - експертних оцінок.. На відміну від типових задач про призначення кількість типів операцій та кількість програмних засобів кандидатів на виконання не співпадає. По третє для кожної роботи (завдання)існує свій жорстко окреслений список кандидатів. Останнє, оновлювати програмне забезпечення потрібно лише у таких випадках коли приріст ефективності виконання робіт (цей параметр теж невідомий) більше ніж затрати на оновлення.

Розділ 2. ОБҐРУНТУВАННЯ МОДЕЛЕЙ ТА МЕТОДІВ

2.1. **Формальні моделі задачі**

В розділі 1 сформульована модель задачі, що базується на типовій задачі про призначення. Є n типів (тип визначає призначення) програм (видів робіт, визначає тип) та n конкретних програмних та темничних засобів – потенційних кандидатів для виконання кожного типу робіт (виконавців). Вважається, що кожен з програмних і технічних комплексів кандидатів $I \in Jm = \{1, ..., M\}$ може виконувати будь-яку роботу (вище був наведений приклад Visual studio 10, 15, 17, 19, 22.) $J \in J_M = \{1, ..., M\}$, при цьому повинна бути визначена упорядкованість $ФС_{ij}$ – ієрархія ефективності виконаної роботи j-го виду і-м технічно - програмним засобом, та витрати на впровадження $ВС_{ij}$ (це теж може бути упорядкована ієрархія). У сферах організаційного пошуку методи поетапного пошуку упорядкування дозволяють визначити ступень змін у термінах дистанції нових знайдених варіантів порівняно зі статус-кво, тобто дослідницькими, експлуатаційними стратегіями пошуку.

Необхідно знайти максимальні значення в упорядкованості, що задовольняють обмеженням на витрати.

Якщо використати стандартний детермінований підхід для максимізації корисності програм, то функція мети матиме вигляд

$$F(MX) = \sum_{I=1}^{N}\sum_{J=1}^{N} F_{IJ}(ФС_{IJ})X_{IJ} \rightarrow \text{MAX}.$$

Повинні виконуватись такі умови:

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

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

$$– X_{ij} \in \{0;1\} . \text{ при умовах } g_j \geq 0.$$

Всі функціональні залежності та параметри цієї моделі можуть бути визначені лише методом експертних оцінок. Функціональні залежності , що є не чіткими це

F_{IJ} ($ФС_{IJ}$) та $Ф_{IJ}$ (X_{IJ}Ї.

Враховуючі технології розділу 1 необхідно констатувати, що модель 1.1 містить багато факторів, що необхідно визначити. Нижче наведемо модель що суттєво менше невизначених функціональних залежностей. Задача визначення або комплектування необхідного для роботи пакету програмного забезпечення , що має потрібну функціональність та вимагає мінімальних витрат, в цьому випадку, може бути сформульованою простіше. В наявності існує n видів програмного та технічного забезпечення з відомими стандартними функціями, Вони належать до типів які виконують споріднені функції. Кожен вид містить відому та визначену кількість програм, які виконують ці споріднені функції. Програмні пакети відрізняються вартістю і якісними показниками.

Також відомо, що

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

це множини програм, які визначені та належать і-му типу програмного технічного забезпечення (ПтЗ). Елементи, що містять множини

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

можуть бути відомим чином перераховані

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

В останньому рівнянні n – кількість типів програмного забезпечення.

1,2… - номери ПтЗ, які відповідають множині або групі ***i*** **– того** програмного забезпечення**.**

Після цих означень задача вибору оптимального програмного забезпечення може бути сформульована так. Для кожного типу необхідно

визначити такий номер для якого досягається максимальне значення оцінок якості:

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

при виконанні умови, що

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

де v – максимальні можливі витрати, що пов’язані з купівлею та налагодженням програмного та технічного забезпечення, f_i - функція якості, що зв'язана (визначена) з кожним вибраним програмним продуктом.

Задача, що була сформульована, базується на функціональних залежностях, що не можуть бути визначені точно. Ці залежності визначаються розподілено, різними учасниками процесу планування. В цьому випадку принципи їх взаємодії визначає модель розподіленого прийняття рішень. Модель розподіленого прийняття рішень в організаціях, які стикаються з проблемою прийняття рішень, яка зростає з часом, і це відображається в організаційному зростанні труднощів, відповідно. Учасники дізнаються про відповідні способи моделі розподіленого прийняття рішень координації з точки зору синтезу плану у процесі зростання. Опис моделі складається з наступних кроків: по-перше, описується модель зростаючих проблем прийняття рішень; наступний крок представляє два типи агентів, які «перебувають» в організаціях до кроку 3, Цей крок зокрема, детально визначає розподілених (або локальних) учасників прийняття рішень, включаючи їхні відповідні локальні проблеми, їхні стратегії пошуку та формування їхніх переваг. Різні схеми синтезу плану для координації між розподіленими особами, які приймають рішення, зафіксовані в моделі.

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

1. Рівень вхідних та результативних документів для формування моделей та даних та аналізу результатів.
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.

Проблеми рішення, які повинні вирішуватися агентами, моделюються відповідно до рамок діаграм NK-fitness (узгоджених рішень) [33, 34],. Основною особливістю діаграм NK є те, що вони дозволяють легко контролювати складність N вимірної проблеми прийняття рішень, яка визначається параметром K ; з «технічної» точки зору діаграм NK є стохастичної генерованими псевдо двійковими функціями з N бітами,

$$F : \{0, 1\}^N \longrightarrow \mathbb{R}^1$$

тобто [33,34], для короткого опису діаграм NK, що використовується в управлінській науці [35]). Відповідно до NK-структури, на кроці часу t організації стикаються з N - вимірною проблемою бінарного рішення, тобто різних можливих двійкових векторів.

$$\vec{d}_i = (d_{i1}, \ldots, d_{Ni}) , \qquad d_{ii} \in \{0, 1\}, \ i = 1, \ldots, N,$$

Кожен із двох станів забезпечує внесок у загальну продуктивність , де випадковим чином беруться з рівномірного розподілу. Параметр K відображає кількість тих варіантів вибору , які також впливають на внесок вибору в продуктивність і, таким чином, фіксує складність проблеми прийняття рішення, яку потрібно вирішити в термінах взаємодії між окремими рішеннями. Отже, внесок може залежати не лише від одного вибору, а й від K інших варіантів: . У разі відсутності взаємодії між виборами

K дорівнює 0, а K це максимальний рівень складності, де кожен окремий вибір і впливає на внесок у продуктивність кожного іншого бінарного вибору. Загальна продуктивність,досягнута за період t, є нормалізованою сумою

$$V_t = V\left(\vec{d}_t\right) = \frac{1}{N}\sum_{i=1}^{N} C_{it}.$$

Незважаючи на те, що наразі це охоплює ключові елементи «стандартної» структури NK, відмінною рисою моделі, представленої в цьому дослідженні, є те, що через зростання кількість окремих рішень, які має зробити організація, може збільшуватися протягом час, тобто , і разом з цим також може підвищитися рівень складності.

(

$$N(t) = \begin{cases} N_1, & \text{for } 0 < t < t_1, \\ \vdots & \\ N_s, & \text{for } t_{s-1} < t \le t_s, \\ \vdots & \\ N_S, & \text{for } t_{S-1} < t \le T, \end{cases}$$

Загальна продуктивність «динамічно» нормалізується до розміру проблеми Ns; крім того, під час аналізу результатів моделювання кінцева продуктивність надається по відношенню до глобальних максимумів відповідних ландшафтів продуктивності: інакше результати не можна порівнювати в часі та в різних діаграмах продуктивності.

2.2. Алгоритми узгодження

Для спрощення рішення розмірну задачу розбивають на непересічні часткові задачі однакового розміру. Кожна з цих під проблем делегується одній призначеній особі, яка приймає рішення *r* (тобто, одному агенту типу 1) — причому конкретні повноваження особи, яка приймає рішення *r* , пов’язані з відповідною під проблемою, підлягають режиму координації, реалізованому в цьому час. Однак призначені особи, які приймають рішення,

принаймні, готують вибір щодо свого розподілу загальної проблеми прийняття рішень у кожному періоді t .

Процеси зростання в проблемному просторі відображаються відповідно в кількості розподілених осіб, які приймають рішення (тип 1). Якщо говорити більш формально, ми маємо кількість розподілених осіб, які приймають рішення, як функцію часу, тобто на кожній заданій стадії зростання s обсяг компетенції призначених осіб, які приймають рішення, є однаковим за кількістю окремих варіантів вибору.

У моделі, як згадувалося вище, розподілені особи, які приймають рішення, не можуть оглянути весь простір пошуку своєї проблеми прийняття рішення і, отже, вони не можуть «відразу знайти» оптимальне рішення своєї часткової проблеми прийняття рішень. ; радше, вони поетапно шукають кращі рішення. На кожному кроці часу t розподілена особа, яка приймає рішення r , розглядає два альтернативні рішення та для своєї часткової проблеми рішення порівняно зі статус-кво . Таким чином, у кожній із змодельованих стратегій пошуку та на кожному часовому етапі особа, яка приймає рішення r , має три варіанти на вибір, включаючи збереження статус-кво.

Однак модель фіксує як частина системи границь у MCS деякі межі, пов’язані з відстанями між двома альтернативами та порівнювані із статус-кво, встановлюються та забезпечуються центральним блоком (агентом типу 2). Зокрема, гранична система запроваджує стратегії пошуку, які мають експлуатаційний або дослідницький характер [8, 18, 20,] і, відповідно, модель містить наступні три стратегії пошуку: «тільки експлуатація»: як альтернативи , так і відкриті розподіленою особою, яка приймає рішення r , відрізняються від статус-кво на одну цифру відповідно. Зміни визначає відстань Хеммінга. Відстань Хеммінга між двома двійковими словами це число позицій, у яких відповідні цифри двох двійкових слів однакової довжини різні [1]. У загальнішому випадку відстань Хеммінга застосовується

для рядків однакової довжини будь-яких абеток, що складаються з q символів, і служить метрикою відмінності (функцією, що визначає відстань в метричному просторі) об'єктів однакової вимірності.

Отже, відстань Хеммінга для першої альтернативи до статус-кво дорівнює 1, як і для другої альтернативи «Зручність та якість»: спеціалістам, які приймають рішення, дозволено альтернативно розглядати варіант із перевернутим одним бітом та іншим, де дві цифри змінено, тобто відстані Хеммінга дорівнюють 2 « лише дослідження »: у кожній з обох альтернатив і два біти змінюються порівняно зі начальним.

Іншими словами, відстань Хеммінга вимірює мінімальну кількість замін, необхідних для зміни одного рядка в інший, або мінімальну кількість помилок, які могли перетворити одну стрічку в іншу. У більш загальному контексті відстань Хеммінга є однією з метрик рядків для вимірювання відстані редагування між двома послідовностями.

Кожна особа, яка приймає рішення r , переслідує свою «власну» мету (тобто пов’язану з її відповідною часткова проблемою). Зокрема, кожна розподілена особа, яка приймає рішення, прагне визначити той варіант , і який обіцяє найвищу чисту продуктивність , тобто найвищу різницю між частковою продуктивністю — результатом внесків тих конкретних окремих варіантів вибору, призначених особі, яка приймає рішення A_r - і відповідні витрати на зусилля для реалізації :

$$A_t^r\left(\vec{d}_t^r\right) = P_t^r\left(\vec{d}_t^r\right) - Z_t^r\left(\vec{d}_t^r\right),$$

Де V^r часткова ефективність внеску особи, яка приймає рішення r , у загальну продуктивність (див. рівняння нижче)) визначається як для V_i

$$V_i = V\left(\vec{d}_t\right) = \frac{1}{N_s}\sum_{i=1}^{N_s} C_{it}.$$

.

У випадку взаємодії між охопленими часткова проблемами, вибір особи, яка приймає рішення r , може вплинути на внесок вибору інших осіб, які приймають рішення, на роботу q , і навпаки.

У моделі кількість одиничних варіантів змінилася в термінах відстані Хеммінга до статус-кво (тобто, кількість перевернутих бітів) розглядається як зусилля , які має вжити особа, яка приймає рішення r , щоб реалізувати варіант i. Таким чином, можливий діапазон зусиль формується стратегією пошуку, представленою раніше : нижня межа зусиль у всіх трьох стратегіях пошуку дорівнює 0 у випадку, коли вибрано збереження статус-кво; верхня межа дорівнює 1 у «тільки дослідження» та дорівнює 2 у «тільки впровадження», а також у двосторонній стратегії

$$h(\overrightarrow{d^{r,a1}}) = \sum_{i=1}^{N^r} |\overrightarrow{d_{t-1}^{r,*}} - \overrightarrow{d_t^{r,a1}}|$$

Для моделювання вартості зусиль, як це прийнято в економіці (наприклад, [3 , 8]), передбачається, що більш високі рівні зусиль є все більш дорогими. Отже, для розподіленої вартості зусиль особи, яка приймає рішення r , ми маємо $Z^r(h)' > 0$ Зокрема, вартість зусиль особи, яка приймає рішення r , моделюється як квадратична зростаюча з відстанню Хеммінга h до статус-кво,

$$Z_t^r\left(\overrightarrow{d_t^r}\right) = z^r \cdot \left(h\left(\overrightarrow{d_{t-1}^{r,*}}, \overrightarrow{d_t^r}\right)\right)^2,$$

де z^e – коефіцієнт витрат. Для простоти в експериментах з моделюванням коефіцієнт витрат є однаковим для всіх осіб, які приймають рішення r , і не змінюється з часом. З цієї причини згодом індекс r , що вказує на особу, яка приймає рішення, r пропускається при зверненні до коефіцієнта витрат (враховуючи, що особи, які приймають рішення, неоднорідні щодо своїх функцій витрат (оскільки це можна охопити різними коефіцієнтами витрат) , буде природним продовженням зусиль.

Оцінка варіантів

Система надає учасникам інформацію. Оцінюючи свої варіанти, децентралізовані особи, які приймають рішення, використовують для взаємодії два типа обмеженої інформації: по-перше, без можливого подальшого спілкування, передбаченого механізмом координації, особа, яка приймає рішення r, не може передбачити інформацію інших осіб, які приймають рішення. вибір і, таким чином, припускаємо, що вони залишаться при статус-кво, тобто, що вони виберуть $d_{t-}^{i|*}$

$$V_t = V\left(\vec{d}_t\right) = \frac{1}{N_s}\sum_{i=1}^{N_s} C_{it}.$$

Це особливо важливо у випадку взаємозалежності між частковими проблемами, коли дії інших місцевих осіб, які приймають рішення, можуть вплинути на продуктивність особи, яка приймає рішення r.

$$\widetilde{P_t^r}\left(\vec{d}_t^{\,r}\right) = P_t^r\left(\vec{d}_t^{\,r}\right) + e^r\left(\vec{d}_t^{\,r}\right),$$

По-друге, особи, які приймають рішення, не в змозі досконало оцінити свої нещодавно відкриті варіанти та вплив на їх часткову ефективність .

Скоріше попередня оцінка вражена певним шумом. Зокрема, задля простоти, уявна часткова ефективність особи, яка приймає рішення , спотворюється відносною помилкою , яка приписується справжній продуктивності. Завдяки цьому наші місцеві особи, які приймають рішення, шукають шумові ландшафти часткової продуктивності. Члени похибки відповідають розподілу Гаусаз очікуваним значенням 0 і стандартними відхиленнями для кожної особи, що приймає рішення r ; помилки вважаються незалежними одна від одної. Отже, сприймана продуктивність розподіленої особи, яка приймає рішення r , визначається які цільова функція

$$A_t^r\left(\vec{d}_t^{\,r}\right) = P_t^r\left(\vec{d}_t^{\,r}\right) - Z_t^r\left(\vec{d}_t^{\,r}\right), \qquad \textbf{2.1}$$

Кожна призначена особа, яка приймає рішення r , має чітке часткове та викривлене «уявлення» про справжню пристосованість: кожна особа, яка приймає рішення r , виключно відповідає за «власну» частину всього N - вимірного рішення. Проблема, оскільки вона формується на поточній стадії зростання та недосконало оцінює внесок у продуктивність нещодавно відкритих варіантів. Таким чином, на певній стадії зростання модель фіксує неоднорідні (спотворені) види справжнього ландшафту, тобто один погляд на місцевого співробітника плюс перспективу керівника. (Однак для варіанту статус-кво ми припускаємо, що кожна особа, яка приймає рішення r , пам'ятає продуктивність, отриману за останній період, і, таким чином, знає фактичну ефективність статус-кво, якщо вона знову буде реалізована в період t .

На основі оцінки варіантів кожна особа, яка приймає рішення, складає список уподобань, $L_t^r - \{d_t^{r,p1}, d_t^{r,p2}, d_t^{r,p3}\}$ де позначає найбільш бажаний d_{t-1}^{r*} варіант. Відповідно, позначає другий і третій за перевагою варіанти.

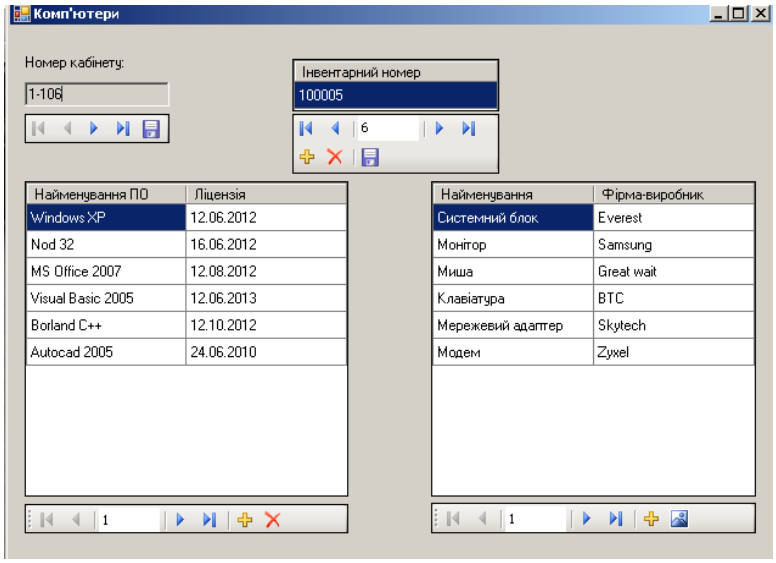
N	x_i	$f_i(x_i)$	x_i	$f_i(x_i)$	$F_1(y)$	$F_2(y)$
1	5	92	0.0	0.12	92	
2	5.3	42.32	0.6	1.08	42.32	32.12
3	5.6	54.08	1	3	54.08	33.08
4	5.9	67.28	1.4	5.88	67.28	46.4
5	6.2	81.92	1.8	9.72	81.92	55.16	
6	6.5	97.000000000000	2.2	14.52	97.000000000000	66.36
7	6.8	115.52	2.6	20.28	115.52	83
8	7.1	134.48	3	27	134.48	96.0000000000000
9	7.4	154.88	3.4	34.68	154.88	116.6
10	7.7	176.72	3.8	43.32	176.72	135.56
11	8	200	4.2	52.92	200	155.96
12	8.3	224.72	4.6	63.48	224.72	177.8

Таблиця 2.2.1. Розрахунок бінарних переваг

В результаті пошуку та (недосконалої) оцінки варіантів, наразі кожна розподілена особа, яка приймає рішення, має впорядкований список переваг, який, з точки зору систем, відображає індивідуальний [2] або локальний план [9]. Наступним кроком протягом кожного періоду t є прийняття рішення щодо загальної проблеми організації , тобто об’єднання уподобань децентралізованих осіб, які приймають рішення, зафіксованих у списках , у загальну конфігурацію. Це вимагає використання режиму координації для синтезу місцевих планів — і сама суть цієї статті полягає у дослідженні того, який режим виникає за яких умов стратегії пошуку та вартості зусиль.

У кожному періоді організації зазнають зростання проблемного простору , а також кількості розподілених осіб, які приймають рішення :(i)На своїй першій стадії зростання організації стикаються з багатовимірною загальною проблемою прийняття рішень, розбитою на дві підпроблеми однакового розміру , призначені двом особам, які приймають рішення , відповідно. На другому етапі зростання організації мають зробити три додаткові двійкові варіанти, за які відповідає один додатковий місцевий керівник, отже, організації складаються з децентралізованих осіб, які приймають рішення, тобто. На третьому етапі проблема прийняття рішення зростає трьома подальшими бінарними варіантами (Таблиця 2.2.1) та; отже, нарешті організації мають справу з проблемою розмірності , і в гру вступає четверта децентралізована особа, яка приймає рішення.

Як було зазначено в Розділі 1, у різних сферах було запропоновано безліч режимів для синтезу місцевих планів. З численних можливих механізмів модель включає три режими, які можуть представляти особливо виражені (якщо не сказати «екстремальні») форми координації (Таблиця 2.2.2) .



Таблиця 2.2.2

Зокрема, три типи координації, зафіксовані в моделі, відрізняються за каналами зв'язку (модель припускає, що зв'язок під час координації працює ідеально; для дослідження наслідків ненавмисних помилок зв'язку див. [8]), інформація, яка використовується для прийняття рішень, місце прийняття остаточного рішення та, зрештою, забезпечена якість координації.

В результаті пошуку та (недосконалої) оцінки варіантів, наразі кожна розподілена особа, яка приймає рішення, має впорядкований список переваг, який, з точки зору подібних систем, відображає індивідуальний [2] або локальний план [9]. Наступним кроком протягом кожного періоду є прийняття рішення щодо загальної проблеми організації , тобто об'єднання уподобань децентралізованих осіб, які приймають рішення, зафіксованих у списках , у загальну конфігурацію. Це вимагає використання режиму координації для синтезу місцевих планів — і сама суть цієї статті полягає у дослідженні того, який режим виникає за яких умов стратегії пошуку та вартості зусиль.

Найвищий рівень автономії надається особам, що приймають рішення, якщо кожному з них дозволено вибрати найбільш бажаний варіант. Потім виходить загальна організаційна конфігурація. Отже, для задачі часткового рішення особи, яка приймає рішення r , відповідний варіант реалізується.

2.3. Висновки до розділу

Основою 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 є основою для формування моделей прийняття рішень. Параметри моделей є не чіткими та формуються з допомогою методів узгодження рішень [1].

В розділі розглянуто дві моделі для визначення оптимальної програми планування структури програмних та технічних засобів. Для прийняття рішення прийнята найпростіша модель. Параметри моделі формуються з допомогою методів узгодження рішень.

В обох випадках задача формулюється як специфічна задача дискретного програмування про призначення – це відома модель дослідження операцій. В даному випадку є n типів (тип визначає призначення) програм (видів робіт, визначає тип) та n конкретних програмних та темничних засобів – потенційних кандидатів для виконання кожного типу робіт (виконавців). Вважається, що кожен з програмних і технічних комплексів кандидатів $I \in J_M = \{1, ..., M\}$ може виконувати будь-яку роботу J із множини $J_M = \{1, ..., M\}$, при цьому повинна бути визначена упорядкованість $ФС_{ij}$ – ієрархія ефективності виконаної роботи j-го виду і-м технічно - програмним засобом, та витрати на впровадження $ВС_{ij}$ (це теж може бути упорядкована ієрархія). У сферах організаційного пошуку [4] методи поетапного пошуку упорядкування дозволяють визначити ступень змін у термінах дистанції нових знайдених варіантів порівняно зі статус-кво, тобто дослідницькими, експлуатаційними стратегіями пошуку.

Необхідно знайти максимальні значення в упорядкованості, що задовольняють обмеженням на витрати.

Ця модель задачі має декілька важливих проблемних елементів. В реальній ситуації $ФС_{ij}$ невідома але може бути оцінена з допомогою апарату узгодження рішень - експертних оцінок.. На відміну від типових задач про призначення кількість типів операцій та кількість програмних засобів кандидатів на виконання не співпадає. По третє для кожної роботи (завдання) існує свій наперед відомий список кандидатів. Останнє, оновлювати програмне забезпечення потрібно лише у таких випадках коли приріст ефективності виконання робіт (цей параметр теж невідомий) більше ніж затрати на оновлення.

Якщо використати стандартний детермінований підхід для максимізації корисності програм, то функція мети матиме вигляд

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

Повинні виконуватись такі умови:

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

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

– $X_{ij} \in \{0;1\}$. при умовах $g_j \geq 0$.

Всі функціональні залежності цієї моделі можуть бути визначені лише методом експертних оцінок.

Друга модель, що пропонується містить суттєво менше невизначених функціональних залежностей. Задача визначення або комплектування необхідного для роботи пакету програмного забезпечення , що має потрібну функціональність та вимагає мінімальних витрат, в цьому випадку, може бути сформульованою простіше. В наявності існує n видів програмного та технічного забезпечення з відомими стандартними функціями, Вони належать до типів які виконують споріднені функції. Кожен вид містить

відому та визначену кількість програм, які виконують ці споріднені функції. Програмні пакети відрізняються вартістю і якісними показниками.

Також відомо, що

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

це множини програм, які визначені та належать і-му типу програмного технічного забезпечення (ПтЗ). Елементи, що містять множини

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

можуть бути відомим чином перераховані

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

В останньому рівнянні n – кількість типів програмного забезпечення.

1,2… - номери ПтЗ, які відповідають множині або групі ***i* – того** програмного забезпечення**.**

Після цих означень задача вибору оптимального програмного забезпечення може бути сформульована так. Для кожного типу необхідно визначити такий номер для якого досягається максимальне значення оцінок якості:

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

при виконанні умови, що

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

де v – максимальні можливі витрати, що пов’язані з купівлею та налагодженням програмного та технічного забезпечення, f_i - функція якості, що зв'язана (визначена) з кожним вибраним програмним продуктом.

Задача, що була сформульована, базується на функціональних залежностях, що не можуть бути визначені точно. Основою 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

РОЗДІЛ 3. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Обґрунтування принципів програмної реалізації

У міру розвитку інформаційних технологій та розробки нового програмного забезпечення компанії шукають все більше способів щодо швидкого створення безпечних додатків та систем, які охоплюють усі умови та при цьому легко тестуються та розробляються. На основі спроб і помилок було створено підхід, який багатьма організаціями сприймається як ідеальний спосіб створення програмного забезпечення [11]. **Domain Driven Design** (DDD) це спосіб створення додатків і систем, які мають переважно відображати потреби бізнесу - бізнес-припущення, які водночас стають необхідними умовами, включеними до програмного забезпечення. Одночасно Domain Driven Design – це методика створення програмного забезпечення, яка використовується у найбільших інформаційних системах бізнесу.

Дані що лежать в основі методу DDD це те що проектування будь-якого ПЗ має ґрунтуватися на рівноправній співпраці між програмістами та людьми, безпосередньо пов'язаними з бізнесом, які водночас є цільовим споживачем рішення. Другою умовою створення систем методом DDD є проектування логіки проекту як доменів.

У міру розвитку інформаційних технологій та розробки нового програмного забезпечення компанії шукають все більше способів щодо швидкого створення безпечних додатків та систем, які охоплюють усі умови та при цьому легко тестуються та розробляються. На основі спроб і помилок було створено підхід, який багатьма організаціями сприймається як ідеальний спосіб створення програмного забезпечення. Програмна архітектура застосування наведена на рисунку нижче

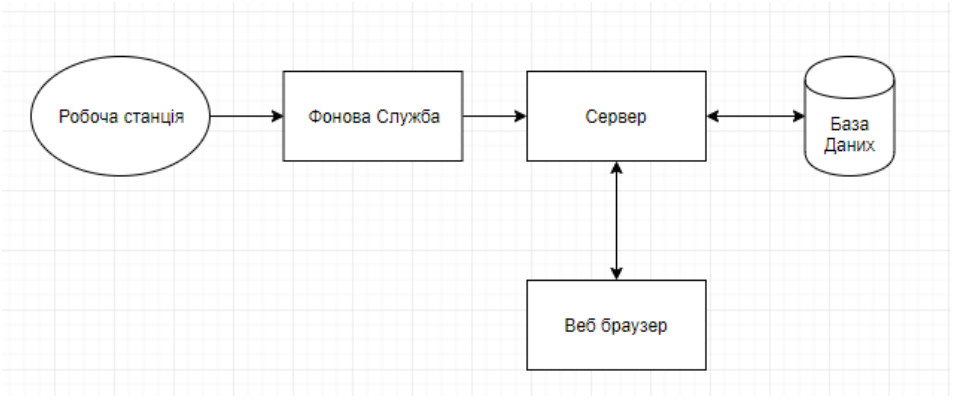


Рисунок 3.1.1

Оператори баз даних працюють за своєю робочою станцію і формують нормативні дані у штатному режимі.

Фонова служба яка завчасно встановлена та авторизована на сервері займається збором даних агентів, та їх пересиланням на сервер. Періодичність та інші опції функціонування налаштовуються безпосередньо на робочій станції.

В свою чергу на сервері буде функціонувати Web API, що буде отримувати дані від різних робочих станцій, проводити перевірку даних, та зберігати в БД.

Агент узгодження рішень, за допомогою браузера формує web-інтерфейс сервера, та отримує доступ до агрегованих, згрупованих за часовими проміжками даних за певними ознаками, з можливістю сортування та матеріали узгоджень (Таблиця 2.2.1).

Алгоритм визначення для таблиці оптимального варіанту відповідно до цих узгоджень відображений на лістингу нижче –(java script)

```
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, "");
    }
}
```

В технології Domain-driven design центром кожної стадії розробки програмного забезпечення є домен. В даному випадку два домени:

Домен операторів баз даних, що працюють за своєю робочою станцію і формують нормативні дані у штатному режимі.

Домен агентів узгодження рішень, що за допомогою браузера отримують доступ до агрегованих, згрупованих за часовими проміжками даних за певними ознаками, з можливістю сортування матеріалів узгоджень.

В основі методу Domain Driven Design лежать принципи проектування будь-якого ПЗ мають ґрунтуватися на рівноправній співпраці між програмістами та людьми, безпосередньо пов'язаними з бізнесом, які водночас є цільовим споживачем рішення. Другою умовою створення систем методом DDD є проектування логіки проекту як доменів.

Domain Driven Design – це методика створення програмного забезпечення, яка використовується у найбільших інформаційних системах бізнесу.

Які дані лежать в основі методу DDD? Насамперед, проектування будь-якого ПЗ має ґрунтуватися на рівноправній співпраці між програмістами та

людьми, безпосередньо пов'язаними з бізнесом, які водночас є цільовим споживачем рішення. Другою умовою створення систем методом DDD є проектування логіки проекту як доменів.

Через відносно невеликий відсоток проектів, створених з використанням методу DDD, дуже важко знайти людину, яка могла б взяти на себе таку роль з повною віддачею. Відсутність належного досвіду у програмістів у створенні за допомогою методології DDD також може спричинити додаткові труднощі.

Домен - це важлива функція компанії, важлива область в якій працює компанія в нашому випадку – програмне та технічне забезпечення, а також програмне забезпечення призначене для вирішення проблем у цій галузі. Домен ділять можна поділити на менші частини.. В даному випадку два домени:

Домен операторів баз даних, що працюють за своєю робочою станцію і формують нормативні дані у штатному режимі.

Домен агентів узгодження рішень, що за допомогою браузера отримують доступ до агрегованих, згрупованих за часовими проміжками даних за певними ознаками, з можливістю сортування матеріалів узгоджень

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, наприклад, формування звітів, парсинг csv, механізми кешування і т.д. Структура домену відповідає BoundedContext'ам. Проекти із папки Domain не залежать від Data. У Data знаходяться DbContext, міграції, конфігурації, що стосуються DAL (Data ACCES LENGVIDG). Data залежить від сутностей Domain для побудови міграцій. Проекти з папки App використовують IOC-контейнер для залучення залежностей. Таким чином виходить досягти максимальної ізоляції коду домену від інфраструктури.

Для проектування моделі Domain-driven design було потрібно:

1. Визначити сутності, які визначає інформаційна модель бази даних.
2. Визначити зв’язки між сутностями.
3. Визначити об’єкти для сутностей і основні властивості (атрибути) об’єктів.

4. Визначити зв’язки між властивостями об’єктів.

5. Побудувати схему бази даних, що має визначити відношення між таблицями баз даних, засновуючись на зв’язках між об’єктами даних, що містяться в таблиці, і включити цю інформацію до словника даних.

6. Визначити операції, що виконуються при створенні та зміні інформації таблиць, включаючи забезпечення цілісності даних.

7. Визначити, як використовувати індекси оптимальної структури для прискорення виконання запитів, щоб уникнути занадто сильного уповільнення роботи при додаванні інформації даних до таблиць, надмірного збільшення об’єму дискового простору, що займається базою та складнощів адміністрування..

8. Визначити права користувачів, яким дозволений доступ до даних, їх редагування, а також дозволи при змінах структури таблиць (при необхідності).

9. Механізмами побудови визначити структуру бази даних в цілому, завершити створення словників даних для бази та для кожної таблиці, що міститься в ній на всій сукупності розподілених пристроїв, розробити процедури обслуговування для операцій з базою даних, що включають створення резервних копій і механізми відновлення вихідних файлів.

Всі ці вимоги реалізує EDM модель. Модель EDM – це набір основних понять, що описують структуру даних незалежно від форми зберігання. Модель EDM запозичує властивості моделі «сутність-зв'язок», описаної Пітером Ченом у 1976 р., більше того, вона будується на моделі «сутність-зв'язок» та розширює можливості її традиційного використання. Спроектована для даної системи модель має вигляд

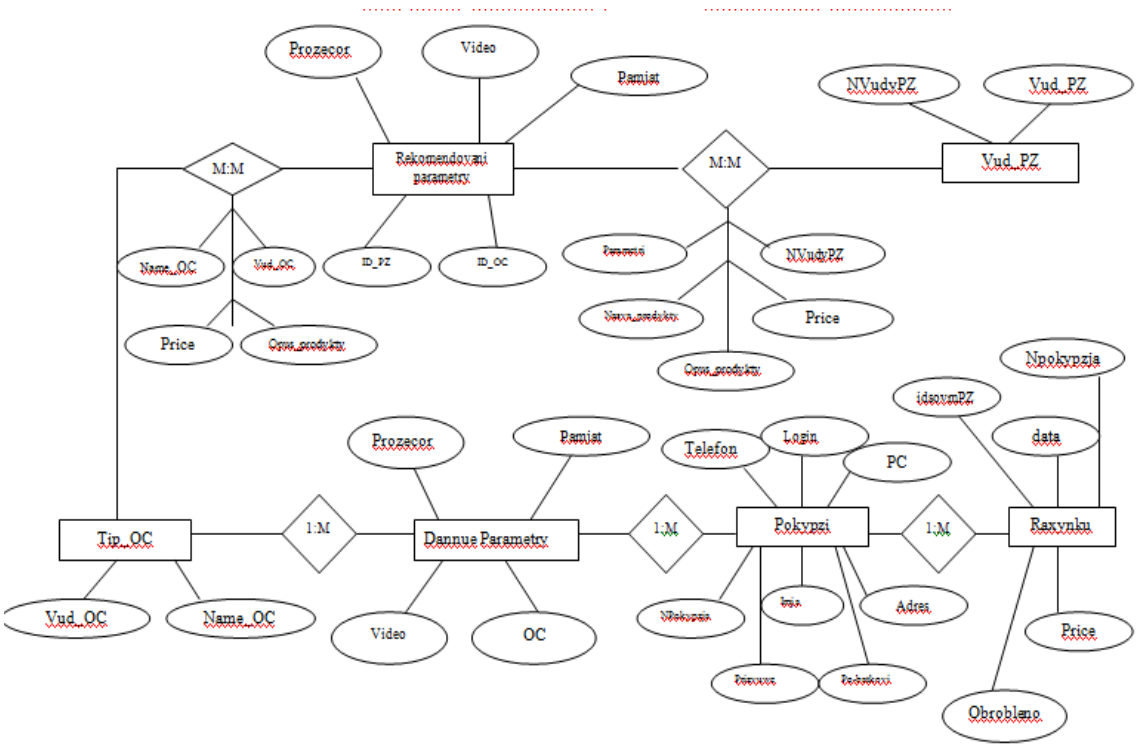


Рисунок 3.1.4

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

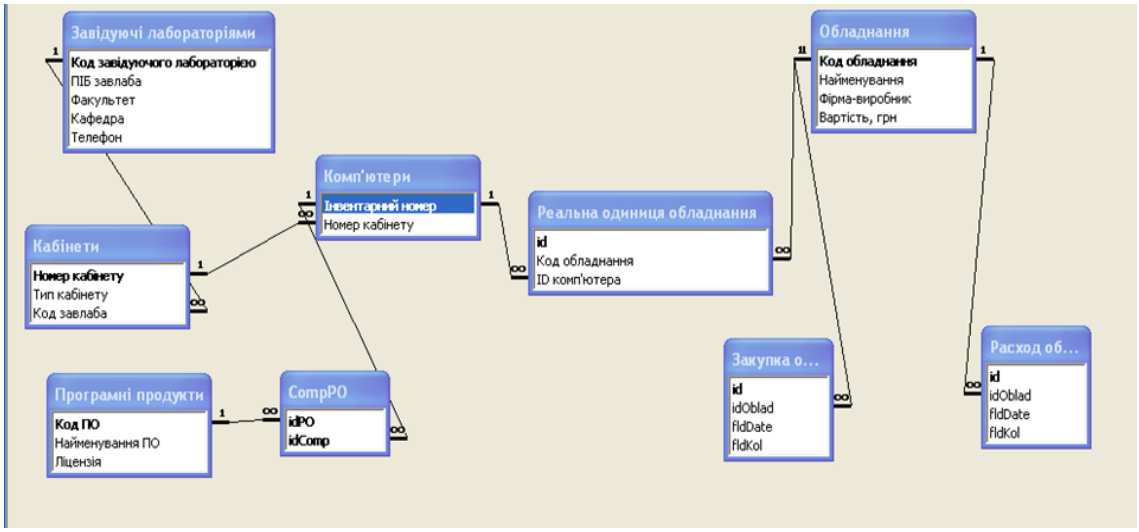


Рисунок 3.1.5

3.2. Керівництво користувача

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

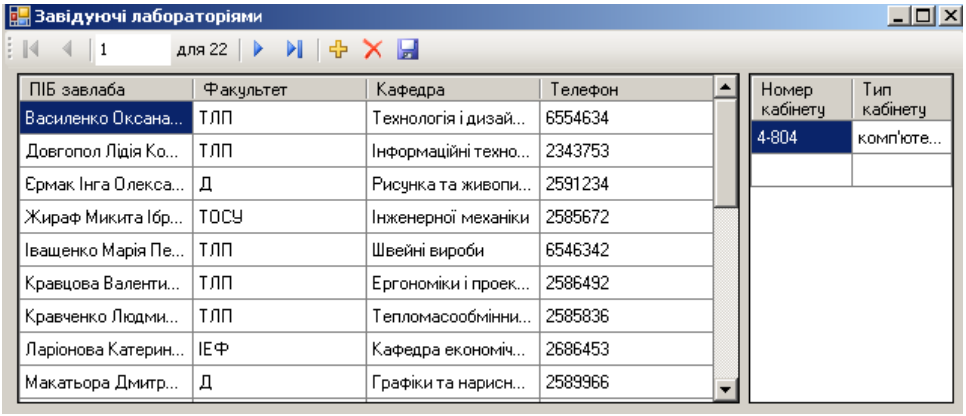


Рисунок 3.2.1

На формі знаходиться перелік зареєстрованих агентів у виді ви падаючого списку для зручності користування. Для оформлення прав потрібно вибрати з списку потрібне прізвище, потім внести інформацію про необхідну технічну платформу (систему: тактову частоту процесора, розмір оперативної пам'яті та відео карти.

Наступна форма Рисунок 3.2.2 визначає. програмні засоби, що займають нижні позиції ієрархії уподобань.

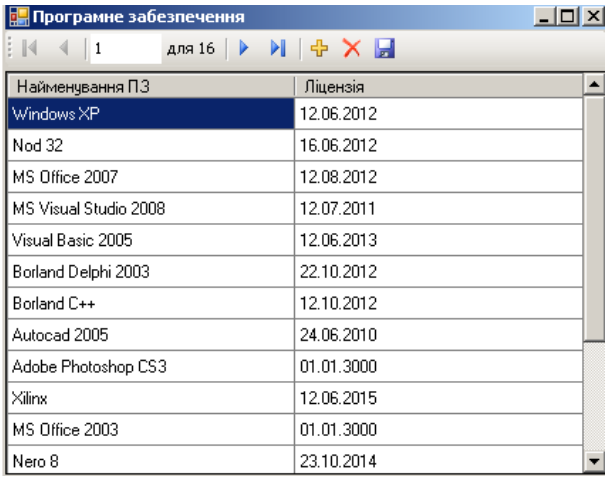


Рисунок 3.2.2

Наступна форма Рисунок 3.2.3 дозволяє реалізувати механізм бінарних порівнянь.

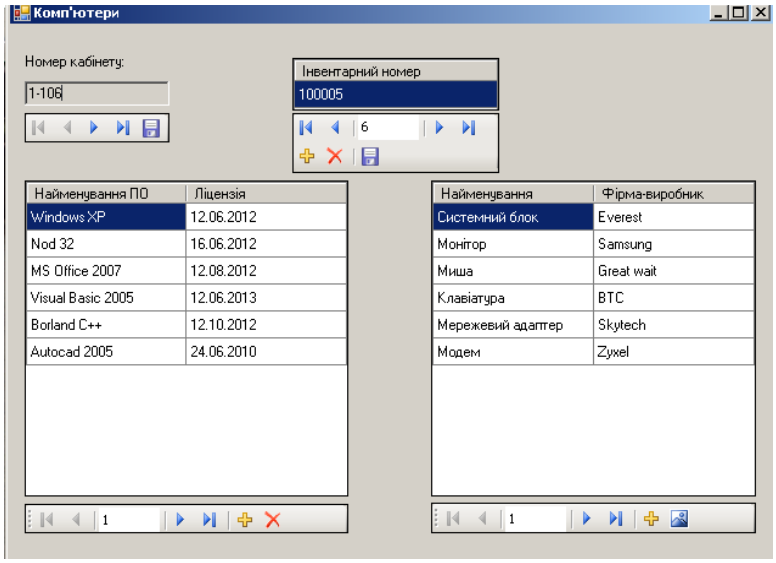


Рисунок 3.2.3

Наступна форма Рисунок 3.2.4 дозволяє реалізувати вибір найбільш узгоджений засіб для всіх уподобань

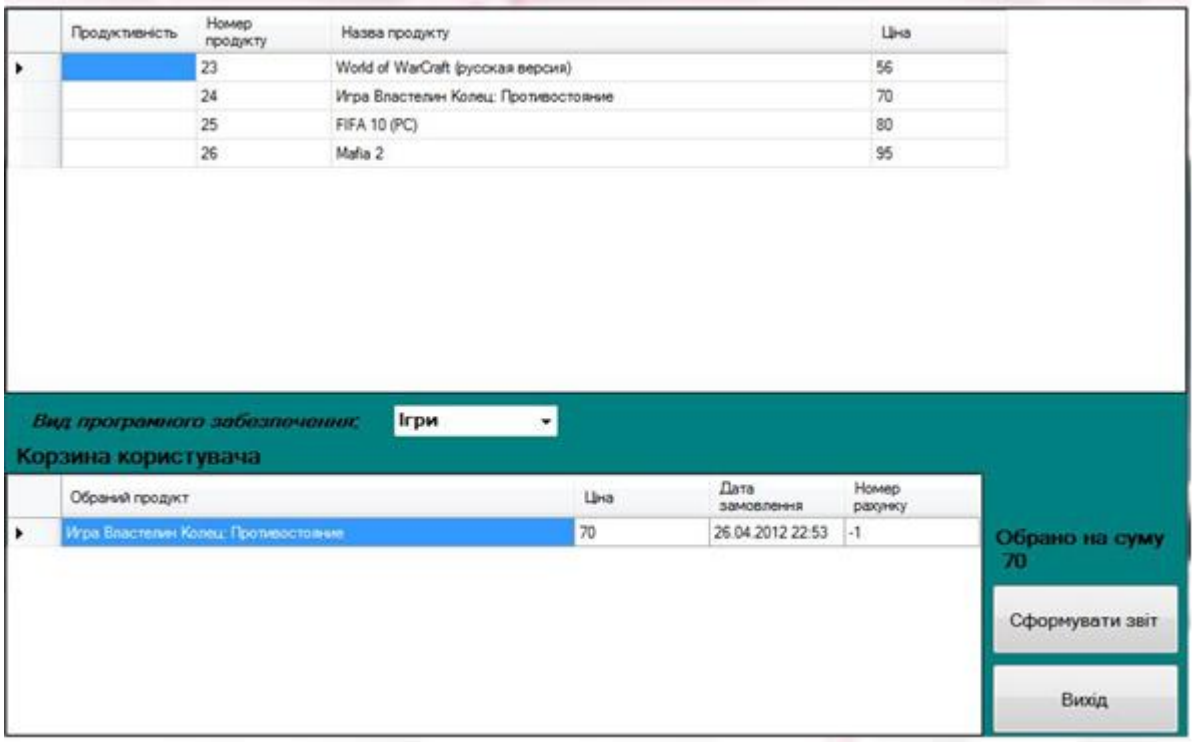


Рисунок 3.2.4

3.3 Висновки до розділу

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

1. Рівень вхідних та результативних документів для формування моделей та даних та аналізу результатів.
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.

Основою програмного засобу для обґрунтування рішень, що пов’язані із прийняттям планів для визначення необхідних нових та оновлення програмних засобів підприємства є інформаційна система програмних засобів, функцій програмних засобів та переваг. Інформація є основою для формування моделей

прийняття рішень. Система є розподіленою. Для формування інформації про програмні засоби, функції програмних засобів та переваги використовуються робочі станції на базі персональних комп’ютерів. Для отримання результатів можна використати будь який пристрій з доступом до Інтернет.

.

Висновки

Розглянуті практичні та теоретичні аспекти прийняття рішень для розв’язання задач оновлення та обслуговування специфічного виду обладнання – програмних засобів. Основою 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 є основою для формування моделей прийняття рішень. Параметри моделей є не чіткими та формуються з допомогою методів узгодження рішень.

СПИСОК ЛІТЕРАТУРИ

1. M. Jacob and R. Radner, Economic Theory of Teams, Yale University Press, London, UK, 2017.
2. Оптимізаційні методи та моделі : підручник / В.С. Григорків, М.В. Григорків. – Чернівці : Чернівецький нац. ун-т, 2016. – 400 с..
3. T. W. Malone and K. Crowston, “The interdisciplinary study of coordination,” ACM Computing Surveys, vol. 26, no. 1, pp. 87–119, 2017
4. K. M. Carley, “Computational organizational science and organizational engineering,” Simulation Modelling Practice and Theory, vol. 10, no. 5–7, pp. 253–269, 2012.
5. M. Wooldridge, An Introduction to MultiAgent Systems, Wiley, Chichester, UK, 2 edition, 2009.
6. J. D. Thompson, Organizations in Action. Social Science Bases of Administrative Theory, McGraw-Hill, New York, NY, USA, 2017.
7. J. R. Galbraith, “Organization design: an information processing view,” Interfaces, vol. 4, no. 3, pp. 28–36, 2017.
8. M. Henry, The Structuring of Organizations, Prentice Hall, Englewood Cliffs, NJ, USA, 2019.
9. K. A. Merchant and W. A. Van der Stede, Management Control Systems: Performance Measurement, Evaluation and Incentives, Pearson, New York, NY, USA, 4 edition, 2017.
10. K. Langfield-Smith, “A review of quantitative research in management control systems and strategy,” in Handbooks of Management Accounting Research, C. S. Chapman, A. G. Hopwood, and M. D. Shields, Eds., vol. 2, pp. 753–783, Elsevier, Amsterdam, Netherlands, 2006.
11. K. A. Merchant and D. T. Otley, “A review of the literature on control and accountability,” in Handbooks of Management Accounting Research, C. S. Chapman, A. G. Hopwood, and M. D. Shields, Eds., vol. 2, pp. 785–802, Elsevier, Amsterdam, Netherlands, 2016.

12. E. H. Durfee, “Distributed problem solving and planning,” in Multi-Agent Systems and Applications, M. Luck, V. Mak, O. Tpnkov et al., Eds., pp. 118–149, Springer, Berlin, Germany, 2011.

13. V. Lesser and D. Corkill, “Challenges for multi-agent coordination theory based on empirical observations,” in Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems, pp. 1157–1160, International Foundation for Autonomous Agents and Multiagent Systems, Paris, France, May 2014.

14. A. H. Van de Ven, M. Ganco, and C. R. Hinings, “Returning to the frontier of contingency theory of organizational and institutional designs,” The Academy of Management Annals, vol. 7, no. 1, pp. 393–440, 2013.

15. H. A. Simon, “The architecture of complexity,” Proceedings of the American Philosophical Society, vol. 106, no. 6, pp. 467–482, 2017.

16. J. W. Rivkin and N. Siggelkow, “Patterned interactions in complex systems: implications for exploration,” Management Science, vol. 53, no. 7, pp. 1068–1085, 2007.

17. N. Siggelkow, “Misperceiving interactions among complements and substitutes: organizational consequences,” Management Science, vol. 48, no. 7, pp. 900–916, 2017.

18. H. A. Simon, “A behavioral model of rational choice,” The Quarterly Journal of Economics, vol. 69, no. 1, pp. 99–118, 2017.

19. J. G. March, “Exploration and exploitation in organizational learning,” Organization Science, vol. 2, no. 1, pp. 71–87, 2017

21 J. G. March, “Exploration and exploitation in organizational learning,” Organization Science, vol. 2, no. 1, pp. 71–87, 1991.

22. O. Baumann, J. Schmidt, and N. Stieglitz, “Effective search in rugged performance landscapes: a review and outlook,” Journal of Management, vol. 45, no. 1, pp. 285–318, 2019.

23. R. Simons, Levers of Control: How Managers Use Innovative Control Systems to Drive Strategic Renewal, Harvard Business Press, Brighton, MA, USA, 1994.

24. S. K. Widener, “An empirical analysis of the levers of control framework,” Accounting, Organizations and Society, vol. 32, no. 7-8, pp. 757–788, 2007.

25. A. M. Kruis, R. F. Speklé, and S. K. Widener, “The levers of control framework: an exploratory analysis of balance,” Management Accounting Research, vol. 32, pp. 27–44, 2016.

26. J. Bisbe and D. Otley, “The effects of the interactive use of management control systems on product innovation,” Accounting, Organizations and Society, vol. 29, no. 8, pp. 709–737, 2004.

27. D. S. Bedford, “Management control systems across different modes of innovation: implications for firm performance,” Management Accounting Research, vol. 28, pp. 12–30, 2015.

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.