ДВНЗ «Донецький національний технічний університет»

Факультет комп'ютерно-інформаційних технологій та автоматизації

Кафедра прикладної математики та інформатики

«До захисту допущено»

Завідувач кафедри ПМІ

_____ **О. А. Дмитрієва**
(підпис) (ініціали, прізвище)

«____»_____2021 р.

Випускна кваліфікаційна робота магістра

на тему «Проєктування та розробка системи підтримки прийняття рішень для організації розподілу робіт»

Виконала: студента 2 курсу, групи КНм-20
(шифр групи)

спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

Ампольського В. С. _____
(прізвище та ініціали) (підпис)

Керівник доц. каф. ПМІ, к.т.н., доц. Маслова Н.О. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Нормоконтроль:

_____ *І. А. Назарова*
(підпис)

_____
(дата)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Покровськ – 2021 р.

ДВНЗ «Донецький національний технічний університет»

Факультет комп'ютерно-інформаційних технологій та автоматизації
Кафедра прикладної математики та інформатики
Освітній ступінь магістр
Спеціальність 122 Комп'ютерні науки
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри ПМІ
_____ /О. А. Дмитрієва/
«____»_____2021 року

З А В Д А Н Н Я

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Ампольському Вадиму Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Проєктування та розробка системи підтримки прийняття рішень для організації розподілу робіт

керівник роботи Маслова Наталя Олександрівна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом від « 22 » вересня 20 21 року № 570
2. Строк подання студентом роботи 7 грудня 2021 року
3. Вихідні дані до роботи результати переддипломної практики
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) дослідження предметної області; 2) розробка моделей і алгоритмів; 3) функціональна структура підсистеми; 4) розробка інформаційного і програмного забезпечення

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) 1) мета роботи; 2) аналогічні системи; 3) основні характеристики роботи; 4) цільова функція; 5) алгоритм відбору робіт для виконання; 6) діаграма варіантів використання підсистеми; 7) розроблений програмний додаток; 8) тривалість проекту в залежності від умов фінансування; 8) висновки по роботі

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Назарова Ірина Акопівна	–	–

7. Дата видачі завдання 4 жовтня 2021 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Порівняльний аналіз методів та засобів в області календарного планування. Постановка мети та задач дослідження	04.10.21-17.10.21	
2	Розробка математичного та алгоритмічного забезпечення системи	18.10.21-31.10.21	
3	Визначення основних функціональних можливостей системи. Розробка та проектування її функціонально-структурної схеми	01.11.21-14.11.21	
4	Розробка програмного додатку. Тестування та аналіз роботи системи	15.11.21-30.11.21	
5	Оформлення пояснювальної записки та опис створеної системи. Проходження нормоконтролю	01.12.21-06.12.21	
6	Перевірка пояснювальної записки антиплагіатною системою. Оформлення презентації до роботи, графічного матеріалу та рецензування роботи	07.12.21-20.12.21	
7	Захист роботи	21.12.21	

Студент _____ **Ампольський В.С.**
(підпис) (прізвище та ініціали)

Керівник роботи _____ **Маслова Н.О.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Ампольський В. С. Проєктування та розробка системи підтримки прийняття рішень для організації розподілу робіт / Випускна кваліфікаційна робота на здобуття освітнього ступеня «магістр» за спеціальністю 122 Комп'ютерні науки. – ДВНЗ ДонНТУ, Покровськ, 2021.

Об'єкт дослідження – процес відкриття фіксованої кількості торгівельних точок із дотриманням вимоги мінімізації фінансових та часових витрат.

Предмет дослідження – мінімізація часу виконання робіт для відкриття торгівельних точок за рахунок використання метаевристичних методів комбінаторної оптимізації.

Метою роботи є мінімізація часу виконання проекту з тим, щоб оптимізувати використання грошових коштів, що виділяються на проект.

Методи дослідження. У роботі використовується евристичний підхід до вирішення поставленої задачі. Для імітації процесу виконання робіт обрано алгоритм комп'ютерного моделювання подій та підхід, подібний до принципу роботи мурашиного алгоритму.

Наукова новизна передбачуваних результатів полягатиме в отримані максимально близького до оптимального рішення без повного перебору усіх можливих варіантів.

Практична цінність передбачуваних результатів – це автоматично згенерований порядок виконання робіт, який дозволить максимально оптимально використовувати наявні ресурси.

Ключові слова: календарне планування, метаевристичні методи, мурашиний алгоритм, система підтримки прийняття рішень, XML, мова C#.

Список публікацій здобувача:

1. Ампольский В.С. Підсистема прийняття рішень для організації розподілу робіт / В. С. Ампольський, Н. О. Маслова // «ТАК»: телекомунікації, автоматика, комп'ютерно-інтегровані технології (2021).

ABSTRACT

Ampolskiy V. Projecting and development of a decision support system for the organization of the division of labor / Graduation qualifying work for obtaining an educational degree «Master» specialty 122 Computer Science. – SHEI DonNTU, Pokrovsk, 2021.

The object of research is the process of opening a fixed number of outlets in compliance with the requirement of minimizing financial and time costs.

The subject of research is the minimization of work time for the opening of outlets through the use of metaheuristic methods of combinatorial optimization.

The aim of the work is to minimize the project implementation time in order to optimize the use of funds allocated for the project.

Research methods. The paper uses a heuristic approach to solving the problem. To simulate the process of performing work, an algorithm of computer simulation of events and an approach similar to the principle of operation of the ant algorithm were chosen.

The scientific novelty of the expected results will be obtained as close as possible to the optimal solution without a complete search of all possible options.

The practical value of the expected results is an automatically generated order of work, which will allow the best use of available resources.

Keywords: calendar planning, metaheuristic methods, ant algorithm, decision support system, XML, C # language.

Publisher’s publication list:

1. Ampolskiy V. Decision-making subsystem for the organization of the division of labor / V. Ampolskiy, N. Maslova // "TAC": telecommunications, automation, computer-integrated technologies (2021).

ЗМІСТ

ВСТУП

Актуальність теми. В умовах світової пандемії серед представників малого та середнього бізнесу спостерігається різке посилення конкуренції в різних сферах. Підприємцям все важче боротися за різні ринки, сфери впливу. Швидкість, правильність, своєчасність прийняття бізнес рішень – запорука успіху в даному виді діяльності. Щоб залишатися конкуренто здатним, необхідно вводити інновації та користуватися перевагами передових технологій [1].

При плануванні стратегій розвитку бізнесу у сфері торгівлі, при розробці стратегії виробництва необхідно контролювати свої витрати, графіки випуску продукції, графіки виконання всіляких виробничих робіт. Терміни виконання кожного типу робіт мають архіважливе значення, оскільки в умовах жорсткої конкуренції можливі стратегічні прорахунки, що призведе до необхідності поступитися перед конкурентом. Це може обернутися величезними фінансовими втратами, а в гіршому випадку – банкрутством і неможливістю відновлення колишніх позицій в даному секторі впливу. Фінансова складова також має величезне значення, оскільки невиконання зобов'язань перед, наприклад, посередниками, постачальниками або партнерами по бізнесу може призвести до стягнення штрафів за невиконання якихось зобов’язань в вказаний термін [2].

Мета і завдання роботи. Головною метою роботи є мінімізація часу виконання проекту з врахуванням оптимізації використання грошових коштів, що виділяються на проект.

Критеріями досягнення мети є:

– відкриття заданої кількості об’єктів торгівлі точок в зазначений термін;

– бюджет проекту не повинен бути перевищений;

– терміни виконання проекту повинні бути мінімальні, але з урахуванням обмеження бюджету.

Завдання дослідження. Для досягнення поставленої мети система повинна скласти графік виконання робіт, що є основним завданням календарного планування [4].

Система, що розробляється, вирішить проблему складання календарного плану [3] робіт по заданому проекту. При збільшенні кількості торгових точок збільшується кількість різних варіантів календарного плану. Працівник, відповідальному за складання календарного графіка, потребує програмний продукт, який допоможе у прийнятті даного рішення.

Предметом дослідження є мінімізація часу виконання робіт для відкриття торгівельних точок.

Об’єктом дослідження у даній роботі є процес, який полягає у відкритті фіксованої кількості торгівельних точок. Кожний новий об’єкт торгівлі повинен бути зданим в експлуатацію до певної дати. Процес відкриття нового магазину полягає у виконанні певного списку робіт, які, в залежності від певних факторів, властиві саме для цієї точки. Всі регламентовані роботи можуть виконуватися як паралельно, так і послідовно згідно технологіям та реальним можливостям власника бізнесу. Інвестування проводиться через фіксовані проміжки часу згідно з графіком, який визначається фінансовим менеджером або власником, наприклад, щомісяця.

Список робіт має наступний вигляд:

1) визначення кількості торгівельних точок;

2) пошук приміщень;

3) проектування структури точки;

4) ремонтні роботи;

5) пошук персоналу;

6) навчання персоналу;

7) процес ліцензування;

8) отримання дозвільних документів на торгівлю;

9) формування складу товарів:

– затвердження асортименту;

– пошук постачальників;

– укладення договорів на поставку;

– завезення товару (коли магазин або складське приміщення готові);

– приймання товару (можливо тільки після пункту 12, на відміну від попередніх).

10) замовлення торгівельного обладнання;

11) замовлення it-обладнання, оргтехніки, інсталяція програм;

12) установка обладнання;

13) закупівля витратних матеріалів [3].

Методи дослідження. У роботі використовується евристичний підхід до вирішення поставленої задачі. Для імітації процесу виконання робіт обрано алгоритм комп’ютерного моделювання подій та підхід, подібний до принципу роботи мурашиного алгоритму.

Наукова новизна передбачуваних результатів полягатиме в отримані максимально близького до оптимального рішення без повного перебору усіх можливих варіантів.

Практична цінність передбачуваних результатів – це автоматично згенерований порядок виконання робіт, який дозволить максимально оптимально використовувати наявні ресурси.

РОЗДІЛ 1
ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальні відомості про календарне планування

Календарне планування – це складання та доведення до структурних підрозділів і робочих місць переліку оперативних планових робіт, що взаємозв’язуються між собою в часі і з можливостями їх забезпечення різними видами матеріально-технічних та трудових ресурсів. [5].

Календарний план виробництва – це документ, який встановлює послідовність і терміни виконання всіх видів виробничих операцій, а також визначає потребу в усіх видах ресурсів, у часі.

Календарне планування в керуванні проектами – це ключовий і важливий процес, результатом якого є затверджений керівництвом компанії календарний план проекту (часто його називають ще планом-графіком, календарним графіком, планом керування проектом) [6]. Мета календарного планування полягає в отриманні точного і повного розкладу всього проекту з урахуванням необхідних робіт, їх тривалості, необхідних ресурсів. Отриманий розклад служить основою для виконання проекту.

Календарний графік (комплексний або зведений) – це графічна інтерпретація календарного плану (таблиця або діаграма Ганта), що конкретизує склад робіт, код робіт, послідовність, терміни виконання робіт. При побудові календарного графіка необхідно враховувати наявність матеріально-технічних та/або людських ресурсів, оскільки одночасне виконання деяких операцій через певні людські або матеріальні обмеження може виявитися неможливим [5].

Складання календарного плану-графіка проекту включає в себе кілька аспектів. Вкрай необхідно спланувати терміни робот, їх тривалість; визначити послідовність виконання робіт та взаємозв'язки між ними; закласти необхідну кількість ресурсів, врахувати витрати на забезпечення ресурсами та на

виконання цих робіт. Коли проект переходить на стадію практичної реалізації запланованих дій, саме за цим планом-графіком власник бізнесу та/або фінансовий директор відстежуватимуть хід виконання робіт. Якщо в проекті виникне непередбачувана ситуація, можна, звіривши з початковим планом проекту, внести відповідні зміни [7].

Як правило, план-графік проекту розробляється менеджером проекту із залученням людей, які є експертами в тій чи іншій галузі. Наприклад, вміст будівельних робіт найкраще знає фахівець з будівництва; а заходи по просуванню продукту, швидше за все, спланує маркетолог [8].

Крім складання календарного план-графіку виконання робіт, обов'язковим є також створення ресурсної моделі проекту. В рамках створення моделі необхідно встановити зв'язок між виконавцями тих чи інших робот або лише певних етапів, яких спеціалістів необхідно залучати до відкриття, хто є відповідальним за результат роботи або етапу. Таким чином, в будь-якому проекті залучаються як людські ресурси, так і витратні матеріали, сировина, а також можливе використання машин і механізмів, техніки, транспорту і т. д. [9].

Зазвичай, для візуалізації послідовності виконання робіт використовується мережевий графік. Мережевий графік – графічне зображення певного комплексу робіт, що відображає їх логічну послідовність, взаємозв'язок і тривалість [10].

Фрагмент мережевого графіка представлений на рис. 1.1.

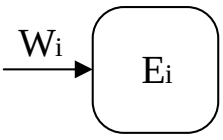

Рисунок 1.1 – Фрагмент мережевого графіка

W_i – і-а робота, E_i – подія завершення і-ой роботи. Шлях – це безперервна послідовність робіт і подій на мережевому графіку. Довжина

шляху визначається сумою тривалості складових його робіт. Повний шлях – це шлях, що ведуть від вихідної події до завершального. Таких шляхів може бути кілька. Тривалість повного шляху визначається як сума тривалостей робіт, що лежать на ньому. Критичний шлях – це повний шлях, що має найбільшу тривалість зі всіх повних шляхів. Тривалість такого шляху визначає тривалість всього комплексу робіт. Зазвичай для розрахунку мережевого графіка використовують метод критичного шляху [11].

1.2 Методи, які застосовуються для календарного планування

Задачі календарного планування відносяться до класу найбільш складних завдань управління виробництвом [12]. Основними факторами, що визначають високу складність календарного планування, є: велика розмірність, багатоваріантність, багатокритеріальність, комбинаторність, стохастичність, необхідність врахування тимчасового чинника, неформалізованності частини обмежень та ін. Завдання календарного планування відносяться до класу задач зі складною структурою алгебри. Методи їх вирішення розглядаються в рамках теорії розкладів та теорії масового обслуговування [13].

Задачі календарного планування поділяються на детерміновані та стохастичні. У першому випадку всі характеристики виконання робіт вважаються заздалегідь відомими і постійними. У другому випадку частина або всі характеристики і результати виконання робіт є випадковими величинами [14].

Для вирішення детермінованих задач використовуються такі підходи: комбінаторний, математичного програмування, евристичного і статистичного моделювання [15]. Дослідження показали, що отримання строго оптимального по якомусь критерію рішення завдання календарного планування можливе лише для найпростіших випадків. Уже при 4 машинах і 10 роботах задача календарного планування, сформульована в термінах цілочисельного програмування, має 220 змінних і 390 обмежень. Встановлено, що єдиними

методами рішення детермінованою завдання календарного в загальному випадку, що дають якусь надію на отримання практичних (не обов'язково оптимальних) результатів, є евристичні методи [16].

В якості критеріїв оцінки графіків робіт в детермінованих задачах використовуються час завершення робіт, час затримки (різниця між фактичним і плановим терміном завершення робіт), коефіцієнт використання машин і ін.

Критерії оцінки розкладів робіт в стохастичною постановці завдання календарного планування, тобто в умовах впливу на хід виконання робіт випадкових факторів, відрізняються від критеріїв, що використовуються в детермінованій постановці. При вирішенні задачі в стохастичною постановці спочатку отримують розподілу ймовірності показників, що оцінюють розклад або порядок проходження робіт, а потім визначають статистичні характеристики розподілів (середнє, дисперсію і т.і.). Дослідження показали, що єдиним ефективним методом дослідження таких задач виявляється комплексне моделювання, зокрема статистичне моделювання з використанням ЕОМ для зберігання, обробки і пошуку даних [17].

Були досліджені наступні методи вирішення задачі календарного планування: лінійне цілочисельне програмування, метод гілок і меж, динамічне програмування, імітаційне моделювання, генетичний алгоритм, мурашиний алгоритм. Далі буде розглянутий кожен метод окремо.

Задача лінійного цілочисельного програмування формулюється таким чином (форм. (1.1)-(1.5)):

$$F(X) = \sum_{j=1}^{n} c_j x_j \tag{1.1}$$

$$\sum_{j=1}^{n} a_{ij} x_j \le b_j \text{ , i=1,…, s;} \tag{1.2}$$

$$\sum_{j=1}^{n} a_{ij} x_j = b_j \text{ , i=s+1,…, s+t;} \tag{1.3}$$

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_j \text{ , i=s+t+1,…, m;} \quad (1.4)$$

$$x_j \geq 0 \text{ , j=1,…, n} \quad (1.5)$$

де x_j – змінні, c_j – коефіцієнти при змінних, a_{ij} – умови, b_j – обмеження [18].

Аналізуючи даний метод, були виділені його переваги і недоліки.

Переваги:

– метод знайде всі можливі варіанти;

– буде знайдений найоптимальніший варіант вирішення задачі.

Недоліки:

– побудова моделі буде займати багато часу при великій кількості змінних;

– рішення великою моделі з перебиранням усіх можливих варіантів потребує багато обчислювальних ресурсів.

Метод гілок і меж – один з методів комбінаторної оптимізації, суть якого полягає в упорядкованому переборі варіантів і розгляді лише тих з них, які виявляються за певними ознаками перспективними, і відкиданні безперспективних варіантів. Метод гілок і меж полягає в наступному: безліч допустимих рішень (компоновок) деяким способом розбивається на підмножини, кожне з яких цим же способом знову розбивається на підмножини. Процес продовжується до тих пір, поки не отримано оптимальне цілочисельне рішення вихідної задачі [19].

Метод має ті ж самі переваги та проблеми, що і попередній.

Динамічне програмування – розділ математичного програмування, сукупність прийомів, що дозволяють знаходити оптимальні рішення, засновані на обчисленні наслідків кожного рішення і виробленні оптимальної стратегії для наступних рішень [20].

Переваги:

– метод рекурсивно прийде до найкращого варіанту.

Недоліки:

– час рішення моделі залежить від кількості вхідних параметрів;

– існує ймовірність отримати субоптимальне рішення.

Імітаційне моделювання (ситуаційне моделювання) – метод, що дозволяє будувати моделі, що описують процеси так, як вони проходили б у дійсності. Таку модель можна «програти» в часі як для одного випробування, так і заданої їх множини. При цьому результати визначатимуться випадковим характером процесів. За цими даними можна отримати достатньо стійку статистику. Аналізуючи отримані результати, можна зробити висновок про оптимальність рішення [21].

Переваги:

– метод знайде всі можливі варіанти;

– буде знайдений найоптимальніший варіант вирішення задачі.

Недоліки:

– процес моделювання може зайняти дуже багато обчислювального часу.

Генетичні алгоритми – це процедури пошуку, засновані на механізмах природного відбору і спадкування. В них використовується еволюційний принцип виживання найбільш пристосованих особин. Вони відрізняються від традиційних методів оптимізації декількома базовими елементами. Зокрема, генетичні алгоритми:

– обробляють не значення параметрів самої задачі, а їх закодовану форму (хромосому);

– здійснюють пошук рішення виходячи не з єдиної точки, а з деякої популяції;

– використовують тільки цільову функцію, а не її похідні або іншу додаткову інформацію;

– застосовують імовірнісні, а не детерміновані правила вибору [22].

Генетичний алгоритм (ГА) являє собою особливий вид стохастичного алгоритму пошуку, який використовує біологічну еволюцію для вирішення

проблеми. ГА працює в просторі пошуку названому популяцією. Основні поняття генетичного алгоритму:

1) хромосома – особина, яка несе в собі закодоване рішення;

2) популяція – група хромосом;

3) кросинговер – процес схрещування двох хромосом.

На кожному кроці алгоритму відбувається відбір батьків для схрещування, отримання нащадків, відбір кращих особин в нову популяцію, також можливі мутації (змінення хромосоми) [23].

Проблема використання даного алгоритму полягає в тому, що необхідно визначити формат хромосоми. Кожна хромосома являє собою рішення, тобто шлях. Виникає питання про те, яка довжина хромосоми і як вона залежить від шляху. Також виникає питання про паралельно виконуваних роботах і про їх кодуванні в хромосому. Необхідно враховувати, що схрещування хромосом має задовольняти вимогам, які пред'являються до порядку виконання робіт. Не виключено, що закодовані дані необхідно буде розкодувати для коректного схрещування.

Переваги:

– швидкість знаходження оптимального рішення;

– мала ймовірність знаходження субоптимальне рішення.

Недоліки:

– складнощі зі способом кодування хромосоми для конкретної задачі;

– незрозуміло, як проводити процес схрещування хромосом.

Мурашиний алгоритм (МА) (алгоритм оптимізації за допомогою імітування мурашиної колонії, англ. Ant colony optimization, ACO) – один з ефективних поліноміальних алгоритмів для знаходження наближених рішень задачі комівояжера, а також аналогічних завдань пошуку маршрутів на графах. Суть підходу полягає в аналізі та використанні моделі поведінки мурах, що шукають шляхи від колонії до джерела живлення і являє собою метаевристичну (англ. Metaheuristic, meta – «за межами» і heuristic – «знайти») оптимізацію. Перша версія алгоритму, запропонована доктором наук Марко

Дориго в 1992 році, була спрямована на пошук оптимального шляху в графі [24].

МА широко використовується для вирішення задачі знаходження гамильтонова шляху в графі [25]. Поставлену задачу можна співвіднести із завданням знаходження гамильтонова шляху. Також дана задача схожа з завданням комівояжера, яка також часто вирішується за допомогою МА [24].

Переваги:

– мала ймовірність знаходження субоптимальне рішення.

Недоліки:

– тонке налаштування параметрів алгоритму;

– потребує налаштування під конкретну задачу.

1.3 Аналіз аналогічних систем

Зараз багато фірм намагаються вирішити завдання календарного планування. Далі будуть коротко розглянуті компанії та їх програмні продукти.

Easy Projects – це продукт, що розробила компанія Logic Software Inc., яка знаходиться у Торонто і займається розробкою програмного забезпечення на замовлення. У 2003 році компанії була потрібна система для керування проектом для своїх власних цілей.

Оскільки нічого відповідного в той час не існувало, то було прийнято рішення зайнятися розробкою подібної системи. У результаті система виявилася настільки зручною, що її перевели на комерційну основу, почали її поширення на ринку подібних систем [24].

На рис. 1.2 представлена одна з ключових можливостей – складання діаграми Ганта (послідовність робіт та їх взаємозв’язок) [26].

Як можна помітити, існує можливість формування списку робіт, їх угруповання, зазначення дати початку та дати кінця, автоматичне формування діаграми Ганта.

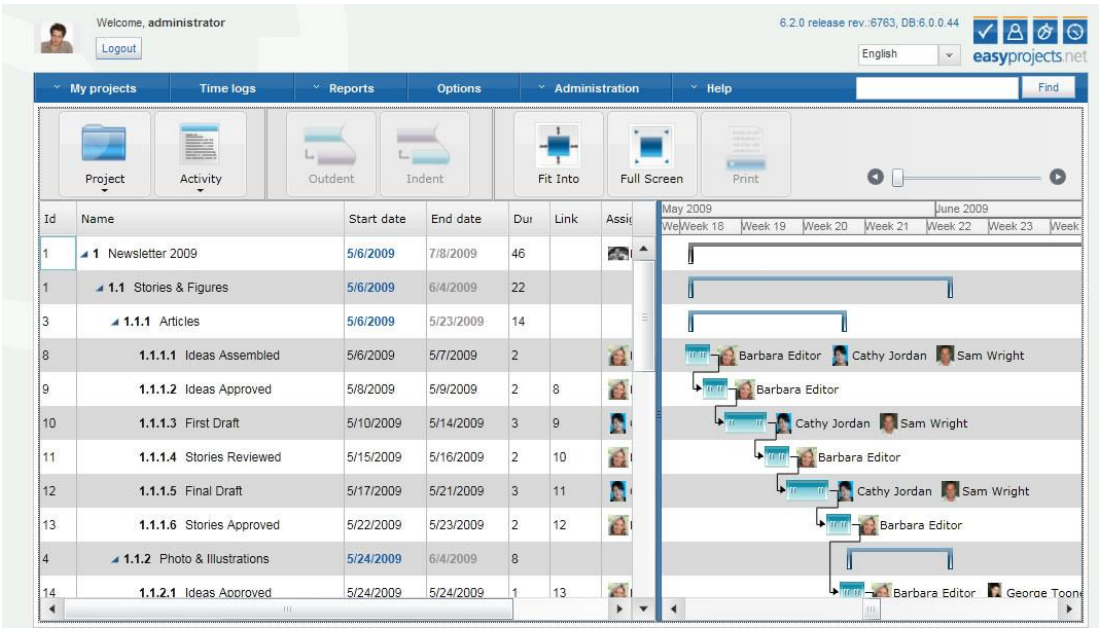

Рисунок 1.2 – Складання діаграми Ганта у Easy Projects

Особливістю даної системи є те, що це веб-додаток. У деяких випадках це недолік, оскільки необхідний доступ до Інтернету, з іншого боку – при наявності Інтернету можна виконати всі необхідні дії для планування.

Microsoft Project – програмний продукт компанії Microsoft, який призначений для планування ресурсів підприємства.

На рис. 1.3-1.5 представлені основні функціональні можливості системи [25].

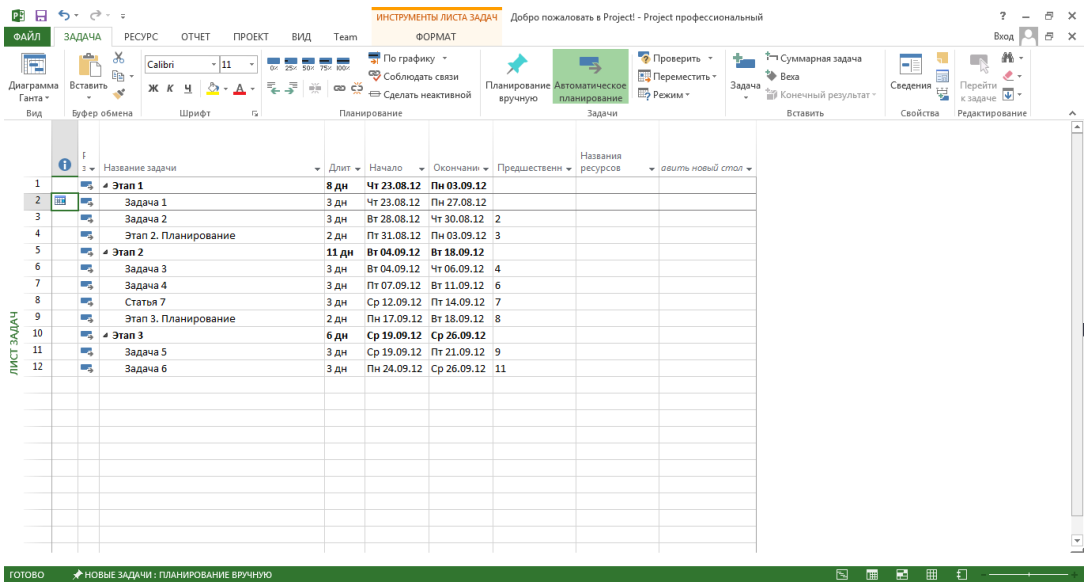

Рисунок 1.3 – Відображення списку задач в Microsoft Project

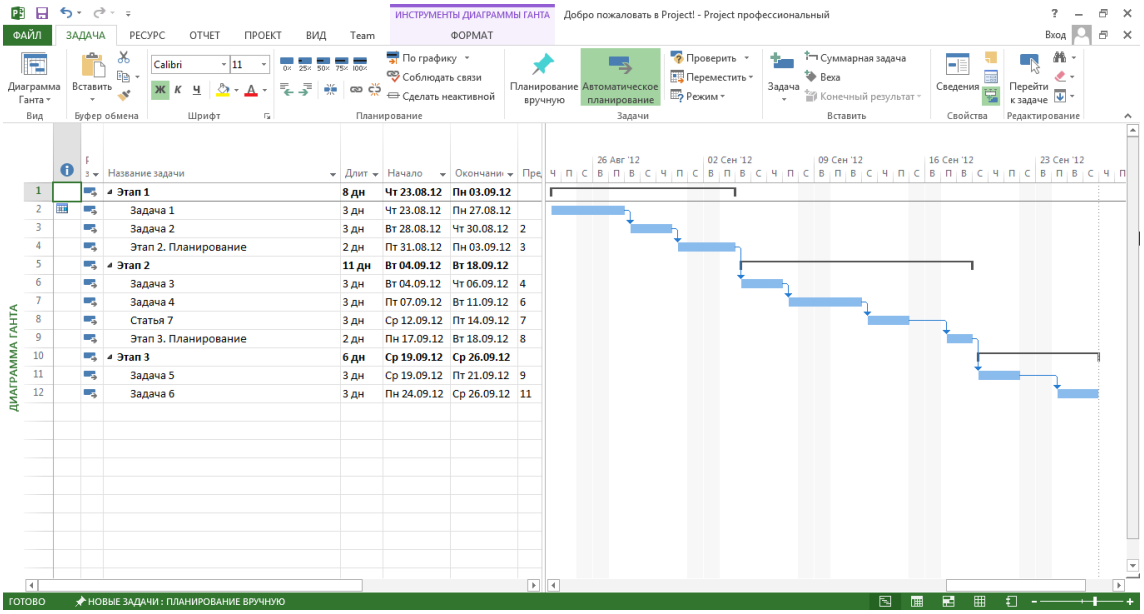

Рисунок 1.4 – Діаграма Ганта в Microsoft Project

На рис. 1.5 представлені роботи, які згруповані по етапах виконання.

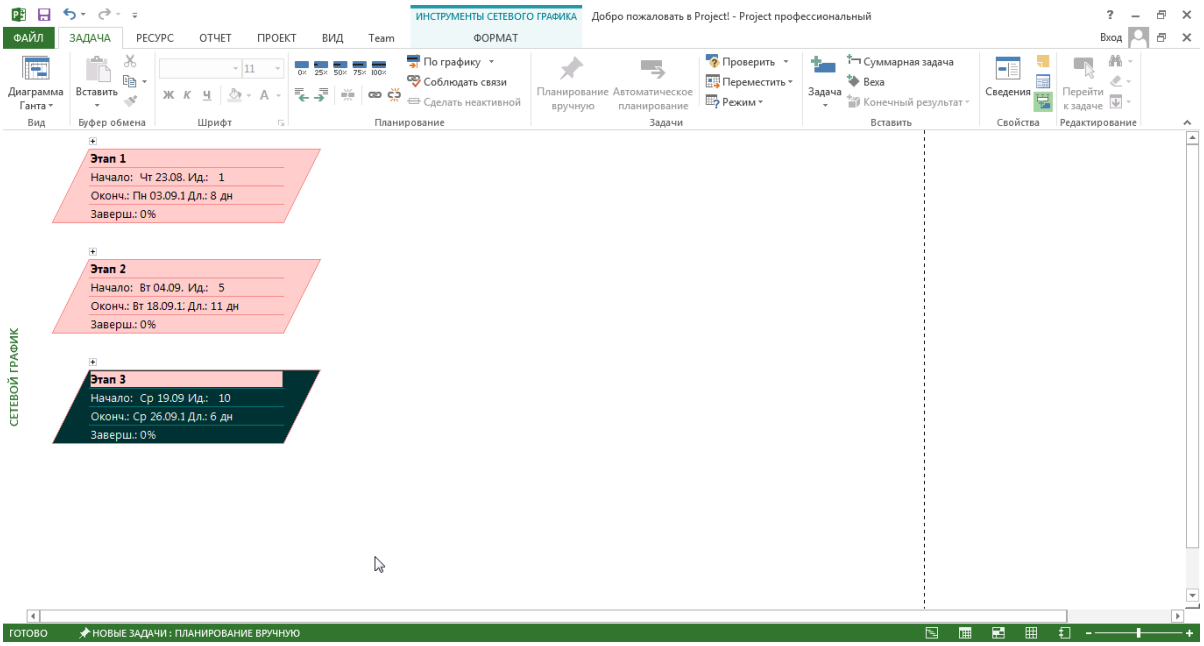

Рисунок 1.5 – Мережевий графік в Microsoft Project

При необхідності кожен етап можна розгорнути кнопкою «+» і переглянути всі роботи даного етапу. Хоча відображення мережевого графіка в Microsoft Project відрізняється від звичного виду робіт і подій, це не завдає ніяких незручностей [27].

PlanWIZARD – програма для автоматизації роботи планово-економічного відділу підприємства. Як і вище перераховані програмні продукти дозволяє занести планові роботи і побудувати діаграму Ганта.

На рис. 1.6 відображена основна функціональність даного продукту [28].

VisualData Планування виробництва робіт – програма для підвищення ефективності керування підприємством шляхом автоматизації процесів формування календарних планів виконання робіт з подальшим контролем їх реалізації, з використанням наочних форм представлення даних (див. рис. 1.7). Програма адресована керівникам і планово-виробничим відділам підприємств насамперед будівельної галузі, хоча може бути використана і в інших галузях економіки [29].

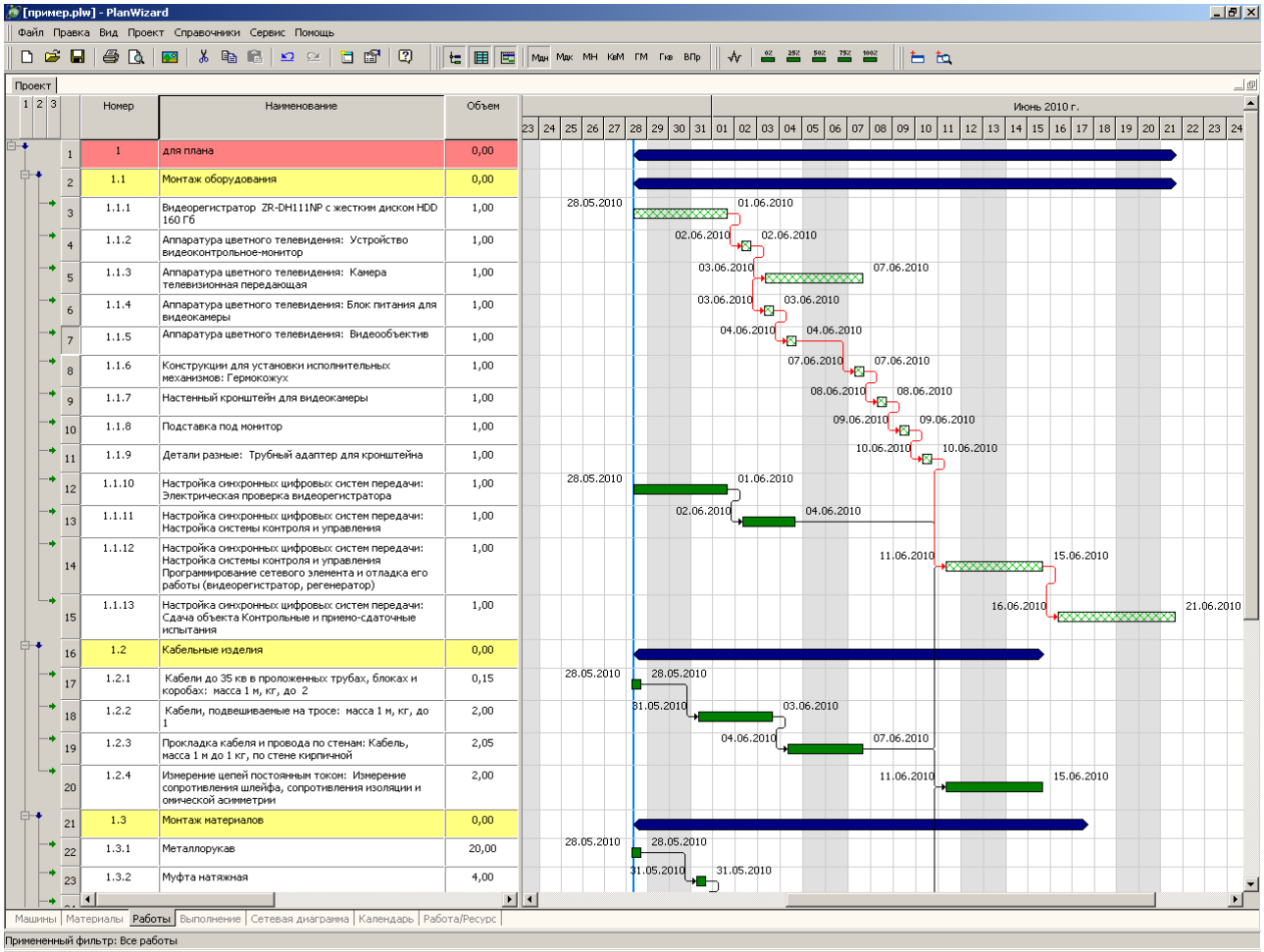

Рисунок 1.6 – Роботи и діаграма Ганта в PlanWIZARD

Дані пакети схожі за функціональними можливостями. Їх основне завдання – побудова графіка робіт у вигляді діаграми Ганта або мережевого графіка. Але всі вище перераховані пакети не вирішують головну задачу – оптимізація графіка виконання робіт з метою ефективного розподілу коштів. Майбутня система дозволить вирішити цю проблему.

Враховуючи всі переваги та недоліки існуючих систем, постає мета в проектуванні та розробці системи, основною перевагою якої буде мінімізація часу виконання проекту за рахунок перерозподілу робіт з урахуванням можливих об’ємів фінансування. В якості обмежень для задачі розподілу робіт обрано саме фінансове забезпечення, тому що воно напряму впливає на можливість використання відновлюваних і невідновлюваних ресурсів для виконання конкретною роботи.

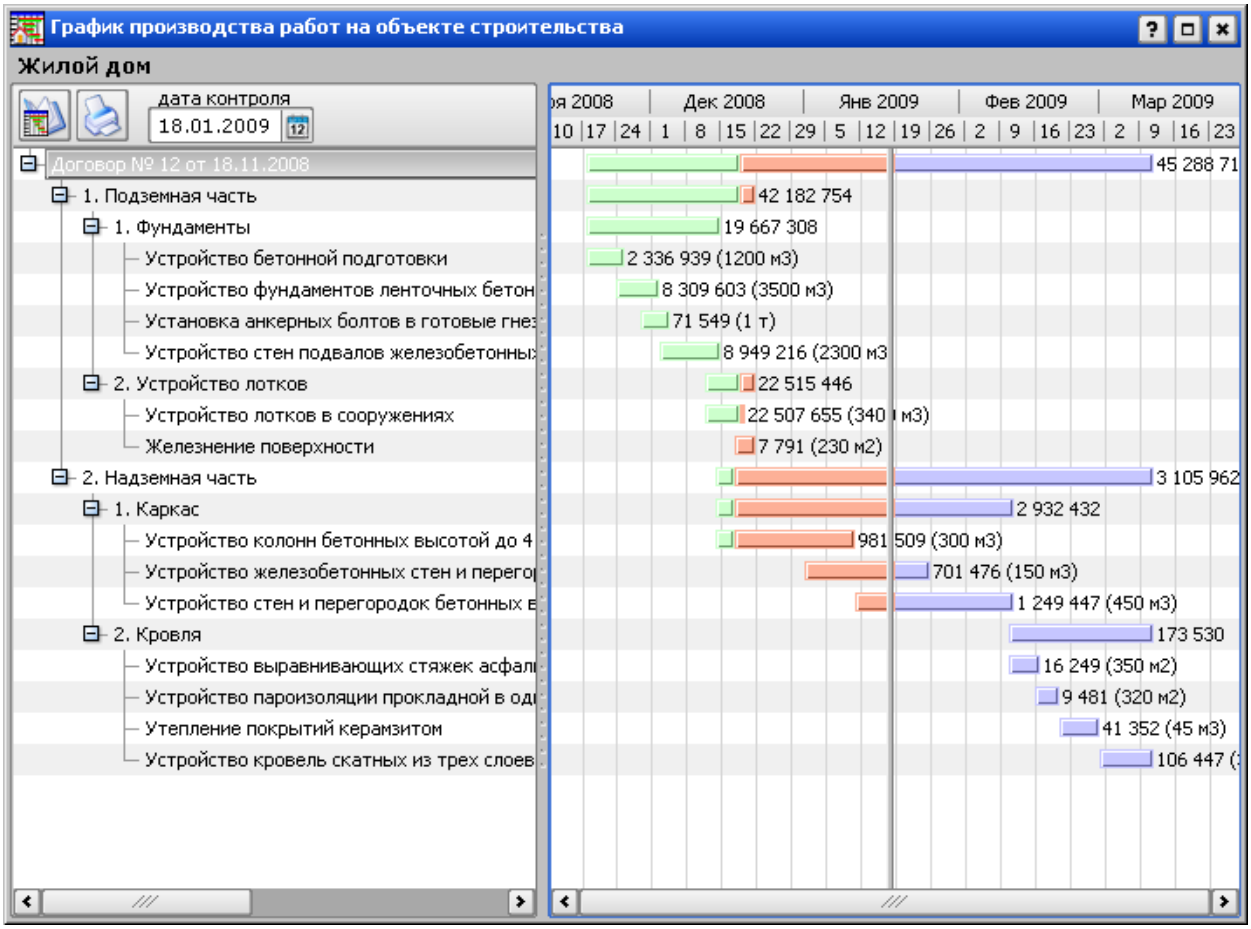

Рисунок 1.7 – Роботи и діаграма Ганта в VisualData

Наприклад, фінансова складова включає в себе використання витратних матеріалів для ремонту приміщення, а також оплату роботи працівників, які будуть робити ремонт.

В разі найму декількох бригад працівників фінансова складова також безпосередньо впливає на вартість виконання роботи.

Для досягнення поставленої мети потрібно:

– вибрати метод для вирішення задачі;

– розробити математичну модель поставленої задачі;

– адаптувати вибраний метод під розроблену модель;

– дослідити отримані результати роботи методу на контрольному прикладі.

Критеріями досягнення поставленої мети є:

– відкриття заданої кількості торгівельних точок до зазначеного терміну;

– бюджет проекту не повинен бути перевищений;

– терміни виконання проекту повинні бути мінімальні, але з урахуванням обмеження бюджету.

1.4 Висновки за розділом

В першому розділі було виконано наступне:

– розглянуто загальні відомості про календарне планування;

– розглянуто та проаналізовано основні методи, які застосовуються для календарного планування;

– серед розглянутих методів, що використовують для вирішення задачі календарного планування було обрано мурашиний алгоритм, не дивлячись на те, що його застосування не є універсальним, та вимагає певних модифікацій для вирішення конкретної задачі;

– виконано порівняльний аналіз існуючих сучасних аналогічних систем;

– було виділено основні переваги на недоліки існуючих систем;

– на основі недоліків була поставлена мета розробки підсистеми, виділено основні задачі для досягнення мети та сформульовано критерії, за якими буде досягнуто переваги перед існуючими системами.

Таким чином, задача календарного планування є актуальною, адже її вирішення та застосування в конкретній предметній галузі надасть інструмент ефективного планування будь-якого обсягу необхідних робіт.

РОЗДІЛ 2
РОЗРОБКА МОДЕЛЕЙ І АЛГОРИТМІВ

2.1 Загальний підхід до розв’язання задачі

Оскільки об’єктом дослідження у даній роботі є процес, який полягає у відкритті фіксованої кількості торгівельних точок, розглянемо можливі підході до його вирішення.

При найпростішому підході, коли необхідно відкрити n торгових точок, даний процес може проходити послідовно, тобто роботи з відкриття і-ої торгової точки будуть виконуватись тільки після виконання всіх робіт торговельної точки і-1 (див. рис. 2.1).

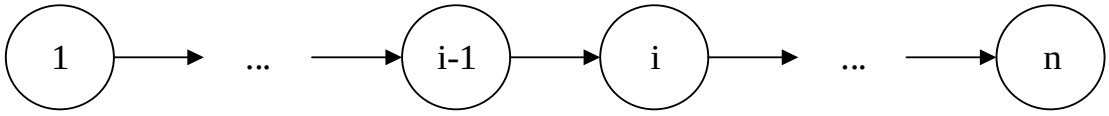

Рисунок 2.1 – Послідовний підхід до виконання проекту

Такий підхід не вигідний з кількох причин:

– з економічної точки зору, бо не гарантує оптимальне використання грошових коштів, які надходять періодично;

– час виконання проекту прямо пропорційний необхідному числу торгових точок.

Альтернативним підходом до вирішення даної проблеми є паралельне виконання робіт по кожній торговельній точці (див. рис. 2.2).

Слід зауважити, що для відображення паралельності організації торгових точок використані два позначення: нульова та N-а позиція. Це початок проекту і закінчення проекту відповідно.

Даний підхід є основою для вирішення поставленого задачі [4].

Для подальшої формалізації задачі детальніше розглянемо, що є роботою у даному процесі відкриття торгівельної точки.

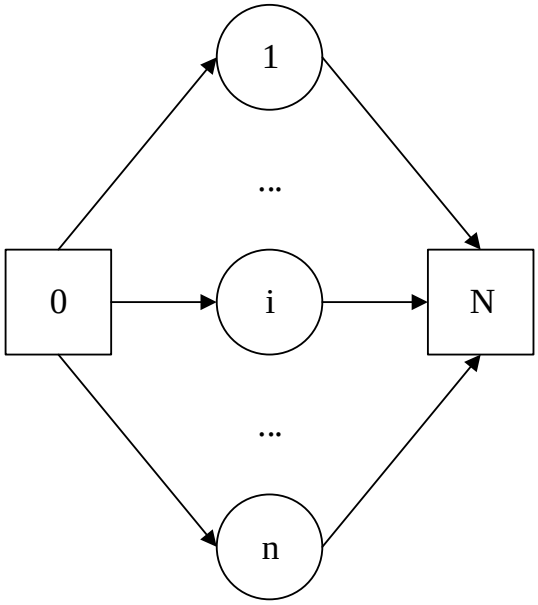

Рисунок 2.2 – Паралельний підхід до виконання проекту

Робота (work) – це дія, яку необхідно виконати для здачі в експлуатацію торгової точки (наприклад, виконати ремонт приміщення). Кожна робота має наступні характеристики:

– тривалість (duration);

– вартість (cost);

– множину робіт, що передують даній (previous works чи PW); позначимо її як {W1, W2, …, Wn}; ці роботи повинні обов’язково бути виконані перед початком даної (множина може бути порожньою);

– дата початку роботи (start date);

– дата завершення роботи (finish date).

Час виконання проекту при послідовному підході – це сума тривалостей всіх робіт, тому можна зробити висновок, що більш раціональним є паралельний підхід.

Слід врахувати особливості, перелічені нижче.

По-перше, кожна робота має множину попередніх робіт, що не дозволяє виконати всі роботи паралельно, виключаючи випадок, коли кожна робота має порожню множину попередників.

По-друге, при паралельному старті k робіт їх сумарна вартість не може перевищувати розмір грошових коштів, доступних в момент часу t (форм. (2.1)):

$$M_t \geq \sum_{i=1}^{k} c_i \tag{2.1}$$

де M_t – грошові кошти в момент t;

c_i – вартість виконання і-ої роботи.

По-третє, дві роботи W_i и W_j можуть виконуватись паралельно, якщо $W_i \notin PW_j$ и $W_j \notin PW_i$.

2.2 Математична постановка задачі

Математичну постановку задачі можна сформулювати наступним чином.

Проект P полягає у відкритті множини торгівельних точок T{T_1, T_2, …, T_n}, де n – кількість торгівельних точок.

Фінансування здійснюється через постійні проміжки часу t сумою M грошових одиниць.

Кожна торгівельна точка характеризується:

– множиною робот W_i {W_{i1}, W_{i2}, …, W_{im}}, де m – кількість робіт, необхідних для відкриття торгівельної точки T_i;

– дата здачі в експлуатацію D_i.

Кожна робота W має:

– тривалість d_k;

– вартість c_k;

– множину робіт, що передують даній PW {PW_1, PW_2, …, PW_k}, де k – кількість робіт, що передують;

– дату початку sd_k;

– дату закінчення fd_k.

Позначимо тривалість усього проекту як P_t, тоді функція, яку треба мінімізувати можна записати так (форм. (2.2)):

$$P_t = f(T, t, M) \to \min \tag{2.2}$$

2.3 Вибір методу для формування послідовності виконання робіт

У п. 1.2 були розглянуті методи вирішення поставленої задачі. Враховуючи переваги і недоліки усіх розглянутих методів, для реалізації було обрано синтез методу імітаційного моделювання та мурашиного алгоритму.

Імітаційне моделювання – метод дослідження, заснований на тому, що досліджувана система замінюється тією, що імітує дану.

З цією системою проводять експерименти (не вдаючись до експериментів на реальному об'єкті) і в результаті отримують інформацію про досліджувану систему.

Метод дозволяє імітувати виконання моделі бізнес-процесів так, як воно відбувалося б в дійсності, з урахуванням графіків робочого часу та зайнятості часових ресурсів і наявності необхідної кількості матеріальних ресурсів.

У результаті, можна оцінити реальний час виконання як одного процесу, так і заданої безлічі [31].

Одним з різновидів імітаційного моделювання є комп’ютерне моделювання.

Комп’ютерне моделювання – метод розв’язування задачі аналізу або синтезу складної системи, що ґрунтується на використанні її комп’ютерної моделі.

Сутність комп’ютерного моделювання полягає у відшуканні кількісних і якісних результатів із залученням наявної моделі. Якісні висновки, зроблені на підставі такого дослідження, дають змогу розкривати невідомі досі властивості складної системи: її структуру, динаміку розвитку, стійкість, цілісність тощо. Кількісні висновки мають переважно характер прогнозу

майбутніх чи пояснення минулих значень змінних, що характеризують систему.

Предметом комп'ютерного моделювання може бути економічна діяльність фірми, банку, промислового підприємства; інформаційно-обчислювальна мережа; технологічний процес; будь-який реальний об'єкт чи процес [32].

Процесом, який моделюється в системі, що підлягає розробці, є виконання робіт для відкриття торгівельних точок.

Розглянемо мурашиний алгоритм з точки зору його застосування до поставленої задачі.

У реальному світі, мурахи ходять у випадковому порядку і по знаходженню продовольства повертаються на свою колонію, прокладаючи феромонами стежки. Якщо інші мурахи знаходять такі стежки, вони, найімовірніше, підуть по ним. Замість того, щоб відстежувати ланцюжок, вони зміцнюють її при поверненні, якщо в кінцевому підсумку знаходять джерело живлення.

З часом феромонна стежка починає випаровуватися, тим самим зменшуючи свою привабливу силу. Чим більше часу потрібно для проходження шляху до мети й назад, тим сильніше випарується феромонна стежка. На короткому шляху, для порівняння, проходження буде більш швидким і як наслідок, щільність феромонів залишається високою. Випаровування феромонів також має властивість уникнення прагнення до локально-оптимального рішення.

Якби феромони не випаровується, то шлях, обраний першим, був би найпривабливішим. У цьому випадку, дослідження просторових рішень були б обмеженими.

Таким чином, коли один мураха знаходить (наприклад, короткий) шлях від колонії до джерела їжі, інші мурахи, швидше за все підуть цим шляхом, і позитивні відгуки в кінцевому підсумку призводять всіх мурах до одного, найкоротшому, шляху (див. рис. 2.3).

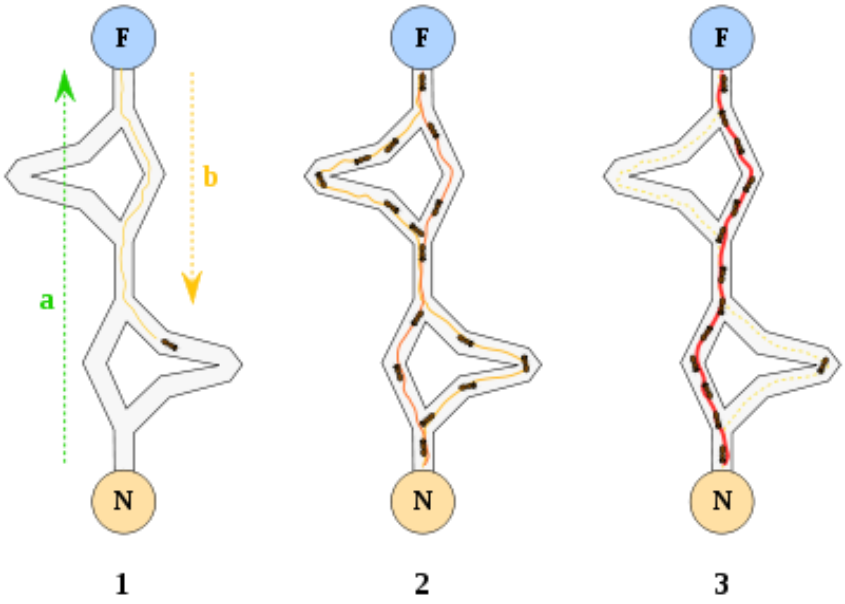

Рисунок 2.3 – Схема проходження мурах

Суть МА полягає в послідовному проходженні колонії мурах по «місцевості» і відкладанні феромонів на зворотному шлюху. Кожна мураха орієнтується, куди йти далі, за допомогою «досвіду» своїх попередників.

Описана модель виходить із спостереження за мурахами в процесі пошуку найкоротшого шляху від колонії до джерела живлення.

Перший мураха знаходить джерело їжі (F) будь-яким способом (a), а потім повертається до гнізда (N), залишивши за собою стежку з феромонів (b).

Потім мурахи вибирають один з чотирьох можливих шляхів, потім зміцнюють його і роблять привабливим.

Мурахи вибирають найкоротший маршрут, оскільки у більш довгих феромони сильніше випарувалися.

Зупинимося на питанні вибору вузла мурахою. Ймовірність вибору вузла виражена формулою (2.3):

$$p_{i,j} = \frac{\tau_{i,j}^{\alpha} \cdot \eta_{i,j}^{\beta}}{\sum \tau_{i,j}^{\alpha} \cdot \eta_{i,j}^{\beta}} \qquad (2.3)$$

де $-\ \tau_{i,j}$ кількість феромонів на ребрі i, j;

α– параметр, який контролює вплив $\tau_{i,j}$;

$\eta_{i,j}$ – привабливість ребра i, j, дорівнює $\frac{1}{d_{i,j}}$;

β– параметр, який контролює вплив $\eta_{i,j}$.

Формула (2.4) відображає випаровування феромонів на ребрі i, j:

$$\tau_{i,j} = (1-p)\tau_{i,j} + \Delta\tau_{i,j} \tag{2.4}$$

де $\tau_{i,j}$ – кількість феромонів на ребрі i, j;

p – швидкість випаровування феромону;

$\Delta\tau_{i,j}$ – кількість відкладеного феромона.

Зазвичай кількість феромону обчислюється за формулою (2.5):

$$\Delta\tau_{i,j}^{k} = \begin{cases} \frac{1}{L_k} \\ 0 \end{cases} \tag{2.5}$$

де k– мураха (якщо мураха k проходив по ребру i, j);

L_k – довжина шляху, пройдена мурахою k.

2.3 Опис алгоритму вирішення задачі

Алгоритм вирішення задачі базується на алгоритмі моделювання процесу виконання робіт для відкриття торгівельних точок. Для пошуку оптимальної послідовності робіт моделювання проводиться в декілька ітерацій. Наприкінці кожної ітерації отримане рішення порівнюється із найкращим. Для забезпечення коректного завершення роботи алгоритму кількість послідовних повторень оптимального рішення потрібно обмежити, наприклад m повторень. Максимальна кількість ітерацій також повинна бути обмежена – n ітерацій. Числа m та n визначаються дослідним шляхом.

Схема загального алгоритму вирішення задачі приведена на рис 2.4.

Приведемо опис процесу моделювання виконання робіт для однієї ітерації алгоритму. Вхідною інформацією для даного кроку алгоритму є інформація про усі торгівельні точки, які входять у проект, усі роботи для відкриття цих торгівельних точок. Для кожної роботи приведено перелік робіт, що передують даній.

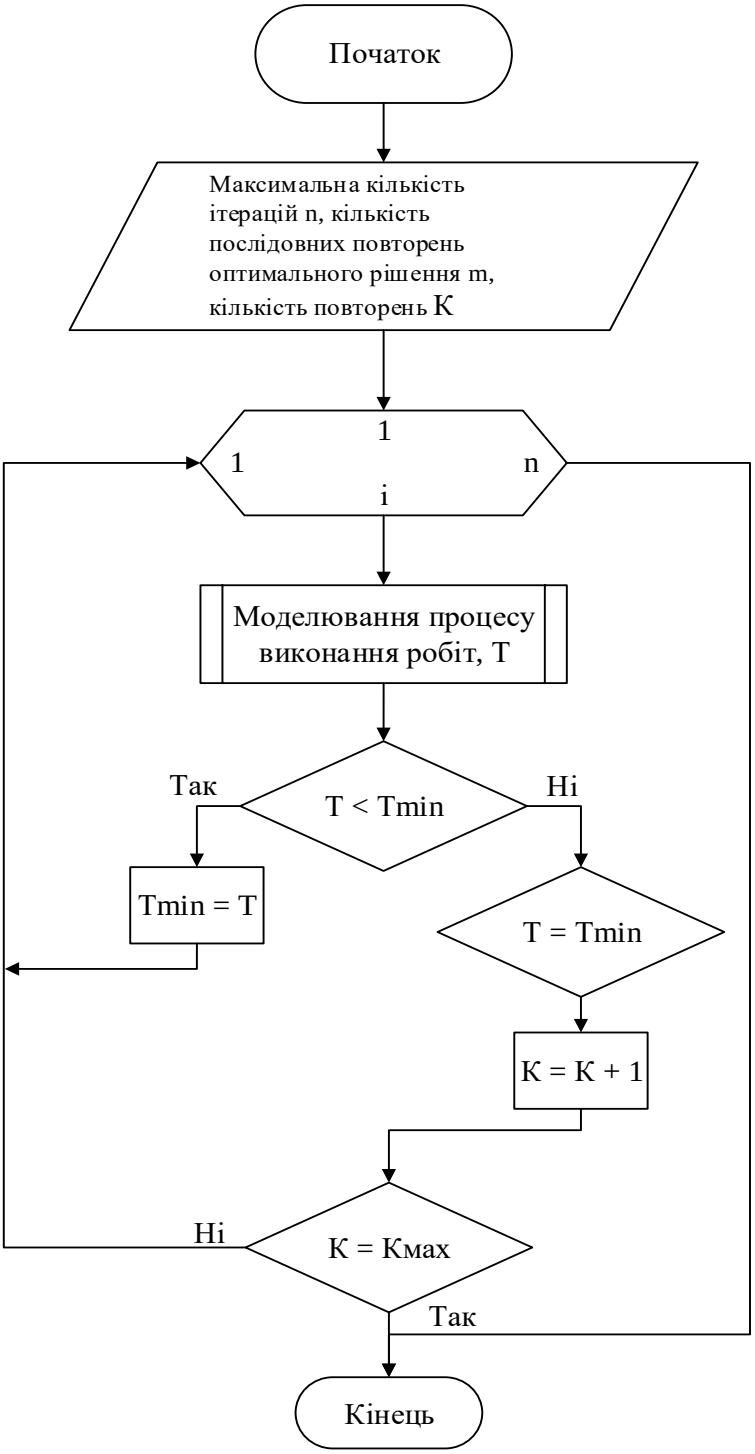

Рисунок 2.4 – Загальний алгоритм вирішення задачі

У даному прикладі немає прив’язки роботи до торгівельної точки, оскільки це ніяк не впливає на роботу алгоритму. Алгоритм ураховує лише залежності між роботами.

Вихідною інформацією алгоритму буде список робіт із зазначенням часу початку і часу закінчення. Алгоритм працює таким чином, що тривалість робіт є абстрактним числом, яке ніяк не пов’язано із датою та часом. Важливо тільки розуміти, що одиниці, в яких задано період інвестування, повинні зіставлятися із тривалістю робіт.

Наприклад, період вимірюється у днях, і тривалість повинна також бути задана у днях. Або період задано у годинах, тоді і тривалість повинна бути задана у годинах.

Після моделювання повного переліку робіт вираховується тривалість проекту загалом. Цей показник буде мінімізуватися у ході алгоритму. Слід враховувати, що кінцевим результатом роботи алгоритму може бути проект не з оптимальним часом виконання, а з близьким до оптимального. Алгоритм не гарантує оптимального рішення, тому що працює на засадах евристичних алгоритмів.

Алгоритм моделювання процесу виконання робіт складається з наступних кроків:

Крок 1. Оновлення поточного балансу (M). На цьому кроці, враховуючи поточний час, буде збільшуватись баланс на суму, яка дорівнює об’єму інвестицій.

Крок 2. Пошук переліку робіт, які можна почати виконувати у даний момент часу. Роботу можна виконувати, якщо усі роботи, що передують даній вже виконані, або дана робота не має попередніх. Також для прискорення алгоритму одразу треба відсіювати роботи, вартість яких перебільшує поточний баланс. Після даного кроку подальші розрахунки ведуться тільки з роботами, що можуть почати виконуватись у даний момент часу.

Крок 3. Розрахунок суми (S) робіт вирахуваних на кроці 2.

Крок 4. Порівняння поточного балансу (M) із сумою (S), яка була вирахувана на кроці 3. Якщо S менше або дорівнює M, то роботи, вирахувані на кроці 3 переходять на крок 5. Інакше, виконується ймовірнісний вибір робіт для подальшого виконання.

Крок 5. Початок виконання робіт, що були відібрані на кроці 4.

Крок 6. Пошук наступного моменту часу. Можливі два варіанти. Перший: наступний момент часу – це момент завершення виконання однієї чи декількох робіт. В цьому випадку треба оновити перелік виконаних робіт, що впливає на подальший вибір робіт, який виконується на кроці 2. У другому варіанті наступний момент часу – плановий перерахунок інвестицій.

Крок 7. Перевірка кількості невиконаних робіт. Якщо ця кількість дорівнює 0, алгоритм завершується, інакше потрібен перехід до кроку 1.

Блок-схема алгоритму приведена на рис. 2.5.

Блок-схема алгоритму виконання кроку 2 приведена на рис. 2.6.

Розглянемо детальніше крок 4 у разі, якщо поточний баланс менший сумарної вартості відібраних для виконання робіт.

Принцип відбору робіт для подальшого виконання запозичений із підходу, який використовується у мурашиному алгоритмі. Для кожної роботи існує така властивість, як оптимальний час виконання. На першій ітерації цей показник дорівнює нулю.

Після кожної ітерації для усього переліку робіт оновлюється оптимальний час виконання. Але тільки у разі, якщо поточна тривалість усього проекту менша, ніж попередня мінімальна тривалість.

Оптимальний час виконання роботи має прямий вплив на ймовірність вибору роботи у поточний момент часу (формули (2.7)-(2.9)):

$$d_i = |currentTime - optimalTime| \tag{2.7}$$

$$S_d = \sum_{i=1}^{n} d_i \tag{2.8}$$

$$p_i = \frac{S_d - d_i}{s_d} \tag{2.9}$$

d_i – різниця між часами;

currentTime – різниця між часами;

optimalTime – оптимальний час;

S_d – сума d_i;

n – кількість робіт;

p_i – ймовірність вибору роботи.

Використовуючи ймовірності робіт, розрахованих за формулою 8 необхідно змоделювати повну групу подій. Якщо робота і була вибрана, то вона автоматично видаляється із списку обраних робіт і подальше моделювання повної групи подій виконується без її участі. Тобто на кожному кроці вибору роботи ймовірності перераховуються. Процес вибору робіт закінчується коли поточний баланс вичерпано. Після цього процес пошуку робіт для виконання закінчений.

Такий підхід до відбору робіт дозволяє генерувати різноманітні варіанти робіт для виконання. Це відрізняє розроблений алгоритм від класичного алгоритму моделювання, в якому кожна робота має однакову ймовірність вибору.

Також даний підхід дозволяє запам’ятовувати оптимальну послідовність проходження робіт, та дозволяє відходити від нього заради пошуку нових, більш оптимальних послідовностей.

Блок-схема алгоритму приведена на рис. 2.7.

Розглянемо детальніше крок 6 (перший варіант). Для кожної роботи, що виконується, розраховується час завершення. Вираховується мінімальний час завершення, який у подальшому стане поточним часом для усього алгоритму моделювання. В залежності від робіт, які були виконані до цього часу, корегуються переліки робіт, що передують для усіх невиконаних робіт. Після цього оновлюється список виконаних робіт.

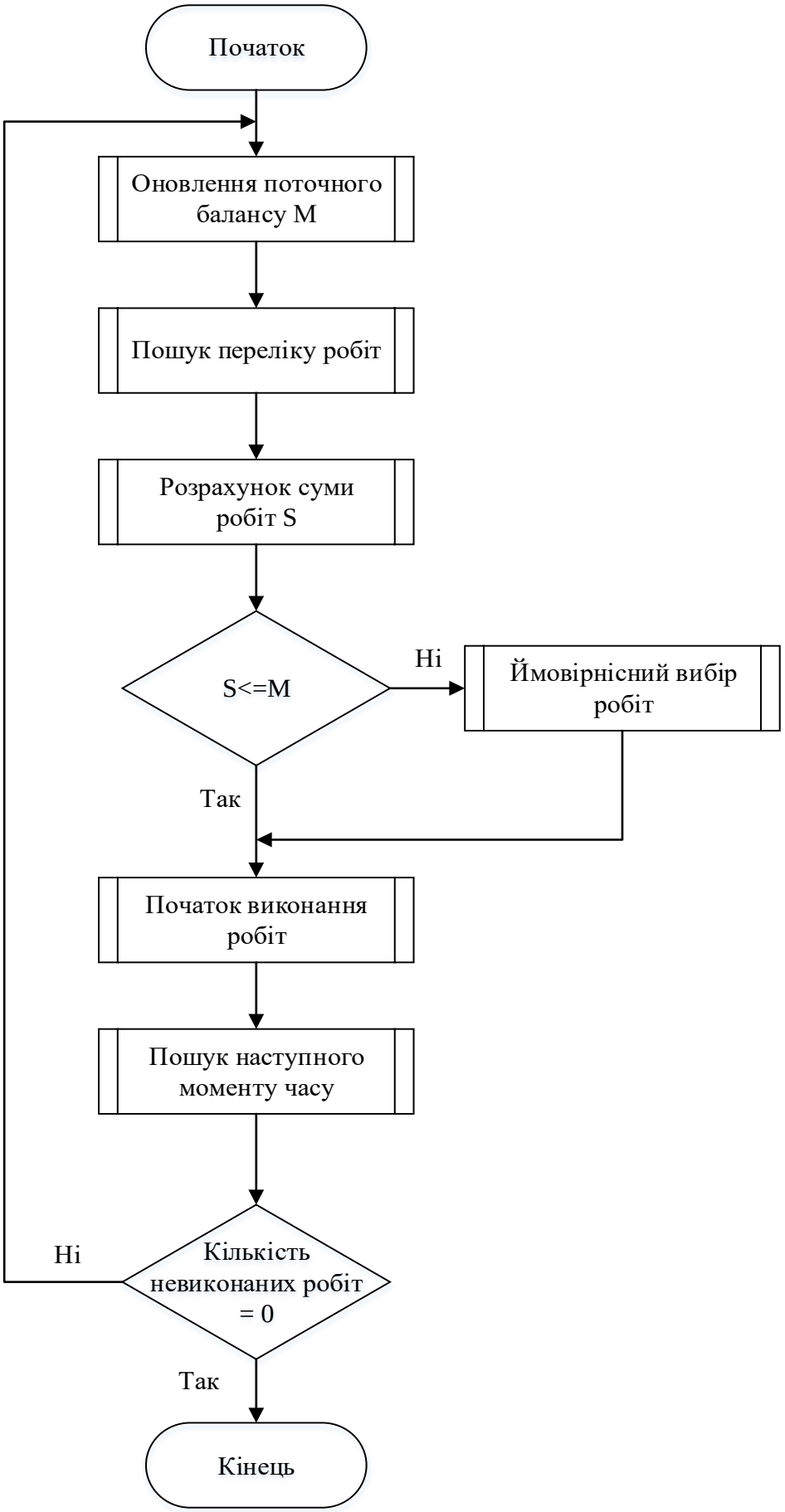

Рисунок 2.5 – Алгоритм моделювання процесу виконання робіт

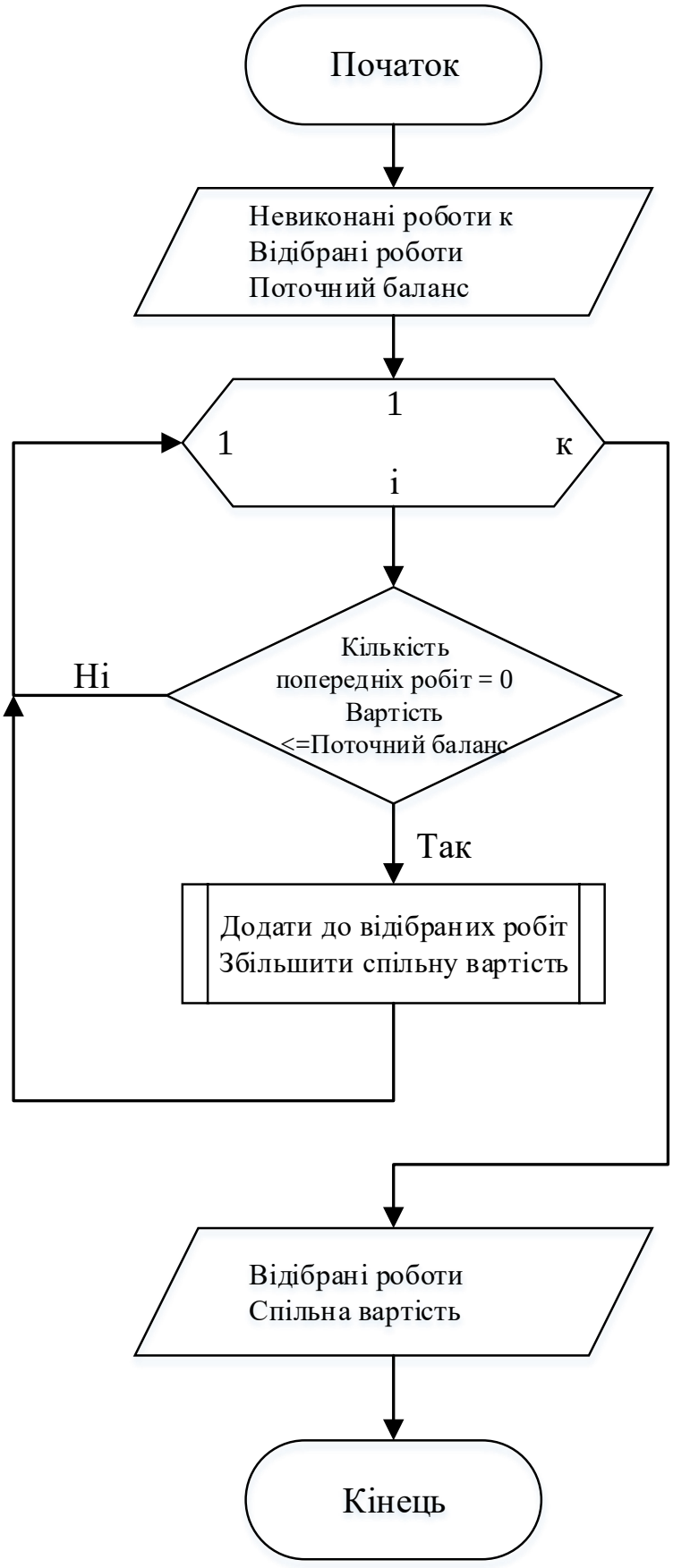

Рисунок 2.6 – Пошук переліку робіт, які можна почати виконувати у даний момент часу

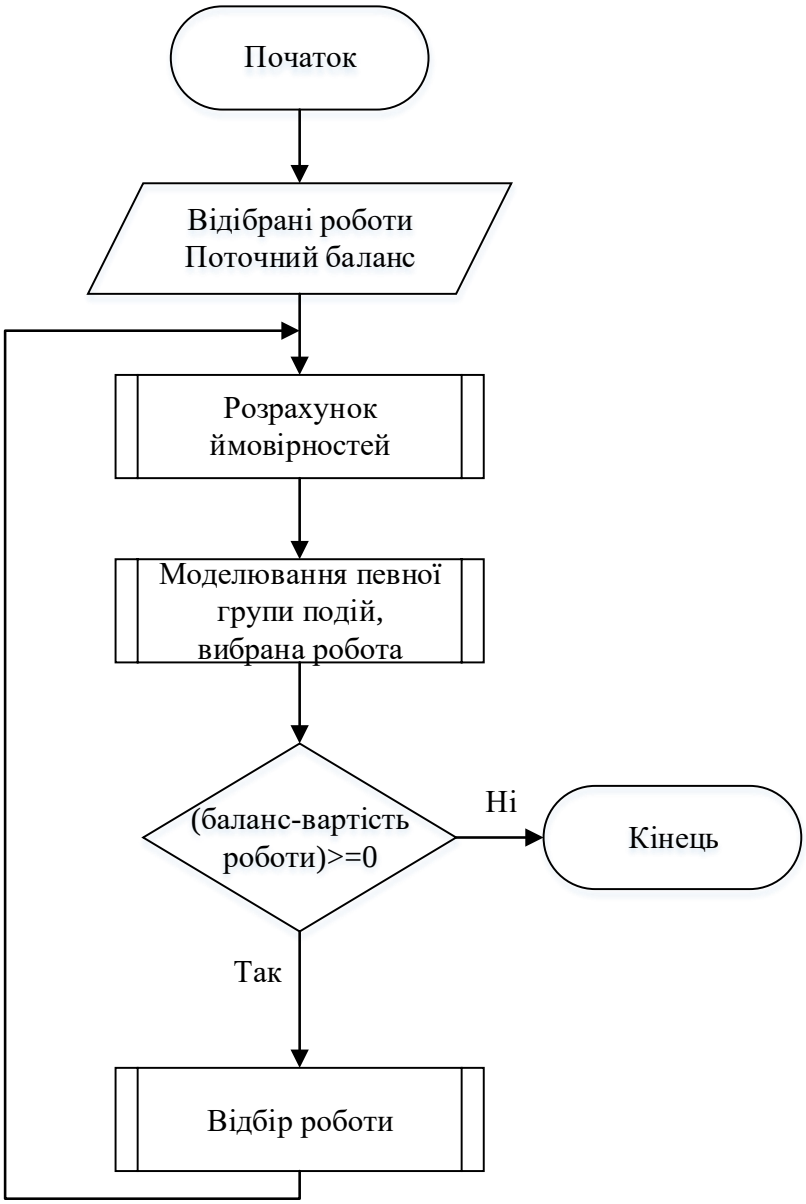

Рисунок 2.7 – Імовірнісний вибір робіт для подальшого виконання

2.4 Висновки за розділом

Таким чином, в розділі 2 було виконано наступне:

– обрано загальний підхід до розв’язання задачі формування послідовності виконання робіт;

– обґрунтовано вибір мурашиного алгоритму в якості такого, що буде використаний для розв’язанні задачі планування робіт при відкритті торгівельних точок;

– обґрунтовано доцільність застосування паралельного підходу в розподілі планових робіт;

– сформульована математична постановка задачі;

– наведено алгоритм роботи підсистеми;

– розроблено метод формування послідовності виконання робіт;

– розроблений метод був детально розглянутий і поділений на окремі кроки (у майбутньому функції).

РОЗДІЛ 3
ФУНКЦІОНАЛЬНА СТРУКТУРА ПІДСИСТЕМИ

3.1 Функції спроектованої підсистеми

Підсистема підтримки прийняття рішень при розподілі робіт з організації торгівельних точок призначена для менеджера проекту. Він встановлює строки виконання, визначає необхідний об’єм коштів для виконання усього проекту. Цю інформацію менеджер вносить до підсистеми і отримує на виході послідовність виконання робіт (рис. 3.1).

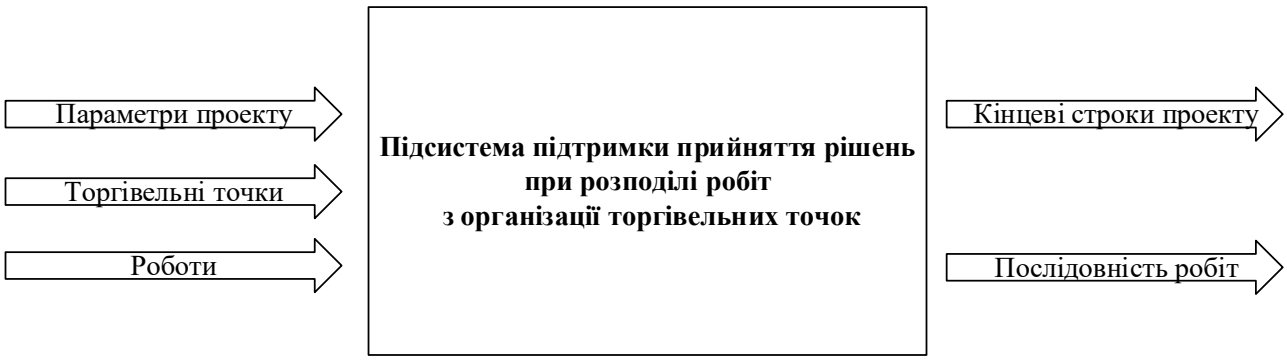

Рисунок 3.1 – Узагальнена схема роботи підсистеми

Виділимо основні функції підсистеми (див. табл. 3.1).

Таблиця 3.1 – Основні функції підсистеми

№	Назва	Призначення
1	2	3
1	Внесення інформації параметрів проекту	Підсистема надає можливість задати дату початку для проекту, період інвестування, суму інвестування
2	Внесення інформації про торгівельні точки	Підсистема надає можливість вводити інформацію про торгівельні точки: номер, назву.

Продовження таблиці 3.1

1	2	3
3	Внесення інформації про роботи по торгівельній точці	Підсистема надає можливість вносити роботи та їх параметри(назву, тривалість, вартість, роботи, що передують) вибраної торгівельній точці
4	Завантаження інформації про проект із файлу	Підсистема надає можливість вибрати файл із інформацією про проект і завантажити її до програми
5	Внесення інформації про проект	Узагальнює усі функції по внесення інформації про проект
6	Формування послідовності робіт	Підсистема формує із вхідної інформації послідовність робіт для виконання
7	Перегляд послідовності робіт	Підсистема відображає послідовність робіт, яка була попередньо розрахована
8	Збереження результатів у файл	Підсистема надає можливість зберегти отриману послідовність робіт у файл

Діаграма варіантів використання підсистеми зображена на рис. 3.2.

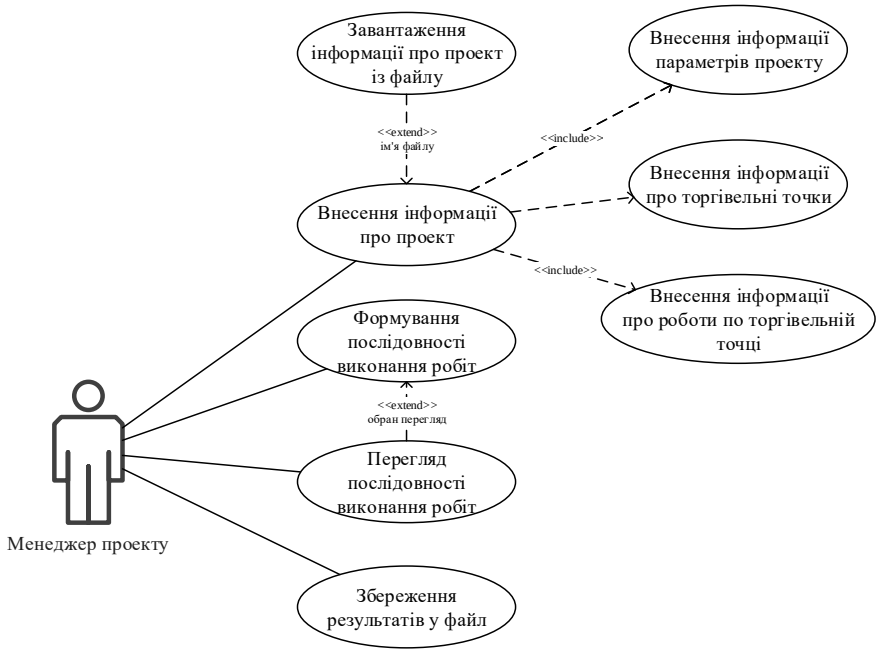

Рисунок 3.2 – Діаграма варіантів використання підсистеми

Функція «Внесення інформації про проект» включає декілька підфункцій, як зазначено на рис. 3.2. Для неї побудовано діаграму діяльності (рис. 3.3).

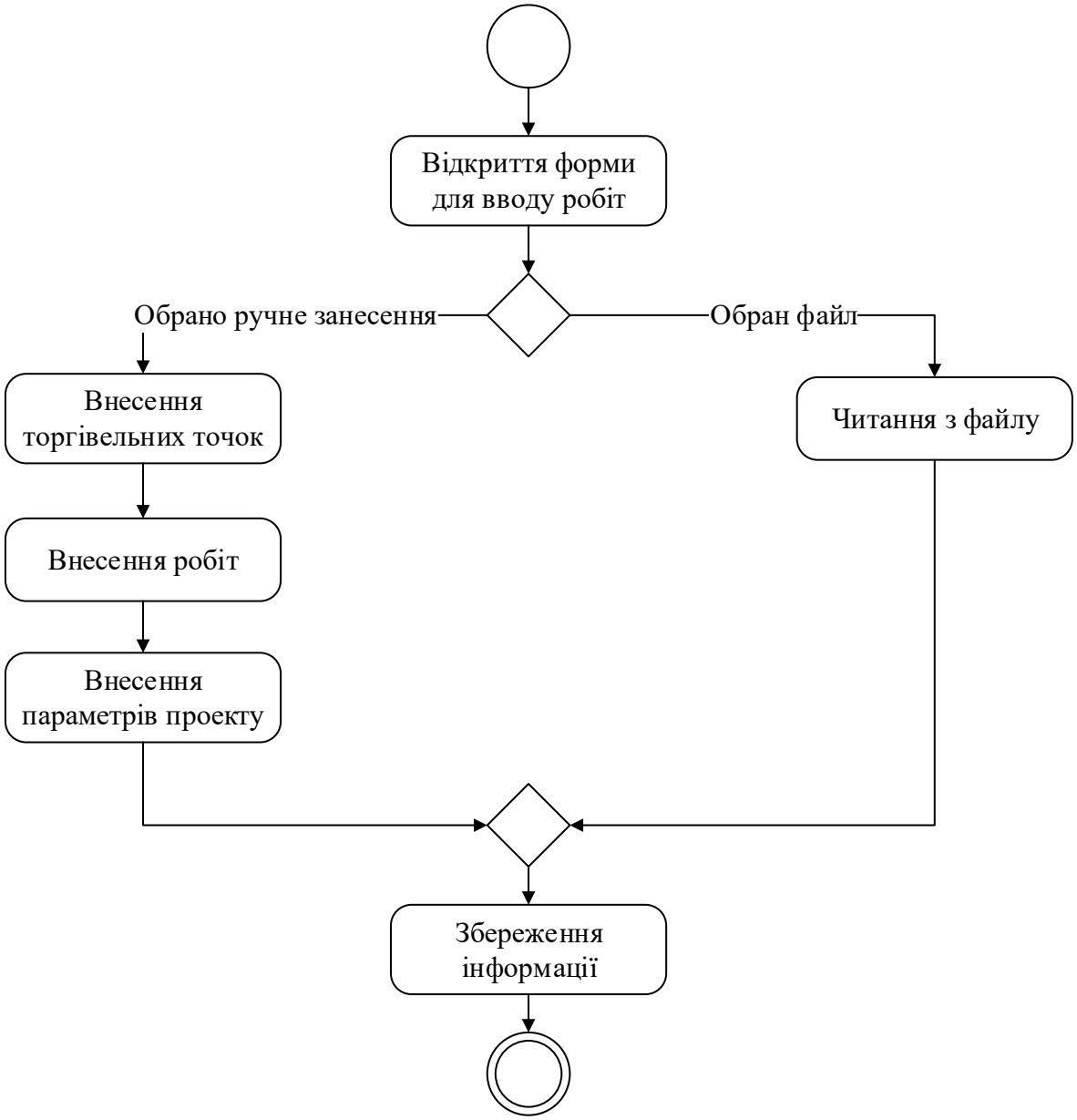

Рисунок 3.3 – Діаграма діяльності функція «Внесення інформації про проект»

Побудуємо діаграму діяльності для роботи підсистеми загалом (рис. 3.4).

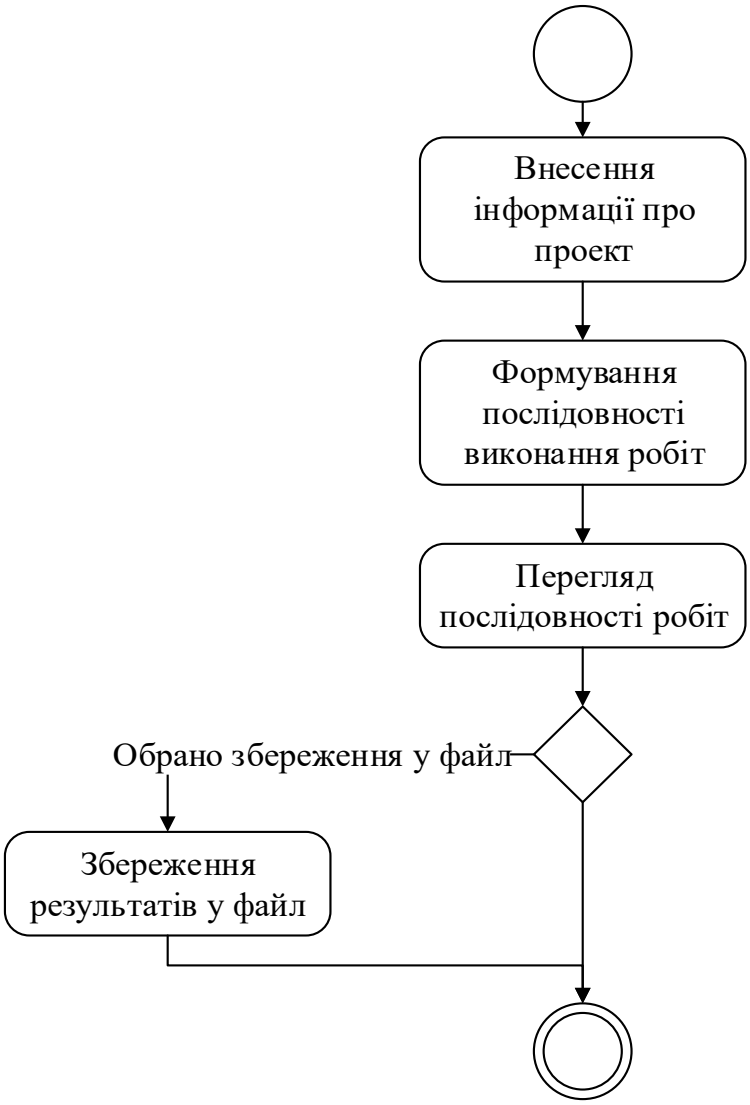

Рисунок 3.4 – Діаграма діяльності для роботи підсистеми

3.2 Виділення сутностей задачі про розподіл робіт

Згідно підрозділу 2.2 поставлену задачу можна представити за допомогою наступних сутностей (табл. 3.2).

Таблиця 3.2 – Сутності підсистеми

№	Назва сутності	Властивість
1	2	3
1	Проект	Торгівельні точки
		Період інвестування
		Сума інвестицій

Продовження таблиці 3.2

1	2	3
2	Торгівельна точка	Номер точки
		Роботи
		Дата здачі у експлуатацію
3	Робота	Назва роботи
		Тривалість
		Вартість
		Роботи, що передують
		Дата початку
		Дата закінчення

Для описаних сутностей побудуємо UML-діаграму для відображення зв’язку між сутностями підсистеми (див. рис. 3.5).

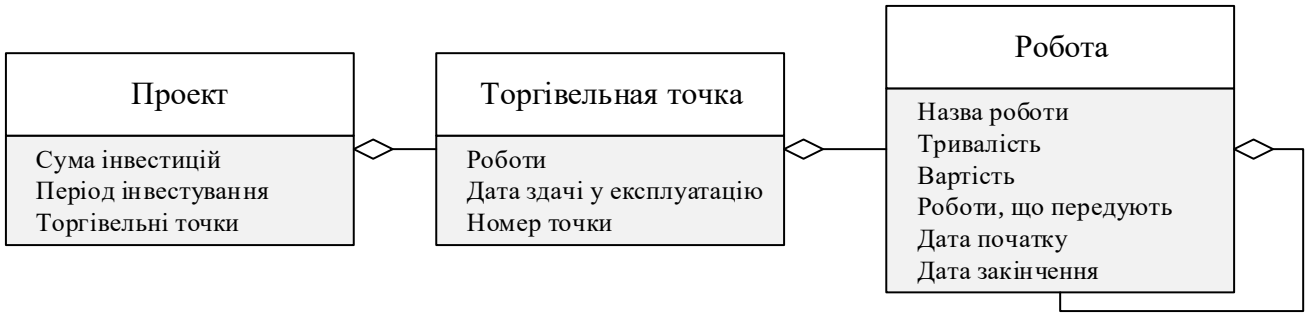

Рисунок 3.5 – Зв’язок між сутностями досліджуваної області

Для кожної сутності проекту необхідно мати обробник, який буде відповідати за роботу з нею (рис. 3.6).

Кожен обробник є нащадком загального обробника.

На рисунку 3.6 подане графічне відображення обробників підсистеми, що відповідають за її функціонування.

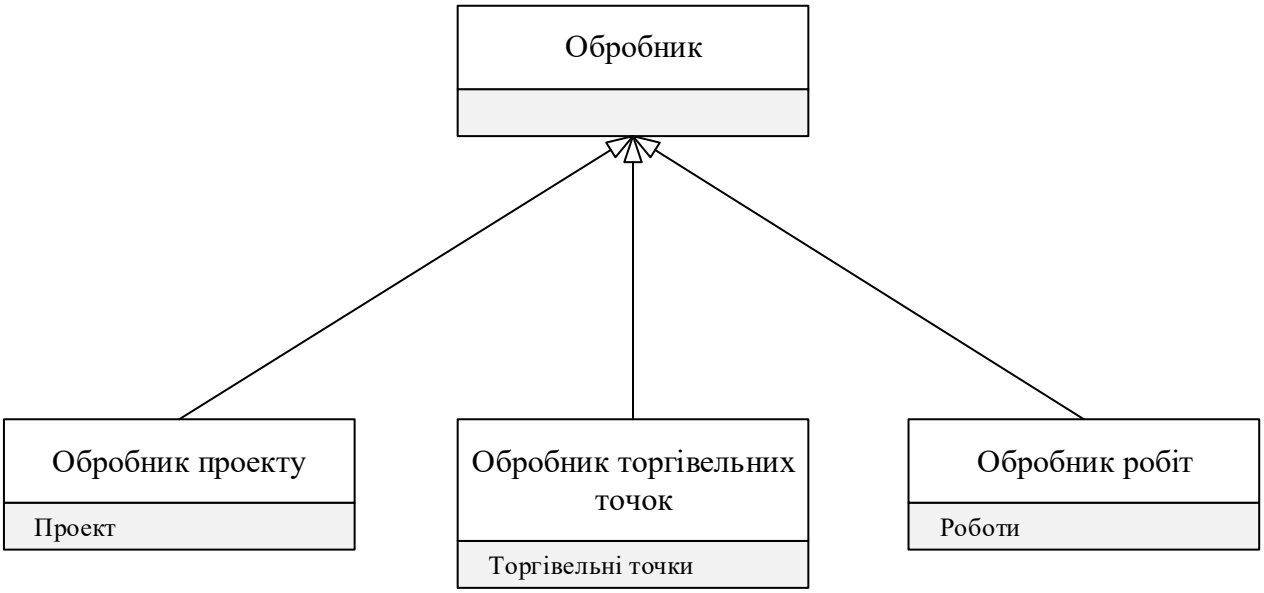

Рисунок 3.6 – Обробники підсистеми

З точки зору користування системою та розподілу прав доступу до неї, передбачено одного користувача, тобто менеджеру проектів, який вноситиме необхідну інформацію та отримуватиме всі необхідні для планування робіт дані.

3.3 Висновки за розділом

В розділі 3 було виконано наступне:

– наведено загальну функціональну схему роботи системи;

– виділено та наведено її основні функції;

– за допомогою інструменту Rational Rose побудовано діаграми використання, діаграми діяльності системи;

– з використанням об’єктно-орієнтованого підходу було визначено основні сутності підсистеми.

Таким чином, наведений набір функцій підсистеми надасть можливість користувачеві можливість вирішувати всі необхідні задачі при плануванні робіт.

РОЗДІЛ 4
РОЗРОБКА ІНФОРМАЦІЙНОГО І ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розробка інформаційного забезпечення

Інформаційне забезпечення включає сукупність даних, методи побудови бази даних (БД), а також проектних рішень за обсягами, розміщенню, формам організації інформації, що циркулює в інформаційній системі організації.

Зазвичай, система будується на основі спроектованої БД. Для поставленої задачі було прийняте рішення не використовувати БД, а забезпечувати підсистему інформацією за допомогою XML-файлів.

Для вхідного файлу задано наступний формат:

```
<project>
  <project_parameteres>
    <invest_period></invest_period>
    <invest_money></invest_money>
    <start_date></start_date>
    <duration></duration>
  </project_parameteres>
  <outlets>
    <outlet>
      <id>1</id>
      <name>АТБ</name>
      <finish_date>18.06.2021</finish_date>
    </outlet>
  </outlets>
  <works>
    <work>
      <id>1004</id>
      <name>Установка оборудования</name>
      <duration>7</duration>
      <start_date>11.01.2021</start_date>
      <finish_date>18.01.2021</finish_date>
      <cost>2</cost>
      <outlet_id>1</outlet_id>
```

```
      <previous_works>
        <id>1001</id>
        <id>1003</id>
      </previous_works>
    </work>
  </works>
</project>
```

Опис тегів наведено в табл. 4.1.

Таблиця 4.1 – Опис тегів вхідного XML-файлу

№	Назва тегу	Призначення
1	Project	Містить усі дані про проект
2	project_parameteres	Містить інформацію про параметри проекту
3	Outlets	Містить список торгівельних точок
4	Works	Містить список робіт
5	Outlet	Містить опис однієї торгівельної точки
6	Work	Містить опис однієї роботи

Формат вихідного файлу співпадає із форматом вхідного. Різниця полягає у тому, що у вихідному файлі заповнені поля, які розраховуються алгоритмом. У вхідному файлі ці поля не заповнені.

4.2 Розробка програмного забезпечення

Розроблене програмне забезпечення (ПЗ) спроектовано на основі об’єктно-орієнтованого підходу. Усі класи ПЗ можна розділити на декілька груп:

– сутності;
– форми;
– обробники;
– утілітні класи.

На рисунку 4.1 приведені групи класів ПЗ.

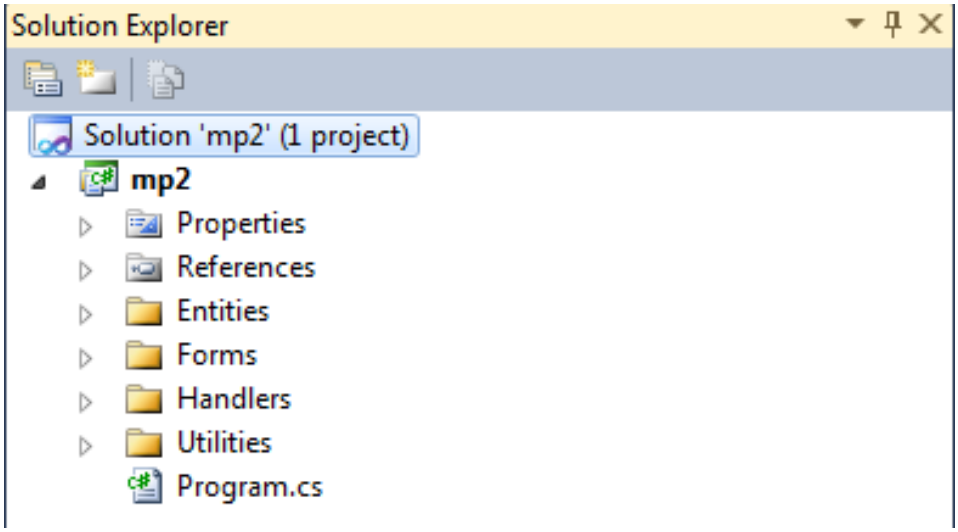

Рисунок 4.1 – Групи класів ПЗ

Розглянемо кожну групу класів окремо.

Опис класів групи «Сутності» приведений у табл. 4.2.

Таблиця 4.2 – Опис класів групи «Сутності»

№	Назва класу	Призначення
1	Outlet	Містить опис та логіку для торгівельних точок
2	Project	Містить опис та логіку для проекту
3	ProjectParameteres	Містить опис для параметрів проекту
4	SequenceWork	Містить опис та логіку для послідовності виконання робіт
5	StopCriterias	Містить опис для параметрів завершення алгоритму пошуку послідовності робіт
6	Work	Містить опис та логіку для робіт

Розроблені класи групи «Сутності» приведені на рис. 4.2.

У табл. 4.3 наведено опис основних властивостей та методів класу Outlet.

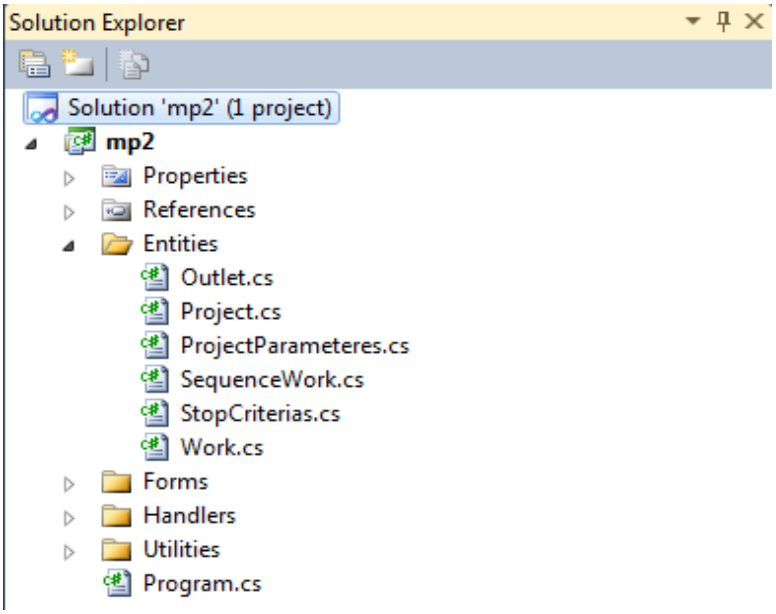

Рисунок 4.2 – Класи групи «Сутності»

Таблиця 4.3 – Опис основних властивостей та методів класу Outlet

№	Тип параметру	Назва	Призначення
1	Властивість	_id	Унікальний номер торгівельної точки
2	Властивість	_name	Назва торгівельної точки
3	Властивість	_finishDate	Дата здачі у експлуатацію роботи
4	Властивість	_works	Список робіт, необхідні для відкриття торгівельної точки
5	Метод	Outlet	Перевантажений конструктор для створення об’єкту класу
6	Метод	Init	Метод для задання початкових значень об’єкта
7	Метод	getWorks	Повертає усі роботи торгівельної точки
8	Метод	setWorks	Задає роботи для відкриття торгівельної точки
9	Метод	countFinishDate	Розраховує дату здачі у експлуатацію торгівельної точки

У табл. 4.4 приведений опис основних властивостей та методів класу Project.

Таблиця 4.4 – Опис основних властивостей та методів класу Project

№	Тип параметру	Назва	Призначення
1	Властивість	Outlets	Перелік торгівельних точок, що входять до проекту
2	Властивість	Works	Перелік усіх робіт для усіх торгівельних точок
3	Властивість	ProjectParameteres	Параметри проекту (період і об’єм інвестицій, дата початку проекту)
4	Метод	Project	Перевантажений конструктор для створення об’єкту класу
5	Метод	getWorks	Повертає список усіх робіт по усім торгівельним точкам
6	Метод	initOutletWorks	Розподіляє роботи в залежності від указаної торгівельної точки
7	Метод	updateOutletWorks	Оновлює параметри усіх робіт в залежності від указаної торгівельної точки

У табл. 4.5 приведений опис основних властивостей та методів класу ProjectParameteres.

У табл. 4.6 приведений опис основних властивостей та методів класу SequenceWork.

У табл. 4.7 приведений опис основних властивостей та методів класу StopCriterias.

У табл. 4.8 приведений опис основних властивостей та методів класу Work.

Таблиця 4.5 – Опис основних властивостей та методів класу ProjectParameteres

№	Тип параметру	Назва	Призначення
1	Властивість	_period	Період перерахунку коштів
2	Властивість	_money	Об’єм інвестицій за період
3	Властивість	_startDate	Дата початку проекту
4	Властивість	_duration	Тривалість проекту
5	Метод	ProjectParameteres	Перевантажений конструктор для створення об’єкту класу
6	Метод	getPeriod	Селектор для властивості _period
7	Метод	getMoney	Селектор для властивості _money
8	Метод	getStartDate	Селектор для властивості _startDate
9	Метод	getDuration	Селектор для властивості _duration
10	Метод	setPeriod	Модифікатор для властивості _period
11	Метод	setMoney	Модифікатор для властивості _money
12	Метод	setStartDate	Модифікатор для властивості _startDate
13	Метод	setDuration	Модифікатор для властивості _duration

Таблиця 4.6 – Опис основних властивостей та методів класу SequenceWork

№	Тип параметру	Назва	Призначення
1	2	3	4
1	Властивість	_works	Список робіт для виконання

Продовження таблиці 4.6

1	2	3	4
2	Властивість	_duration	Тривалість виконання усіх робіт
3	Метод	SequenceWork	Перевантажений конструктор для створення об’єкту класу
4	Метод	setDuration	Модифікатор для властивості _duration
5	Метод	setWorks	Модифікатор для властивості _works
6	Метод	getDuration	Селектор для властивості _duration
7	Метод	getWorks	Селектор для властивості _works
8	Метод	setDateTime	Дату початку для кожної роботи

Таблиця 4.7 – Опис основних властивостей та методів класу StopCriterias

№	Тип параметру	Назва	Призначення
1	Властивість	_maxIterationCount	Максимальна кількість ітерацій алгоритму
2	Властивість	_maxLoopCount	Максимальна кількість повторень найкращого результату

Таблиця 4.8 – Опис основних властивостей та методів класу Work

№	Тип параметру	Назва	Призначення
1	2	3	4
1	Властивість	_id	Унікальний номер роботи
2	Властивість	_name	Назва роботи
3	Властивість	_duration	Тривалість роботи
4	Властивість	_cost	Вартість роботи
5	Властивість	_outlet	Торгівельна точка, до якої відноситься робота

Продовження таблиці 4.8

1	2	3	4
6	Властивість	_previuosWorks	Роботи, що передують даній
7	Властивість	_startDateTime	Дата початку роботи
8	Властивість	_finishDateTime	Дата завершення роботи
9	Метод	Work	Перевантажений конструктор для створення об’єкту класу
10	Метод	getAvailableToStart	Повертає ознаку можливості виконання роботи у даний час
11	Метод	getTimeDifference	Вираховує різницю між оптимальним часом виконання роботи та поточним
12	Метод	Init	Ініціалізує роботу по заданим параметрам

Форми –вікна програми, які забезпечують користувачеві можливість внесення інформації та її відображення в зручному вигляді.

Опис класів групи «Форми» приведений у табл. 4.9.

Таблиця 4.9 – Опис класів групи «Форми»

№	Назва класу	Призначення
1	ApplicationForm	Відображає таблиці для вводу первинної інформації про торгівельні точки та проект загалом. Служить для забезпечення взаємо дії з користувачем.
2	ResultForm	Відображає результат роботи ПрД, тобто послідовність виконання робіт

Опис класів групи «Форми» приведений у табл. 4.10.

У табл. 4.11 наведено опис основних властивостей та методів класу ApplicationHandler.

Таблиця 4.10 – Опис класів групи «Форми»

№	Назва класу	Призначення
1	ApplicationHandler	Обробник для форм ApplicationForm та ResultForm; містить методи для маніпуляції із усіма сутностями системи
2	DependentWorkHandler	Обробник для робіт, що передують даній, та таблиці для їх відображення
3	Handler	Батьківський клас для обробників сутностей системи
4	OutletHandler	Обробник торгівельних точок та таблиці для їх відображення
5	WorkHandler	Обробник робіт для торгівельної точки та таблиці для їх відображення

Таблиця 4.11 – Опис основних властивостей та методів класу ApplicationHandler

№	Тип параметру	Назва	Призначення
1	2	3	4
1	Властивість	_project	Екземпляр класу Project, проект, для якого ведеться розрахунок
2	Властивість	_iLoader	Об’єкт, який відповідає за загрузку даних
3	Властивість	_currentFilename	Поточний файл для роботи із даними
4	Властивість	_iSaver	Об’єкт, який відповідає за збереження даних
5	Властивість	_outletHandler	Обробник торгівельних точок
6	Властивість	_workHandler	Обробник робіт
7	Метод	saveProject	Зберігає дані проекту у файл

Продовження таблиці 4.11

1	2	3	4
8	Властивість	_dependentWorkHandler	Обробник робіт, що передують даній
9	Властивість	_projectManager	Об’єкт, що забезпечує розрахунок послідовності робіт
10	Метод	ApplicationHandler	Конструктор для створення екземплярів класу
11	Метод	Init	Метод для ініціалізації об’єктів класу
12	Метод	StartDateTime OnValueChanged	Обробник події зміни значення початкової дати проекту
13	Метод	MoneyNumericUpDown OnValueChanged	Обробник події зміни значення об’єму інвестування проекту
14	Метод	PeriodNumericUpDown OnValueChanged	Обробник події зміни значення періоду інвестування проекту
15	Метод	getWork	Повертає роботу для поточної торгівельної точки за id
16	Метод	loadProject	Завантажує дані проекту із файлу
17	Метод	setProjectParameters	Переносить значення параметрів проекту у елементи керування
18	Метод	distributeWorks	Викликає метод для розподілу робіт
19	Метод	viewWorks	Викликає метод для відображення робіт
20	Метод	createNew	Створює новий проект
21	Метод	showResults	Відображає форму та заповняє її розрахованими значеннями

У табл. 4.13 наведено опис основних властивостей та методів класу Handler.

Таблиця 4.13 – Опис основних властивостей та методів класу Handler.

№	Тип параметру	Назва	Призначення
1	Властивість	_dataGridView	Таблиця, до якої будуть виводитися дані
2	Властивість	_bindingSource	Об’єкт для зв’язування таблиці та даних
3	Властивість	_headers	Масив значень заголовків колонок
4	Властивість	_blockedColumns	Масив заблокованих колонок
5	Властивість	_columnsAutoSizeMode	Масив властивостей колонок
6	Властивість	entityName	Ім’я сутності для відображення
7	Метод	Handler	Конструктор для створення об’єктів класу
8	Метод	initColumnHeaders	Ініціалізує заголовки колонок об’єктів
9	Метод	setSource	Задає джерело для відображення таблиці
10	Метод	blockColumns	Блокує колонки
11	Метод	setColumnAutoSizeMode	Задає властивості колонок
12	Метод	onSelectedChanged	Обробник зміни поточної сутності у таблиці

Як видно з опису методів класу, він використовується для реалізації більш комфортної роботи менеджера та дозволяє налаштовувати зовнішній вигляд ПрД більш зручним чином.

OutletHandler, WorkHandler, DependentWorkHandler – це похідні класи від Handler, які конкретизують логіку базового класу.

Утілітні класи – це класи, які потрібні для службових задач. Детальний опис цих класів наведений у табл. 4.14.

Таблиця 4.14 – Опис утілітних класів

№	Назва класу	Призначення
1	ControlTextInitializer	Дозволяє задати для елементу керування заголовок із строкових ресурсів
2	ProbabilitySimulator	Дозволяє моделювати одиничну подію та повну групу подій
3	ProjectManager	Містить логіку для побудови послідовності робіт
4	StringManager	Дозволяє вичитати строкові ресурси
5	XMLLoader	Дозволяє завантажити дані проекту із файлу
6	XMLSaver	Дозволяє зберегти дані проекту у файл

Серед наведених утіліт ControlTextInitializer, StringManager, XMLLoader, XMLSaver є такими, що дозволяють використовувати особливості роботи з файлами, строками, дозволяють робити парсинг XML документу за допомогою XPath, та реалізують зберігання інформації у вигляді вихідного документу. Вхідними параметрами цих методів є шлях до вхідного та вхідного файлів.

Особливо важливим є клас ProjectManager, адже він містить реалізацію мурашиного алгоритму та реалізацію всіх описаних у розділі 2 алгоритмів, а саме усю логіку для розрахунку послідовності робіт.

У табл. 4.15 наведено опис основних властивостей та методів класу ProjectManager.

Розглянемо детально основні методи цього класу.

Таблиця 4.15 – Опис основних властивостей та методів класу ProjectManager

№	Тип параметру	Назва	Призначення
1	Властивість	_probabilitySimulator	Об’єкт для моделювання випадкових подій
2	Метод	ProjectManager	Конструктор для створення об’єктів класу
3	Метод	checkWorkAvailable ToStart	Перевіряє, чи можна виконувати роботу у заданий момент часу
4	Метод	getAvailableWorks AndCountCost	Повертає список робіт, які можна виконувати у заданий момент часу, вираховує суму цих робіт
5	Метод	selectWorksToDo	Повертає список робіт, які почнуть виконуватись у поточний час
6	Метод	getProbabilities	Розраховує імовірність вибору для переданих робіт в залежності від поточного часу
7	Метод	manageWorks	Проводить одну ітерацію алгоритму пошуку оптимальної послідовності
8	Метод	manageProject	Вираховує найоптимальнішу послідовність виконання робіт проекту

Таким чином, клас ProjectManager є основним класом розробленої підсистеми, та реалізує всі спроєктовані алгоритми у вигляді методів класу. Як видно з опису методів, кожний етап алгоритму є окремим методом. Такий підхід робить розроблений ПрД більш зручним з точки зору додавання, редагування та будь-яких інших модифікацій коду.

4.3 Результати дослідження

Розглянемо процес відкриття трьох торгівельних точок з продажу спортивного інвентаря та спортивного харчування. Дамо назви трьом точкам «Планета Спорт», «Спортмаркет» та «ФітКервз+». Кожна торгівельна точка має різні початкові умови для введення в експлуатацію. Приведемо опис робіт для відкриття кожної торгівельної точки.

Щоб відкрити точку «Планета Спорт», потрібно виконати наступний перелік робіт (табл. 4.16).

Таблиця 4.16 – Перелік робіт для відкриття точки «Планета Спорт»

№	Назва роботи	Тривалість (дн.)	Вартість (у.о.)	Роботи, що передують
1	2	3	4	5
1	Пошук приміщень	30	1000	
2	Проектування структури точки	10	1500	Пошук приміщень
3	Ремонтні роботи	14	1600	Проектування структури точки
4	Пошук персоналу	20	500	
5	Навчання персоналу	20	1000	Пошук персоналу
6	Ліцензування торгівельної точки	30	4000	
7	Отримання дозвільних документів на торгівлю	15	2000	Ліцензування торгівельної точки
8	Формування товарних запасів	30	5000	Отримання дозвільних документів на торгівлю

Продовження таблиці 4.16

1	2	3	4	5
9	Замовлення торгівельного обладнання	10	7800	Отримання дозвільних документів на торгівлю
10	Замовлення ІТ-обладнання, оргтехніки, інсталяція програм	20	15000	Отримання дозвільних документів на торгівлю
11	Установка обладнання	2	500	Замовлення торг. обладнання; замовлення ІТ-обладнання, оргтехніки, …
12	Закупівля витратних матеріалів	4	400	Отримання дозвільних документів на торгівлю

Для введення в експлуатацію точки «Планета Спорт» необхідно знайти приміщення, найняти та навчити персонал, замовити та встановити обладнання.

Для «Планета Спорт» проводиться саме пошук, а не побудова приміщення, тому що ця точка буде розташована у конкретному місці, де побудова нової споруди неможлива чи не має сенсу, бо існує досить місць для її розташування.

Перелік робіт для відкриття точки «Спортмаркет» наведено у табл. 4.17.

Для здачі в експлуатацію «Спортмаркет» необхідно побудова нового приміщення, тому що в місці, де вона повинна знаходитись, немає можливості орендувати чи купити приміщення. Персонал для роботи в цій торговій точці вже знайдено та навчено.

Таблиця 4.17 – Перелік робіт для відкриття «Спортмаркет»

№	Назва роботи	Тривалість (дн.)	Вартість (у.о.)	Роботи, що передують
1	Побудова приміщення	120	5000	
2	Проектування структури точки	15	1500	
3	Внутрішні ремонтні роботи	14	1600	Проектування структури точки
4	Ліцензування торгівельної точки	30	4000	
5	Отримання дозвільних документів на торгівлю	15	2000	Ліцензування торгівельної точки
6	Формування товарних запасів	30	5000	Отримання дозвільних документів на торгівлю
7	Замовлення торгівельного обладнання	10	7800	Отримання дозвільних документів на торгівлю
8	Замовлення IT-обладнання, оргтехніки, інсталяція програм	20	15000	Отримання дозвільних документів на торгівлю
9	Установка обладнання	2	500	Замовлення торг. обладнання; Замовлення IT-обладнання, оргтехніки, …
10	Закупівля витратних матеріалів	4	400	Отримання дозвільних документів на торгівлю

Роботи, необхідні для відкриття «ФітКервз+», приведені у табл. 4.18.

Таблиця 4.18 – Перелік робіт для відкриття «ФітКервз+»

№	Назва роботи	Тривалість (дн.)	Вартість (у.о.)	Роботи, що передують
1	Ремонтні роботи	10	1200	
2	Ліцензування торгівельної точки	30	4000	
3	Отримання дозвільних документів на торгівлю	15	2000	Ліцензування торгівельної точки
4	Формування товарних запасів	30	5000	Отримання дозвільних документів на торгівлю
5	Замовлення торгівельного обладнання	10	7800	Отримання дозвільних документів на торгівлю
6	Замовлення IT-обладнання, оргтехніки, інсталяція програм	20	15000	Отримання дозвільних документів на торгівлю
7	Установка обладнання	2	500	Замовлення торг. обладнання; Замовлення IT-обладнання, оргтехніки, …
8	Закупівля витратних матеріалів	4	400	Отримання дозвільних документів на торгівлю

Для відкриття «ФітКервз+» не потрібно шукати приміщення, тому що воно було у розпорядженні власника всіх торгівельних точок, що

відкриваються чи було відкрито. Персонал, як у попередньому випадку, також знайдено та навчено заздалегідь.

Для виконання проекту менеджеру запропоновано наступні варіанти фінансування:

– фінансування проводиться з проміжком від 20 до 40 днів;

– розмір інвестицій має бути від 4000 до 10000 у.о.

Менеджеру проекту потрібно вибрати оптимальний проміжок фінансування та розмір інвестицій.

Дослідимо тривалість виконання процесу відкриття точок за допомогою розробленого програмного додатку.

Вхідні дані були введені за допомогою стартового вікна ПрД (рис. 4.3).

Для початку необхідно провести розрахунок тривалості проекту при мінімальному розмірі інвестицій і максимальному періоді фінансування.

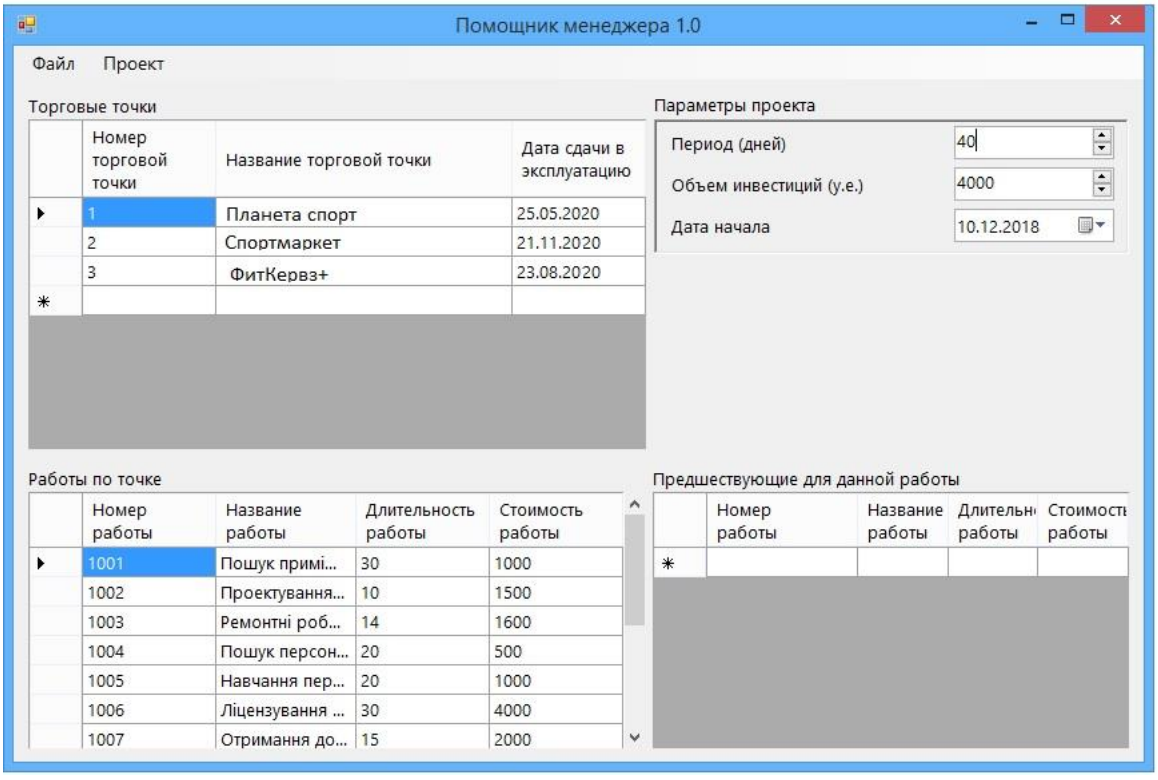

Рисунок 4.3 – Стартове вікно програмного додатку

Проведемо розрахунок та отримаємо тривалість проекту (рис. 4.4).

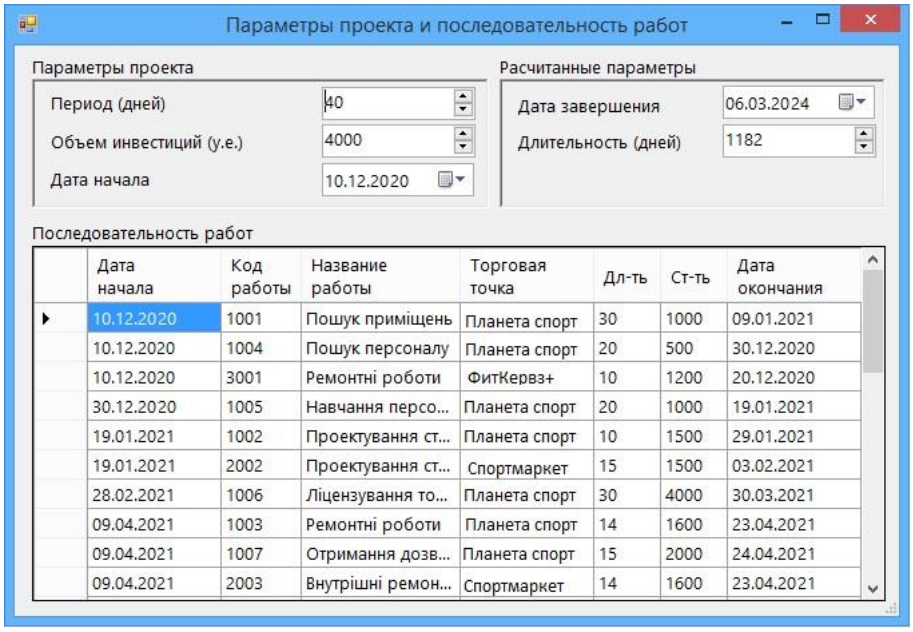

Рисунок 4.4 – Вікно із розрахованою послідовністю робіт та тривалістю проекту

Розрахунок показав, що при періоді у 40 днів і об’ємі інвестування 4000 у.о. буде отримано тривалість усього проекту 1182 дні. Також на рис. 4.4 відображено послідовність робіт для відкриття усіх торгівельних точок із зазначенням дати початку та дати завершення.

Подальші результати зведемо у таблицю (табл. 4.19).

Таблиця 4.19 – Тривалість проекту в залежності від умов фінансування

№	Період (дн.)	Об’єм інвестицій (у.о.)	Тривалість проекту (дн.)
1	2	3	4
1	20	10000	242
2	20	8000	302
3	30	10000	352
4	20	6000	402
5	30	8000	442
6	40	10000	462

Продовження таблиці 4.19

1	2	3	4
7	20	4000	502
8	40	8000	582
9	30	6000	592
10	40	6000	782
11	30	4000	892
12	40	4000	1182

Для більш глибокого дослідження проекту можна перевірити та розрахувати більшу кількість варіантів, якщо це необхідно.

Із усіх перевірених варіантів фінансування найбільш привабливим є звісно варіант № 1 (період 20 днів, об’єм інвестицій 10000 у.о.), але його недоліком є максимальна сума інвестицій. У разі зміни первісного плану проекту, торгівельна мережа буде мати великі збитки.

Для втілення у життя доцільно вибрати варіант № 4 (період 20 днів, об’єм інвестицій 6000 у.о.), тому що така схема фінансування дозволить легко корегувати хід виконання проекту і не завдасть дуже великих збитків мережі магазинів.

Слід зауважити, що дана підсистема є лише інструментом, який допомагає розподілити роботи і дозволяє розрахувати можливу тривалість проекту. Менеджер проекту може корегувати послідовність робіт. Але за допомогою підсистеми він зможе отримати декілька варіантів і обрати той, який задовольняє усім обмеженням.

4.4 Висновки за розділом

В розділі 4 було виконано наступне:

– обґрунтовано вибір способу зберігання початкових та вихідних даних та описано структуру вхідного та вихідного файлів;

– використовуючи об’єктно-орієнтований підхід, було створено 4 глобальні групи класів для реалізації програмного забезпечення системи;

– було наведено детальний опис кожної сутності всіх класів.

Крім того, в цьому розділі було детально описано методи класів, що відповідають за реалізацію основних алгоритмів, що були розроблені в розділі 2. Серед таких класів це:

– ProjectManager, який, власне, і реалізує всі необхідні алгоритми системи;

– Work – клас, в якому реалізована робота з переліком всіх видів робіт, що необхідно виконати для відкриття конкретної точки.

– StopCriterias – клас, в якому реалізована робота з обмеженнями, які дозволяють знайти оптимальне чи субоптимальне рішення.

Оскільки програмний додаток реалізовано у вигляді модулів, класів, методів, це робить ПрД більш гнучким для модифікацій, оскільки зміни, наприклад, класу, створеного для відображення інформації менеджера, не призведе до обов’язкових модифікацій в інших модулях. Обмін даними реалізовано в межах одного проекту завдяки зберіганню даних у вигляду списків (загальний список робіт, список термінів виконання, список послідовності робіт та ін.).

Проведене тестування показало, що всі вимоги до програмного забезпечення підсистеми реалізовано в повному обсязі, а саме – при плануванні відкрити нові торгівельні точки, менеджер отримав не лише оптимальне з точки зору використання грошей рішення, а також і перелік всіх можливих рішень зі всіма можливими об’ємами інвестицій та строками виконання.

ВИСНОВКИ

У кваліфікаційній роботі магістра було запропоноване вирішення актуальної задачі з розподілу робіт при плануванні відкриття нових торгівельних точок. При цьому отримано основні результати, які наведені нижче.

Розглянуто: загальні відомості про календарне планування; методи, які застосовуються для календарного планування; аналіз аналогічних систем: Easy Projects, Microsoft Project 2019, PlanWIZARD, VisualData Планування виробництва робіт. Також була поставлена мета: мінімізація часу виконання проекту за рахунок перерозподілу робіт з урахуванням можливих об’ємів фінансування. В якості обмежень для задачі розподілу робіт обрано саме фінансове забезпечення, тому що воно напряму впливає на можливість використання відновлюваних і невідновлюваних ресурсів для виконання конкретною роботи.

Обрано загальний підхід до розв’язання задачі формування послідовності виконання робіт, який базується на підході, який використовується у мурашиному алгоритмі; сформульована математична постановка задачі; розроблено метод формування послідовності виконання робіт; розроблений метод був детально розглянутий і поділений на окремі кроки.

Виділені основні функції підсистеми, побудовано діаграму варіантів використання, були наведені діаграми діяльності для підсистеми і окремих функцій підсистеми, виділені класи для структурування вхідної інформації.

Наведено опис розробленого програмного додатку. Наведено опис вхідного файлу, який має формат XML. Описані найбільш важливі класи підсистеми. Класи згруповані по функціям, які вони виконують.

Розглянуто процес відкриття фіксованої кількості торгівельних точок. Для розрахунку тривалості було використано розроблений програмний

додаток. Отримані результати біли проаналізовані та був обраний найкращій варіант виконання проекту.

Серед перспектив розвитку розробленої підсистеми є такі:

– поширення використання програмного додатку на всіх торгівельних точках;

– реалізація версії додатка на базі сервіс-орієнтованого підходу, що надасть програмному додатку можливість бути ще більш гнучким, ефективним у використанні та кросплатформеним у використанні;

– при збільшенні відкритих торгівельних точок перейти від використання XML файлів до повноцінної СКБД, що дасть можливість зберігати значно більше інформації по всіх точках;

– з точки зору реалізації алгоритму – перейти на сучасний підхід з використанням нейромережевого підходу з метою можливості покращення отриманих результатів за рахунок навчання мережі на базі реалізованих проєктів;

– реалізація функції відстежування процесу виконання робіт з метою вчасного редагування загального процесу відкриття точок;

– реалізація мобільного додатку, який дозволить отримувати інформацію в режимі реального часу.

У цілому ж можна зробити висновок, що запропонований спосіб розподілу робіт при з організації торгівельних точок є дієвим способом формування календарного плану проекту. Розроблений програмний додаток може використовуватись не лише для планування робіт для побудови та обладнання торгових приміщень, а і для проектів, які поділяються на окремі задачі та підзадачі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інформаційні технології в бізнесі. Частина 1: Навч. посіб. / [Шевчук І.Б., Старух А.І., Васьків О.М. та ін.]; за заг. ред. І.Б. Шевчук. Львів: Видавництво ННВК «АТБ», 2020. 455 с.

2. Івахненков С.В. Сучасні інформаційні технології управління підприємством та бухгалтерія: проблеми і виклики. Т. : ТНТУ, 2019. Том 55. № 5. С. 1–11. (Економіка та управління національним господарством).

3. Івахненков С.В. Інформаційні технології в організації бухгалтерського обліку: історія, теорія, перспективи. Наукове видання. – Житомир: АСА, 2001. – 416 с.

4. Іванова В.В. Планування і контроль на підприємстві: навч. посіб. / Суми: Університетська книга, 2015. 443 с.

5. Струтинська І. В. Інформаційні технології організації бізнесу – імператив інноваційного розвитку бізнес-структур / Ірина Струтинська // Галицький економічний вісник. Т. : ТНТУ, 2018. Том 55. № 2. С. 40–49. (Економіка та управління національним господарством).

6. Шведа Н. Особливості календарного планування проекту. «Інноваційний розвиток: стратегічний погляд у майбутнє» ТНТУ імені Івана Пулюя. Тернопіль, 2017. С. 69-70.

7. Зміст і завдання оперативно-календарного планування. URL: http://www.readbook.com.ua/book/31/800/ (дата звернення: 16.09.2021).

8. Оперативно-календарне планування на підприємстві. URL: http://www.dlearn.pu.if.ua/data/users/7588/import/%D0%A2%D0%B5%D0%BC%D0%B0%206 %20%D0%9F%D0%94%D0%9F.pdf (дата звернення: 18.09.2021).

9. Зміст та основні принципи планування діяльності підприємства URL:https://pidruchniki.com/84332/ekonomika/planuvannya_diyalnosti_pidpriyemstva (дата звернення: 18.09.2021).

10. Волошин Д.О., Жданова О.Г., Сперкач М.О. Задача складання календарного плану виконання робіт на підприємстві з мінімізацією сумарного випередження директивних термінів та максимізацією моменту початку виконання робіт паралельними пристроями. Інформатика та математичні методи в моделюванні. 2019. Том 9, №4. С. 304-314.

11. Павлов А.А., Чернов С.К. , Мисюра Е.Б. Модели и алгоритмы теории расписаний в задачах планирования и управления проектами. Пр. Одес. політехн. ун-ту. 2006. Вип.1. С. 150-159.

12. Мережевий графік проекту: параметри, побудова та оптимізація. URL: http://tpu.net.ua/uk/seo-merezevij-grafik-proektu-parametri-pobudova-ta-optimizacia.html (дата звернення: 20.09.2021).

13. Дудник І.М. Вступ до загальної теорії систем: Навчальний посібник/ Київ: Кондор, 2009. 205 с.

14. Литвинов А. Л. Теорія систем масового обслуговування : навч. посібник / Харків : ХНУМГ ім. О. М. Бекетова, 2018. 141 с.

15. Максимов О. В., Рашевський О. В. Елементи теорії масового обслуговування : Навчальний посібник / Кривий Ріг : Видавничий дім, 2004. 147 с.

16. Сеньо П. С. Теорія ймовірностей та математична статистика : підручник / Мін-во освіти і науки України, ЛНУ. – Київ: Центр навчальної літератури, 2004. 448 с.

17. Сеньо П. С. Випадкові процеси : підручник /Мін-во освіти і науки України, ЛНУ. – Львів : Компакт-ЛВ, 2006. 288 с.

18. Математичні методи дослідження операцій : підручник / Є. А. Лавров, Л. П. Перхун, В. В. Шендрик та ін.; Суми : Сумський державний університет, 2017. 212 с.

19. Оптимізаційні методи та моделі : навчальний посібник / Н. В. Бурєннікова, О. В. Зелінська, І. М. Ушкаленко, Ю. Ю. Бурєнніков. – Вінниця : ВНТУ, 2019. 121 с.

20. . Оптимізаційні методи і моделі: навч. Посібник / Вовк В. М., Зомчак Л. М. – Львів: ЛНУ імені Івана Франка, 2014. 360 с.

21. Долгова О. Э., Пересветов В.В. Составление расписаний с минимизацией суммарного запаздывания на одном приборе методом параллельных муравьиных колоний / Информатика, вычислительная техника и управление, 2012. №2. С. 45–52.

22. Управление проектами: учебное пособие / А. А. Дульзон; Национальный исследовательский Томский политехнический университет. – 3-е изд., перераб. и доп. – Томск : Изд-во Томского политехнического университета, 2010. – 334 с.

23. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы: Пер. с польск. И.Д. Рудинского. – 2-е изд., стереотип. – М.: Горячая линия – Телеком, 2013. – 384 с.

24. Ковалев М. Я. Модели и методы календарного планирования. Курс лекций. – Минск : БГУ, 2004. – 62 с.

25. Календарное планирование многостадийных производственных систем URL: https://cyberleninka.ru/article/n/kalendarnoe-planirovanie-mnogostadiynyhproizvodstvennyhsistem-svzaimozamenyaemym-oborudovaniem/viewer (01.06.2021).

26. Комп’ютерне моделювання систем та процесів. Методи обчислень. Частина 1 : навчальний посібник / Р. Н. Квєтний, І. В. Богач, О. Р. Бойко; за заг. ред. Р. Н. Квєтного. – Вінниця: ВНТУ, 2012. – 193 с.

27. Рачков М. Ю. Оптимальное управление детерминированными и стохастическими системами : Учебное пособие. – М. : МГИУ, 2005. – 136 с.

28. Кулюткин Ю. Н. Эвристические методы в структуре решений / Ю. Н. Кулюткин. – М. : Педагогика, 1970. – 232 с.

29. Климов Г. П. Стохастические системы обслуживания. 1966. – 279 с.

30. Зайченко Ю. П. Исследование операций : учебник. / К. : Слово, Киев, 2003. – 688 с.

31. Бурштейн И. М. Динамическое программирование в планировании. / И. М. Бурштейн – Экономика, 1968. – 127 с.

32. Духанов А.В. Имитационное моделирование сложных систем: курс лекций / А. В. Духанов, О. Н. Медведева; Владим. гос. ун-т. – Владимир : Изд-во Владим. гос. ун-та, 2010. – 115 с.

33. Ємець О.О., Парфьонова Т.О. Оцінювання допустимих множин розв'язків комбінаторної транспортної задачі на перестановках, що розв'язується методом гілок і меж / Наукові вісті НТУУ "КПІ". – Київ, 2010. – №1(69). – С.21-27.

34. Ібрагім, С. А. Розв'язування задач теорії розкладів генетичними алгоритмами / Вісник ЖДТУ – Житомир, 2004 - № 2. С. 180-185.

35. Тимофієва Н. К.; Гриценко В. І. Розв'язання задачі планування з теорії розкладів методом структурно-алфавітного пошуку та гібридним алгоритмом / Управляющие системы и машины. 2011. С. 21-36.

36. Данильченко А.О., Кравченко С.М. Порівняння результатів експерименту складання розкладу процедур методами: генетичний алгоритм, мурашиний алгоритм та метод гілок і меж / Вісник ЖДТУ.Серія: Технічні науки – Житомир, 2015 - № 2. С. 159-166.

37. Авдусь А. В. Дослідження та використання алгоритмів еволюційного моделювання в інтелектуальних системах прийняття рішень : пояснювальна записка до атестаційної роботи здобувача вищої освіти на другому (магістерському) рівні, спеціальність 122 – Комп'ютерні науки / А. В. Авдусь ; М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. – Харків, 2020. – 139 с.

38. Быстрое и простое управление проектом с Easy Project. URL: https://www.easyproject.com/EasyProject/media/images/Easy_Project_User_Guide_RU.pdf (дата звернення 20.04.2021)

39. Сравнение сервисов Easy Projects и Юникор. URL: https://startpack.ru/compare/easy-projects/unicore (дата звернення 22.04.2021)

40. Портал по Microsoft Project URL: http://www.microsoftproject.ru (дата звернення 26.04.2021).

41. PlanWIZARD – программа для календарного планирования URL: http://www.wizardsoft.ru/product/planwizard (дата звернення 26.04.2021).

42. VisualData Планирование производства работ URL: http://www.visualdata.ru/planirovanie-proizvodstva-rabot/programmy/visualdata-planirovanie-proizvodstva-rabot (дата звернення 12.05.2021).

43. Приклад проекту в Microsoft Project URL: https://blog.ganttpro.com/ru/primer-proekta-microsoft-ms-project/ (дата звернення 05.05.2021).

44. Тимофієва Н. К. Критерії подібності динамічних задач комбінаторної оптимізації. Математичне та комп'ютерне моделювання. Серія: Фізико-математичні науки, 2019. № 19. С. 168-174.

45. UML для бізнес-моделювання: для чого потрібні діаграми процесів URL: https://evergreens.com.ua/ua/articles/uml-diagrams.html (дата звернення 12.10.2021)

46. Чередніченко, О. Ю., Грінченко, М. А., Василенко, А. В., Матвєєв, О. М. Метод пошуку та аналізу даних з Інтернет ресурсів для формування актуальних вимог до кандидатів. Вісник Національного технічного університету ХПІ. Серія: Стратегічне управління, управління портфелями, програмами та проектами, (1), 31-38.

47. Робота з XML. URL: https://helpx.adobe.com/ua/incopy/using/xml.html (дата звернення 10.10.2021)

48. Керниган Б., Ритчи Д. / Kernighan B., Ritchie D. Язык программирования C, 2-е издание./ Вильямс, 2009 - 292 с.

49. Ерік Фрімен, Елізабет Робсон Паттерни проєктування / Харків: Фабула, 2020. 672 с.

Додаток А

Зауваження нормоконтролера

Таблиця А.1 – Зауваження нормоконтролера

Сторінка	Позначення	Зміст зауваження

Додаток Б

Програмний код

Б.1 Програмний код класів групи «Сутності».

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using mp2.Properties;
namespace mp2 {
      public class Outlet {
            private string _id;
            private string _name;
            private DateTime _finishDate;
            private List<Work> _works;

            public Outlet() {
                  _id = "";
                  _name = "";
                  _works=  new List<Work>();
            }
            public Outlet(string id, string name) {
                  this.init(id, name);
            }
            public Outlet(string id, string name, DateTime date) {
                  this.init(id, name);
                  this._finishDate = date;
            }
            public override bool Equals(object obj) {
                  return this._id.Equals(((Outlet) obj)._id);
            }
            public void init(string outletId, string outletName) {
                  this._id = outletId;
                  this._name = outletName;
            }
            public string Id {
                  set {
                        _id = value;
                  }
                  get {
                        return _id;
                  }
            }
            public string Name {
                  set {
                        _name = value;
                  }
                  get {
                        return _name;
                  }
            }
            public DateTime FinishDate {
                  set {
                        _finishDate = value;
                  }
                  get {
                        return _finishDate;
```

```
                  }
            }
            public override string ToString() {
                  return _name;
            }
            public List<Work> getWorks() {
                  return _works;
            }
            public void setWorks(List<Work> list) {
                  _works = list;
            }
            public void countFinishDate() {
                  DateTime date = _works.Max(w => w.getFinishDate());
                  _finishDate = date;
            }
      }
}
using System;
using System.Collections.Generic;
using System.Drawing.Printing;
using System.Linq;
using System.Text;
namespace mp2 {
      public class Project {
            public List<Outlet> Outlets {
                  private set; get;
            }
            public List<Work> Works {
                  private set;
                  get;
            }
            public ProjectParameteres ProjectParameteres {
                  private set; get;
            }
            public Project() {
                  Outlets = new List<Outlet>();
                  Works = new List<Work>();
                  ProjectParameteres = new ProjectParameteres();
            }
            public    Project(List<Outlet>    outlets,    List<Work>    works,
ProjectParameteres projectParameteres) {
                  Outlets = outlets;
                  this.initOutletWorks(Outlets, works);
                  Works = works;
                  ProjectParameteres = projectParameteres;
            }
            public List<Work> getWorks() {
                  List<Work> works = new List<Work>();
                  foreach (Outlet outlet in Outlets) {
                        works.AddRange(outlet.getWorks());
                  }
                  return works;
            }
            private  void  initOutletWorks(List<Outlet>  outlets,  List<Work>
works) {
                  foreach (Outlet outlet in outlets) {
                        outlet.setWorks(works.Where(w  =>  w.getOutlet().Id  ==
outlet.Id).ToList());
                  }
            }
            internal  void  updateOutletWorks(List<Outlet>  list,  List<Work>
works) {
                  foreach (Outlet outlet in list) {
                        List<Work> outletWorks = outlet.getWorks();
```

```
                        foreach (Work work in outletWorks) {
                              Work newWork = works.Find(w => w.Id == work.Id);
                              work.setStartDate(newWork.getStartDate());
                              work.setFinishDate(newWork.getFinishDate());
                              works.Remove(newWork);
                        }
                  }
            }
      }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace mp2 {
      public class ProjectParameteres {
            private decimal _period;
            private decimal _money;
            private DateTime _startDate;
            private int _duration;
            public ProjectParameteres() {
                  this._period = 0;
                  this._money = 0;
                  this._startDate = DateTime.Now;
            }
            public ProjectParameteres(decimal period, decimal money, DateTime
startDate) {
                  this._period = period;
                  this._money = money;
                  this._startDate = startDate;
            }
            public ProjectParameteres(decimal period, decimal money, DateTime
startDate, int duration) {
                  this._period = period;
                  this._money = money;
                  this._startDate = startDate;
                  this._duration = duration;
            }
            public decimal getPeriod() {
                  return _period;
            }
            public decimal getMoney() {
                  return _money;
            }
            public DateTime getStartDate() {
                  return _startDate;
            }
            public void setPeriod(decimal period) {
                  _period = period;
            }
            public void setMoney(decimal money) {
                  _money = money;
            }
            public void setStartDate(DateTime dateTime) {
                  _startDate = dateTime;
            }
            public void setDuration(int value) {
                  _duration = value;
            }
            public int getDuration() {
                  return _duration;
            }
      }
}
```

```
using System;
using System.Collections.Generic;
namespace mp2 {
      class SequenceWork {
            private List<Work> _works;
            private decimal _duration;
            public SequenceWork() {
                  _duration = Decimal.MaxValue;
                  _works = new List<Work>();
            }
            public Work this[int i] {
                  get { return _works[i]; }
            }
            public void setDuration(decimal duration) {
                  _duration = duration;
            }
            public void setSourceWorks(List<Work> works) {
                  _works = works;
            }
            public void setWorks(List<Work> finishedWorks) {
                  _works.Clear();
                  foreach (Work work in finishedWorks) {
                        Work w = (Work) work.Clone();
                        w.setBestTime(w.getStartTime());
                        _works.Add(w);
                  }
            }
            public decimal getDuration() {
                  return _duration;
            }
            public List<Work> getWorks() {
                  return _works;
            }
            public void setDateTime(DateTime dateTime) {
                  foreach (Work work in _works) {
                        work.setDateTime(dateTime);
                  }
            }
      }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace mp2 {
      class StopCriterias {
            private int _maxIterationCount;
            private int _maxLoopCount;

            public int MaxIterationCount {
                  get { return _maxIterationCount; }
                  set {
                        _maxIterationCount = value;
                        _maxLoopCount = _maxIterationCount/10;
                  }
            }
            public int MaxLoopCount {
                  get { return _maxLoopCount; }
            }
      }
}
using System;
using System.Collections.Generic;
namespace mp2 {
```

```
	public class Work : ICloneable {
		private static int guidSource = 0;
		private int guid;

		#region Entity properties
		private string _id;
		private string _name;
		private decimal _duration;
		private decimal _cost;
		private Outlet _outlet;
		private List<Work> _previuosWorks;
		private DateTime _startDateTime;
		private DateTime _finishDateTime;
		#endregion Entity properties
		#region Algorithm properties
		private decimal _bestTime;
		private decimal _startTime;
		private decimal _finishTime;
		#endregion Entity properties
		public Work() {
			guid = guidSource++;
			init("", "", 0, 0, new Outlet(), new List<Work>());
		}
		public Work(string id, string name, decimal duration, decimal cost,
Outlet outlet, List<Work> prevs) {
			guid = guidSource++;
			this.init(id, name, duration, cost, outlet, prevs);
		}
		public Work(string id, string name, decimal duration, decimal cost,
Outlet outlet, List<Work> prevs, DateTime startDate, DateTime finishDate) {
			guid = guidSource++;
			this.init(id, name, duration, cost, outlet, prevs);
			this._startDateTime = startDate;
			this._finishDateTime = finishDate;
		}
		public Work(string id,
			string name,
			decimal duration,
			decimal cost,
			Outlet outlet,
			List<Work> prevs,
			decimal startTime,
			decimal finishTime,
			decimal bestTime) : this(id, name, duration, cost, outlet,
prevs) {
			guid = guidSource++;
			_startTime = startTime;
			_finishTime = finishTime;
			_bestTime = bestTime;
		}
		public bool getAvailableToStart() {
			return _previuosWorks.Count == 0;
		}
		public string getId() {
			return _id;
		}
		public decimal getCost() {
			return _cost;
		}
		public object Clone() {
			List<Work> works = new List<Work>();
			foreach (Work work in _previuosWorks)
			{
				works.Add((Work)work.Clone());
```

```
                }
                return new Work(_id, _name, _duration, _cost, _outlet, works,
_startTime, _finishTime, _bestTime);
            }
            public void setDates(Work work) {
                _startDateTime = work._startDateTime;
                _finishDateTime = work._finishDateTime;
            }
            public override bool Equals(object obj) {
                if (obj.GetType() == typeof(Work)) {
                    Work w = (Work)obj;
                    if (w._id == this._id) {
                        return true;
                    }
                }
                return false;
            }
            public decimal getTimeDifference(decimal currentTime) {
                return Math.Abs(currentTime - _bestTime);
            }
            public void setStartTime(decimal currentTime) {
                _startTime = currentTime;
                _finishTime = _startTime + _duration;
            }
            public decimal getFinishTime() {
                return _finishTime;
            }
            public void removeIfContains(Work work) {
                if (_previuosWorks.Contains(work)) {
                    _previuosWorks.Remove(work);
                }
            }
            public void setBestTime(decimal bestTime) {
                _bestTime = bestTime;
            }
            public decimal getBestTime() {
                return _bestTime;
            }
            public decimal getStartTime() {
                return _startTime;
            }
            public  void  init(string  workId,  string  workName,  decimal
workDuration, decimal workCost, Outlet outlet, List<Work> list) {
                _id = workId;
                _name = workName;
                _duration = workDuration;
                _cost = workCost;
                _outlet = outlet;
                _previuosWorks = list;
                _startTime = 0;
                _bestTime = 0;
                _bestTime = 0;
            }
            public string Id {
                set {
                    _id = value;
                }
                get {
                    return _id;
                }
            }
            public string Name {
                set {
                    _name = value;
```

```
                }
                get {
                        return _name;
                }
        }

        public decimal Duration {
                set {
                        _duration = value;
                }
                get {
                        return _duration;
                }
        }
        public decimal Cost {
                set {
                        _cost = value;
                }
                get {
                        return _cost;
                }
        }
        public Outlet getOutlet()
        {
                return _outlet;
        }
        public void setOutlet(Outlet o)
        {
                _outlet = o;
        }
        public List<Work> getPreviousWorks()
        {
                return _previuosWorks;
        }
        public void setDateTime(DateTime dateTime) {
                _startDateTime = dateTime.AddDays((int) _bestTime);
                _finishDateTime = _startDateTime.AddDays((int) _duration);
        }
        public string[] getString() {
                List<string> list = new List<string>();
                list.Add(_startDateTime.ToShortDateString());
                list.Add(_id);
                list.Add(_name);
                list.Add(_outlet.Name);
                list.Add(_duration.ToString());
                list.Add(_cost.ToString());
                list.Add(_finishDateTime.ToShortDateString());
                return list.ToArray();
        }
        public DateTime getFinishDate() {
                return _finishDateTime;
        }
        public DateTime getStartDate() {
                return _startDateTime;
        }
        public void setStartDate(DateTime dateTime) {
                _startDateTime = dateTime;
        }
        public void setFinishDate(DateTime dateTime) {
                _finishDateTime = dateTime;
        }
    }
}
```

Б.2 Програмний код класів групи «Форми».

```
using System;
using System.Windows.Forms;
namespace mp2 {
      public partial class ApplicationForm : Form {
            private ApplicationHandler _applicationHandler;
            private ResultForm _resultForm;
            public ApplicationForm() {
                  InitializeComponent();
            }
            private void Form1_Load(object sender, EventArgs e) {
                  _applicationHandler                  =                  new
ApplicationHandler(outletsDataGridView,                    worksDataGridView,
dependentWorksDataGridView,
                        numericUpDown1, numericUpDown2, dateTimePicker1);
                  _resultForm = new ResultForm();
                  this.initMenuItemsCaptions(menuStrip1.Items);
                  this.initApplicationName();
                  this.initHeaders();
            }
            private void openToolStripMenuItem_Click(object sender, EventArgs
e) {
                  openFileDialog1.Filter                                    =
StringManager.getString("fileExtensionsFilter");
                  if (openFileDialog1.ShowDialog() == DialogResult.OK) {

      _applicationHandler.loadProject(openFileDialog1.FileName);
                  }
            }
            private void saveAsToolStripMenuItem_Click(object sender, EventArgs
e) {
                  saveFileDialog1.DefaultExt                                =
StringManager.getString("defaultFileExtension");
                  saveFileDialog1.FileName = String.Empty;
                  if (saveFileDialog1.ShowDialog() == DialogResult.OK) {
                  _applicationHandler.saveProject(saveFileDialog1.FileName);
                  }
            }
            private   void   distributeToolStripMenuItem_Click(object   sender,
EventArgs e) {
                  SequenceWork                  sequenceWork                  =
_applicationHandler.distributeWorks();
                  _applicationHandler.showResults(_resultForm, sequenceWork);
            }
            private void saveToolStripMenuItem_Click(object sender, EventArgs
e) {
                  _applicationHandler.saveProject();
            }
            private void newToolStripMenuItem_Click(object sender, EventArgs e)
{
                  _applicationHandler.createNew();
            }
            private     void     initMenuItemsCaptions(ToolStripItemCollection
menuItems) {
                  foreach (ToolStripMenuItem menuItem in menuItems) {
                        String itemName = StringManager.getString(menuItem.Name
+ "Caption");
                        menuItem.Text = itemName;
                        ToolStripItemCollection items = menuItem.DropDownItems;
                        if (items.Count > 0) {
                              initMenuItemsCaptions(items);
                        }
```

```
            }
        }
        private void initApplicationName() {
            String appName = StringManager.getString("applictionName");
            String                    appVersion                    =
StringManager.getString("applictionVersion");
            this.Text = String.Format("{0} {1}", appName, appVersion);
        }
        private void initHeaders() {
            ControlTextInitializer.setControlText(outletsGridLabel);
            ControlTextInitializer.setControlText(worksGridLabel);
        ControlTextInitializer.setControlText(dependentWorksGridLabel);
            ControlTextInitializer.setControlText(parametersLabel);
            ControlTextInitializer.setControlText(periodLabel);
            ControlTextInitializer.setControlText(moneyLabel);
            ControlTextInitializer.setControlText(startDateLabel);
        }
        private   void   viewDistributionMenuItem_Click(object   sender,
EventArgs e) {
            SequenceWork sequenceWork = _applicationHandler.viewWorks();
            _applicationHandler.showResults(_resultForm, sequenceWork);
        }
    }
}
using System;
using System.Windows.Forms;
namespace mp2 {
    public partial class ResultForm : Form {
        public ResultForm() {
            InitializeComponent();
        }
        private void ResultForm_Load(object sender, EventArgs e) {
            initFormHeader();
            initHeaders();
        }
        private void initHeaders() {
            ControlTextInitializer.setControlText(parametersLabel);
        ControlTextInitializer.setControlText(countedParametersLabel);
            ControlTextInitializer.setControlText(periodLabel);
            ControlTextInitializer.setControlText(moneyLabel);
            ControlTextInitializer.setControlText(startDateLabel);
            ControlTextInitializer.setControlText(finishDateLabel);
            ControlTextInitializer.setControlText(durationLabel);
            ControlTextInitializer.setControlText(workSequenceLabel);
        }
        private void initFormHeader() {
            String                  resultFormHeader                  =
StringManager.getString("resultFormHeader");
            this.Text = resultFormHeader;      }     }}
```

Б.3 Програмний код класів групи «Обробники».

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;
namespace mp2 {
    class ApplicationHandler {
        private Project _project;
        private ILoader _iLoader;
        private string _currentFilename;
        private ISaver _iSaver;
```

```
            private OutletHandler _outletHandler;
            private WorkHandler _workHandler;
            private DependentWorkHandler _dependentWorkHandler;
            private ProjectManager _projectManager;
            private List<Work> _works;
            private ProjectParameteres _projectParameteres;
            #region Controls
            private DataGridView _outletGrid;
            private DataGridView _workGrid;
            private DataGridView _dWorkGrid;
            private NumericUpDown _period;
            private NumericUpDown _money;
            private DateTimePicker _startDate;
            #endregion Controls
            public     ApplicationHandler(DataGridView     outletDataGridView,
DataGridView   workDataGridView,   DataGridView   dependentWorksDataGridView,
NumericUpDown    periodNumericUpDown,    NumericUpDown    moneyNumericUpDown,
DateTimePicker startDateTime) {
                  _outletGrid = outletDataGridView;
                  _workGrid = workDataGridView;
                  _dWorkGrid = dependentWorksDataGridView;
                  _period = periodNumericUpDown;
                  _money = moneyNumericUpDown;
                  _startDate = startDateTime;
                  _period.ValueChanged += PeriodNumericUpDownOnValueChanged;
                  _money.ValueChanged += MoneyNumericUpDownOnValueChanged;
                  _startDate.ValueChanged += StartDateTimeOnValueChanged;
                  init(new Project(), _outletGrid, _workGrid, _dWorkGrid);
                  _iLoader = new XMLLoader();
                  _iSaver = new XMLSaver();

                  _projectManager = new ProjectManager();
            }
            private void init(Project project, DataGridView outletDataGridView,
DataGridView workDataGridView, DataGridView dependentWorksDataGridView) {
                  _outletHandler = new OutletHandler(outletDataGridView);
                  _workHandler = new WorkHandler(workDataGridView);
                  _dependentWorkHandler                  =                  new
DependentWorkHandler(dependentWorksDataGridView);
                  _outletHandler.EntityChangedEvent                          +=
_workHandler.refreshWorkList;
                  _workHandler.EntityChangedEvent                            +=
_dependentWorkHandler.refreshWorkList;
                  _dependentWorkHandler.GetWork += this.getWork;
                  _outletHandler.setSource(project.Outlets);
                  _project = project;

            }
            private void StartDateTimeOnValueChanged(object sender, EventArgs
eventArgs) {
                  DateTime dateTime = _startDate.Value;
                  _project.ProjectParameteres.setStartDate(dateTime);
            }
            private   void   MoneyNumericUpDownOnValueChanged(object   sender,
EventArgs eventArgs) {
                  decimal money = _money.Value;
                  _project.ProjectParameteres.setMoney(money);
            }
            private   void   PeriodNumericUpDownOnValueChanged(object   sender,
EventArgs eventArgs) {
                  decimal period = _period.Value;
                  _project.ProjectParameteres.setPeriod(period);
            }
            private Work getWork(string id) {
```

```
                  Outlet outlet = _outletHandler.getCurrentOutlet();
                  return outlet.getWorks().Find(w => w.Id == id);
            }
            public void loadProject(string filename) {
                  _project = _iLoader.loadProject(filename);
                  init(_project, _outletGrid, _workGrid, _dWorkGrid);
                  setProjectParameters(_project.ProjectParameteres);
                  _currentFilename = filename;
            }

            private        void        setProjectParameters(ProjectParameteres
projectParameteres) {
                  _period.Value = projectParameteres.getPeriod();
                  _money.Value = projectParameteres.getMoney();
                  _startDate.Value = projectParameteres.getStartDate();
            }
            public void saveProject(string filename) {
                  _currentFilename = filename;
                  _iSaver.saveProject(_project, filename);
            }
            public SequenceWork distributeWorks() {
                  SequenceWork                  sequenceWork                  =
_projectManager.manageProject(_project);
                  List<Work> works = _project.getWorks();
                  sequenceWork.setSourceWorks(works);
                  return sequenceWork;
            }
            public SequenceWork viewWorks() {
                  SequenceWork sequenceWork = new SequenceWork();
                  List<Work> works = _project.getWorks();
                  sequenceWork.setSourceWorks(works);
                  return sequenceWork;
            }
            public void saveProject() {
                  this.saveProject(_currentFilename);
            }
            public void createNew() {
                  init(new Project(), _outletGrid, _workGrid, _dWorkGrid);
                  setProjectParameters(_project.ProjectParameteres);
            }
            public   void   showResults(ResultForm   _resultForm,   SequenceWork
sequenceWork) {
                  int                     projectDuration                     =
_project.ProjectParameteres.getDuration();
                  _resultForm = new ResultForm();
                  _resultForm.periodNumericUpDown.Value                       =
_project.ProjectParameteres.getPeriod();
                  _resultForm.moneyNumericUpDown.Value                        =
_project.ProjectParameteres.getMoney();
                  _resultForm.startDateTimePicker.Value                       =
_project.ProjectParameteres.getStartDate();
                  _resultForm.finishDateTimePicker.Value =

      _project.ProjectParameteres.getStartDate().AddDays(projectDuration);
                  _resultForm.durationNumericUpDown.Value = projectDuration;
                  List<Work>  works  =  sequenceWork.getWorks().OrderBy(w  =>
w.getStartDate()).ToList();
                  DataGridView grid = _resultForm.worksDataGridView;
                  grid.Rows.Clear();
                  foreach (Work work in works) {
                        grid.Rows.Add(work.getString());
                  }
                  _resultForm.ShowDialog();
            }
```

```
      }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace mp2 {
      public class DependentWorkHandler : WorkHandler {
            public delegate Work getWork(string id);
            public getWork GetWork;
            public   DependentWorkHandler(DataGridView   workDataGridView)   :
base(workDataGridView) {
                  _dataGridView.CellEndEdit += this.replaceWorkId;
                  _blockedColumns = new int[] {1, 2, 3};
                  _columnsAutoSizeMode = new DataGridViewAutoSizeColumnMode[] {
DataGridViewAutoSizeColumnMode.DisplayedCells,
DataGridViewAutoSizeColumnMode.Fill,      DataGridViewAutoSizeColumnMode.Fill,
DataGridViewAutoSizeColumnMode.Fill };
            }
            private void replaceWorkId(object sender, DataGridViewCellEventArgs
e) {
                  string id = _dataGridView.CurrentCell.Value.ToString();
                  if (e.ColumnIndex == 0 && id != "") {
                        Work neededWork = GetWork(id);
                        if (neededWork != null) {
                              _dataGridView.CurrentRow.Cells[0].Value        =
neededWork.Id;
                              _dataGridView.CurrentRow.Cells[1].Value        =
neededWork.Name;
                              _dataGridView.CurrentRow.Cells[2].Value        =
neededWork.Duration.ToString();
                              _dataGridView.CurrentRow.Cells[3].Value        =
neededWork.Cost.ToString();
                        } else {
                        _dataGridView.Rows.Remove(_dataGridView.CurrentRow);
                        }
                  }
            }
            protected override void onSelectedChanged(object sender, EventArgs
e) {}
                        public void refreshWorkList(Object work) {
                  _currentWork = (Work)work;
                  if (_currentWork != null) {
                        setSource(_currentWork.getPreviousWorks());
                  } else {
                        setSource(new List<Work>());
                  }
                  this.onSelectedChanged(null, null);
            }
      }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace mp2 {
      public class Handler {
            protected DataGridView _dataGridView;
            protected BindingSource _bindingSource;
            protected string[] _headers;
            protected int[] _blockedColumns;
            protected DataGridViewAutoSizeColumnMode[] _columnsAutoSizeMode;
```

```
            protected string entityName;
            public delegate void entityChanged<T>(T entity);
            public Handler(DataGridView dataGridView) {
                  _bindingSource = new BindingSource();
                  _dataGridView = dataGridView;
                  _dataGridView.DataSource = _bindingSource;
                  this._bindingSource.CurrentChanged                              +=
this.onSelectedChanged;
            }
            protected virtual void initColumnHeaders() {
                  for (int i = 0; i < _dataGridView.Columns.Count; i++) {
                        _dataGridView.Columns[_headers[i]].HeaderText        =
StringManager.getString(entityName + _headers[i] + "Header");
                  }
            }
            public virtual void setSource<T>(List<T> list) {
                  _bindingSource.DataSource = list;
                  initColumnHeaders();
                  blockColumns();
                  setColumnAutoSizeMode();
            }
            private void blockColumns() {
                  for (int i = 0; i < _blockedColumns.Length; i++) {
                        _dataGridView.Columns[_blockedColumns[i]].ReadOnly    =
true;
                  }
            }
            protected virtual void setColumnAutoSizeMode() {
                  for (int i = 0; i < _columnsAutoSizeMode.Length; i++) {
                        _dataGridView.Columns[i].AutoSizeMode                 =
_columnsAutoSizeMode[i];
                  }
            }
            protected virtual void onSelectedChanged(object sender, EventArgs
e) {}
      }
}
using System;
using System.Windows.Forms;
namespace mp2 {
      public class OutletHandler : Handler {
            public entityChanged<Outlet> EntityChangedEvent;
            private Outlet _currentOutlet;
            public     OutletHandler(DataGridView     outletDataGridView)     :
base(outletDataGridView) {
                  _headers = new string[] {"Id", "Name", "FinishDate"};
                  entityName = "outlet";
                  _blockedColumns = new int[] {2};
                  _columnsAutoSizeMode = new DataGridViewAutoSizeColumnMode[]
                  {DataGridViewAutoSizeColumnMode.None,
DataGridViewAutoSizeColumnMode.Fill, DataGridViewAutoSizeColumnMode.None};
            }

            protected override void onSelectedChanged(object sender, EventArgs
e) {
                  _currentOutlet = (Outlet) _bindingSource.Current;
                  if (_currentOutlet != null) {
                        EntityChangedEvent(_currentOutlet);
                  }
            }
            public Outlet getCurrentOutlet() {
                  return _currentOutlet;
            }
      }
```

```
}
using System;
using System.Collections.Generic;
using System.Windows.Forms;
namespace mp2 {
      public class WorkHandler : Handler {
            protected Work _currentWork;
            private Outlet _currentOutlet;
            public entityChanged<Work> EntityChangedEvent;
            public    WorkHandler(DataGridView    workDataGridView)    :    base
(workDataGridView) {
                  _headers = new string[] { "Id", "Name", "Duration", "Cost" };
                  entityName = "work";
                  _blockedColumns = new int[] { 0 };
                  _columnsAutoSizeMode = new DataGridViewAutoSizeColumnMode[] {
DataGridViewAutoSizeColumnMode.None,      DataGridViewAutoSizeColumnMode.Fill,
DataGridViewAutoSizeColumnMode.None, DataGridViewAutoSizeColumnMode.None };
            }
            protected override void onSelectedChanged(object sender, EventArgs
e) {
                  if (_bindingSource.Current != null) {
                        _currentWork = (Work) _bindingSource.Current;
                        Outlet o = _currentWork.getOutlet();
                        if (o.Id == "") {
                              _currentWork.setOutlet(_currentOutlet);
                        }
                        if (_currentWork.Id == "") {
                              _currentWork.Id     =     _currentOutlet.Id     +
(_bindingSource.Count + 1000).ToString().Remove(0, 1);
                        }
                  } else {
                        _currentWork = new Work();
                  }
                  EntityChangedEvent(_currentWork);
            }
            public void refreshWorkList(Object outlet) {
                  _currentOutlet = (Outlet)outlet;
                  if (_currentOutlet != null) {
                        setSource(_currentOutlet.getWorks());
                  } else {
                        setSource(new List<Work> ());
                  }
                  this.onSelectedChanged(null, null);
            }
      }
}
```

Б.4 Програмний код класів групи «Утілітні класи»

```
using System;
using System.Windows.Forms;

namespace mp2 {
      public class ControlTextInitializer {
            public static void setControlText(Control control) {
                  control.Text = getControlText(control.Name);
            }
            private static String getControlText(String controlName) {
                  String   header   =   StringManager.getString(controlName   +
"Header");
                  return header;
            }
      }
```

```
}
using System.Collections.Generic;
using System.Xml.Linq;
namespace mp2 {
      public interface ILoader {
            Project loadProject(string filename);
      }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace mp2 {
      interface ISaver {
            void saveProject(Project _project, string filename);
      }
}
using System;
using System.Collections.Generic;
namespace mp2 {
      public class ProbabilitySimulator {
            private Random _random;
            public ProbabilitySimulator() {
                  _random = new Random();
            }
            public int fullEventsGroup(List<decimal> probabilities) {
                  decimal sum = probabilities[0];
                  int current = 0;
                  decimal e = (decimal)_random.NextDouble();
                  while (e >= sum) {
                        current++;
                        sum += probabilities[current];
                  }
                  return current;
            }
            public bool singleEvent(double probability) {
                  return _random.NextDouble() <= probability;
            }
      }
}
using System.Collections.Generic;
using System.Linq;
namespace mp2 {
      class ProjectManager {
            private ProbabilitySimulator _probabilitySimulator;
            private StopCriterias _stopCriterias;
            public ProjectManager() {
                  _probabilitySimulator = new ProbabilitySimulator();
                  _stopCriterias = new StopCriterias();
            }
            private   bool   checkWorkAvailableToStart(Work   work,   decimal
currentBalance) {
                  return  work.getAvailableToStart()  &&  (work.getCost()  <=
currentBalance);
            }
            private  void  copyList(List<Work>  targetList,  IEnumerable<Work>
sourceList) {
                  targetList.Clear();
                  targetList.AddRange(sourceList.Select(work     =>     (Work)
work.Clone()));
            }
            public      SequenceWork      manageProject(List<Work>      works,
ProjectParameteres projectParametres) {
                  bool stop = false;
```

```
                SequenceWork currentSequence = null;
                SequenceWork bestSequence = new SequenceWork();
                int loopCount = 0;
                int currentIteration = 0;
                decimal currentSequenceDuration;
                decimal bestSequenceDuration;
                _stopCriterias.MaxIterationCount = works.Count * 10;
                while (!stop) {
                      currentSequence       =       this.manageWorks(works,
projectParametres);
                      currentSequenceDuration                                  =
currentSequence.getDuration();
                      bestSequenceDuration = bestSequence.getDuration();
                      if (currentSequenceDuration < bestSequenceDuration) {
                            loopCount = 0;
                            bestSequence = currentSequence;
                            for (int i = 0; i < works.Count; i++) {

      works[i].setBestTime(bestSequence[i].getBestTime());
                            }
                      } else {
                            if          (currentSequenceDuration          ==
bestSequenceDuration) {
                                  ++loopCount;
                            }
                      }
                      ++currentIteration;
                      if                (currentIteration                  ==
_stopCriterias.MaxIterationCount || loopCount == _stopCriterias.MaxLoopCount)
{
                            stop = true;
                      }
                }
                bestSequence.setDateTime(projectParametres.getStartDate());
            projectParametres.setDuration((int)bestSequence.getDuration());
                return bestSequence;
            }
            private  decimal  getAvailableWorksAndCountCost(List<Work>  works,
decimal currentBalance, List<Work> selectedWorks) {
                decimal totalCost = 0;
                selectedWorks.Clear();
                foreach (Work work in works) {
                      if (checkWorkAvailableToStart(work, currentBalance)) {
                            selectedWorks.Add((Work)work.Clone());
                            totalCost += work.getCost();
                      }
                }
                return totalCost;
            }
            private      SequenceWork      manageWorks(List<Work>      works,
ProjectParameteres projectParametres) {
                List<Work> projectWorks = new List<Work>();
                List<Work> processingWorks = new List<Work>();
                List<Work> finishedWorks = new List<Work>();
                List<Work> availableWorks = new List<Work>();
                List<Work> worksToDo = new List<Work>();
                SequenceWork managedWorksResult = new SequenceWork();
                decimal costOfAvailableWorks = 0;
                decimal currentBalance = 0;
                decimal currentTime = 0;
                decimal minTime = 0;
                int currentInvestment = 0;
                bool stop = false;
                copyList(projectWorks, works);
```

```
                while (!stop) {
                      int     investment     =     (int)(currentTime     /
projectParametres.getPeriod());
                      if (currentInvestment < (investment + 1)) {
                            currentInvestment++;
                            currentBalance += projectParametres.getMoney();
                      }
                      costOfAvailableWorks                                    =
this.getAvailableWorksAndCountCost(projectWorks,               currentBalance,
availableWorks);
                      if (availableWorks.Count > 0) {
                            if (costOfAvailableWorks <= currentBalance) {
                                  this.copyList(worksToDo, availableWorks);
                            } else {
                                  this.copyList(worksToDo,
this.selectWorksToDo(availableWorks, currentBalance, currentTime));
                            }
                            foreach (Work work in worksToDo) {
                                  work.setStartTime(currentTime);
                                  currentBalance -= work.getCost();
                                  projectWorks.Remove(work);
                                  processingWorks.Add((Work) work.Clone());
                            }
                      }
                      if (processingWorks.Count != 0) {
                            minTime     =     processingWorks.Min(work     =>
work.getFinishTime());
                      }
                      if      ((minTime     <=     (currentInvestment     *
projectParametres.getPeriod())) && (processingWorks.Count != 0)) {
                            currentTime = minTime;
                            List<Work> worksList = processingWorks.Where(work
=> work.getFinishTime() == minTime).ToList();
                            finishedWorks.AddRange(worksList.Select(work  =>
(Work)work.Clone()).ToList());
                            processingWorks.RemoveAll(work                =>
work.getFinishTime() == minTime);
                            foreach (Work work in worksList) {
                                  foreach (Work projectWork in projectWorks)
{
                                        projectWork.removeIfContains(work);
                                  }
                            }
                      } else {
                            currentTime     =     (currentInvestment     *
projectParametres.getPeriod());
                      }
                      if (projectWorks.Count == 0) {
                            stop = true;
                      }
                }
                if (processingWorks.Count > 0) {
                      finishedWorks.AddRange(processingWorks.Select(work  =>
(Work)work.Clone()).ToList());
                }
                currentTime        =        finishedWorks.Max(work        =>
work.getFinishTime());
                managedWorksResult.setWorks(finishedWorks);
                managedWorksResult.setDuration(currentTime);
                return managedWorksResult;
          }
          private IEnumerable<Work> selectWorksToDo(List<Work> works, decimal
currentBalance, decimal currentTime) {
                List<Work> worksToDo = new List<Work>();
```

```
                decimal currentCost = 0;
                do {
                      int index = -1;
                      if (works.Count > 1) {
                            /*
                             * Расчет вероятностей работ
                             */
                            List<decimal>          probabilities          =
this.getProbabilities(works, currentTime);
                            /*
                             * Симуляция полной группы событий
                             */
                            index                                         =
_probabilitySimulator.fullEventsGroup(probabilities);
                            /*
                             * Проверка возможности выполнения работы
                             */
                      } else {
                            index = 0;
                      }
                      if    ((currentCost    +    works[index].getCost())    <=
currentBalance) {
                            currentCost += works[index].getCost();
                            /*
                             * Добавление работы в список на выполнение
                             */
                            worksToDo.Add((Work)works[index].Clone());
                      }
                      /*
                      * Удаление работы из общего списка
                      */
                      works.RemoveAt(index);
                } while (works.Count > 0);
                return worksToDo;
          }
          private  List<decimal>  getProbabilities(IEnumerable<Work>  works,
decimal currentTime) {
                List<decimal> probabilities = new List<decimal>();
                decimal sum = 0;
                foreach (Work work in works) {
                      decimal                workDifference                =
work.getTimeDifference(currentTime);
                      sum += workDifference;
                      probabilities.Add(workDifference);
                }
                if (sum == 0) {
                      for (int i = 0; i < probabilities.Count; i++) {
                            probabilities[i]                              =
(decimal)1.0/probabilities.Count;
                      }
                } else {
                      for (int i = 0; i < probabilities.Count; i++) {
                            probabilities[i]  =  (sum  -  probabilities[i])  /
sum;
                      }
                }
                return probabilities;
          }
          public SequenceWork manageProject(Project project) {
                SequenceWork                  sequenceWork                  =
this.manageProject(project.getWorks(), project.ProjectParameteres);
                List<Outlet> outlets = project.Outlets;
                List<Work> works = sequenceWork.getWorks();
                project.updateOutletWorks(project.Outlets, works);
```

```
                  foreach (Outlet outlet in outlets) {
                        outlet.countFinishDate();
                  }
                  return sequenceWork;
            }
      }
}
using System.Resources;
using mp2.Properties;
namespace mp2 {
      public class StringManager {
            private static ResourceManager _resourceManager;

            static StringManager() {
                  _resourceManager = Resources.ResourceManager;
            }
            public static string getString(string stringName) {
                  return _resourceManager.GetString(stringName);
            }
      }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Xml.Linq;
namespace mp2 {
      public class XMLLoader : ILoader {
            public Project loadProject(string filename) {
                  XDocument doc = XDocument.Load(filename);
                  XElement outletsElement = doc.Root.Element("outlets");
                  XElement worksElement = doc.Root.Element("works");
                  XElement             projectParameteresElement             =
doc.Root.Element("project_parameteres");
                  List<Outlet> outlets = this.loadOutlets(outletsElement);
                  List<Work> works = this.loadWorks(worksElement, outlets);
                  ProjectParameteres           projectParameteres            =
this.loadProjectParameteres(projectParameteresElement);
                  return new Project(outlets, works, projectParameteres);
            }
            private List<Outlet> loadOutlets(XElement outletsElement) {
                  string id;
                  string name;
                  DateTime date;
                  List<Outlet> outlets = new List<Outlet>();
                  try {
                        var outletsNodes = outletsElement.Elements("outlet");
                        foreach (var outletNode in outletsNodes) {
                              id = outletNode.Element("id").Value;
                              name = outletNode.Element("name").Value;
                              date                                          =
Convert.ToDateTime(outletNode.Element("finish_date").Value);
                              outlets.Add(new Outlet(id, name, date));
                        }
                  } catch (Exception exception) {
                        throw exception;
                  }
                  return outlets;
            }
            private  List<Work>  loadWorks(XElement  worksElement,  List<Outlet>
outlets) {
                  string id, idp, outletId;
                  string name;
                  int duration;
                  decimal cost;
```

```
                DateTime startDate;
                DateTime finishDate;
                List<Work> works = new List<Work>();
                try {
                      var workNodes = worksElement.Elements("work");
                      workNodes        =        workNodes.OrderBy(w        =>
w.Element("id").Value);
                      foreach (var workNode in workNodes) {
                            id = workNode.Element("id").Value;
                            name = workNode.Element("name").Value;
                            duration                                          =
Convert.ToInt32(workNode.Element("duration").Value);
                            cost                                              =
Convert.ToDecimal(workNode.Element("cost").Value);
                            outletId = workNode.Element("outlet_id").Value;
                            Outlet  outlet  =  outlets.Find(o  =>  o.Id  ==
outletId);
                            startDate                                         =
Convert.ToDateTime(workNode.Element("start_date").Value);
                            finishDate                                        =
Convert.ToDateTime(workNode.Element("finish_date").Value);
                            List<Work> prevs = new List<Work>();
                            var                  prevWorks                    =
workNode.Element("previous_works").Elements("id");
                            if (prevWorks.Count() != 0) {
                                  foreach (var work in prevWorks) {
                                        idp = work.Value;
                                        prevs.Add(works.First(a           =>
a.getId().Equals(idp)));
                                  }
                            }
                            works.Add(new  Work(id,  name,  duration,  cost,
outlet, prevs, startDate, finishDate));
                      }
                } catch (Exception exception) {

                      throw exception;
                }
                return works;
          }
          private     ProjectParameteres     loadProjectParameteres(XElement
projectParameteresElement) {
                decimal period = 0;
                decimal money = 0;
                int duration = 0;
                DateTime startDate;
                try {
                      period                                                  =
Convert.ToDecimal(projectParameteresElement.Element("invest_period").Value);
                      money                                                   =
Convert.ToDecimal(projectParameteresElement.Element("invest_money").Value);
                      startDate                                               =
Convert.ToDateTime(projectParameteresElement.Element("start_date").Value);
                      duration                                                =
Convert.ToInt32(projectParameteresElement.Element("duration").Value);
                } catch (Exception exception) {
                      throw exception;
                }
                return  new  ProjectParameteres(period,  money,  startDate,
duration);
          }
     }
}
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Xml.Linq;
namespace mp2 {
      class XMLSaver : ISaver{
            public void saveProject(Project _project, string filename)
            {
                  XDocument doc = new XDocument();
                  XElement root = new XElement("project");
            root.Add(saveProjectParameteres(_project.ProjectParameteres));
                  root.Add(saveOutlets(_project.Outlets));
                  root.Add(saveWorks(_project.getWorks()));
                  doc.AddFirst(root);
                  doc.Save(filename);
            }
            private     XElement     saveProjectParameteres(ProjectParameteres
projectParameteres)
            {
                  decimal period = projectParameteres.getPeriod();
                  decimal money = projectParameteres.getMoney();
                  DateTime startDate = projectParameteres.getStartDate();
                  int duration = projectParameteres.getDuration();
                  XElement el = new XElement("project_parameteres");
                  XElement periodEl = new XElement("invest_period");
                  periodEl.SetValue(period);
                  el.Add(periodEl);
                  XElement moneyEl = new XElement("invest_money");
                  moneyEl.SetValue(money);
                  el.Add(moneyEl);
                  XElement startDateEl = new XElement("start_date");
                  startDateEl.SetValue(startDate.ToShortDateString());
                  el.Add(startDateEl);
                  XElement durationEl = new XElement("duration");
                  durationEl.SetValue(duration.ToString());
                  el.Add(durationEl);
                  return el;
            }
            private XElement saveOutlets(List<Outlet> outletList) {
                  XElement el = new XElement("outlets");
                  foreach (var outlet in outletList)
                  {
                        XElement outletEl = new XElement("outlet");
                        XElement idEl = new XElement("id");
                        idEl.SetValue(outlet.Id);
                        outletEl.Add(idEl);
                        XElement nameEl = new XElement("name");
                        nameEl.SetValue(outlet.Name);
                        outletEl.Add(nameEl);
                        XElement finishDateEl = new XElement("finish_date");
            finishDateEl.SetValue(outlet.FinishDate.ToShortDateString());
                        outletEl.Add(finishDateEl);
                        el.Add(outletEl);
                  }
                  return el;
            }
            private XElement saveWorks(List<Work> workList) {
                  XElement el = new XElement("works");
                  foreach (var work in workList)
                  {
                        XElement workEl = new XElement("work");
                        XElement idEl = new XElement("id");
                        idEl.SetValue(work.Id);
                        workEl.Add(idEl);
```

```
                        XElement nameEl = new XElement("name");
                        nameEl.SetValue(work.Name);
                        workEl.Add(nameEl);
                        XElement durationEl = new XElement("duration");
                  durationEl.SetValue(work.Duration.ToString().Replace('.',
','));
                        workEl.Add(durationEl);
                        XElement costEl = new XElement("cost");
                        costEl.SetValue(work.Cost.ToString().Replace('.',
','));
                        XElement startDateEl = new XElement("start_date");
            startDateEl.SetValue(work.getStartDate().ToShortDateString());
                        workEl.Add(startDateEl);
                        XElement finishDateEl = new XElement("finish_date");
            finishDateEl.SetValue(work.getFinishDate().ToShortDateString());
                        workEl.Add(finishDateEl);
                        workEl.Add(costEl);
                        XElement outletIdEl = new XElement("outlet_id");
                        outletIdEl.SetValue(work.getOutlet().Id);
                        workEl.Add(outletIdEl);
                        XElement prevWorksEl = new XElement("previous_works");
                        foreach (var prevWork in work.getPreviousWorks())
                        {
                              XElement prevWorkIdEl = new XElement("id");
                              prevWorkIdEl.SetValue(prevWork.Id);
                              prevWorksEl.Add(prevWorkIdEl);
                        }
                        workEl.Add(prevWorksEl);
                        el.Add(workEl);
                  }
                  return el;
            }
      }
```

Додаток В

Роздатковий матеріал

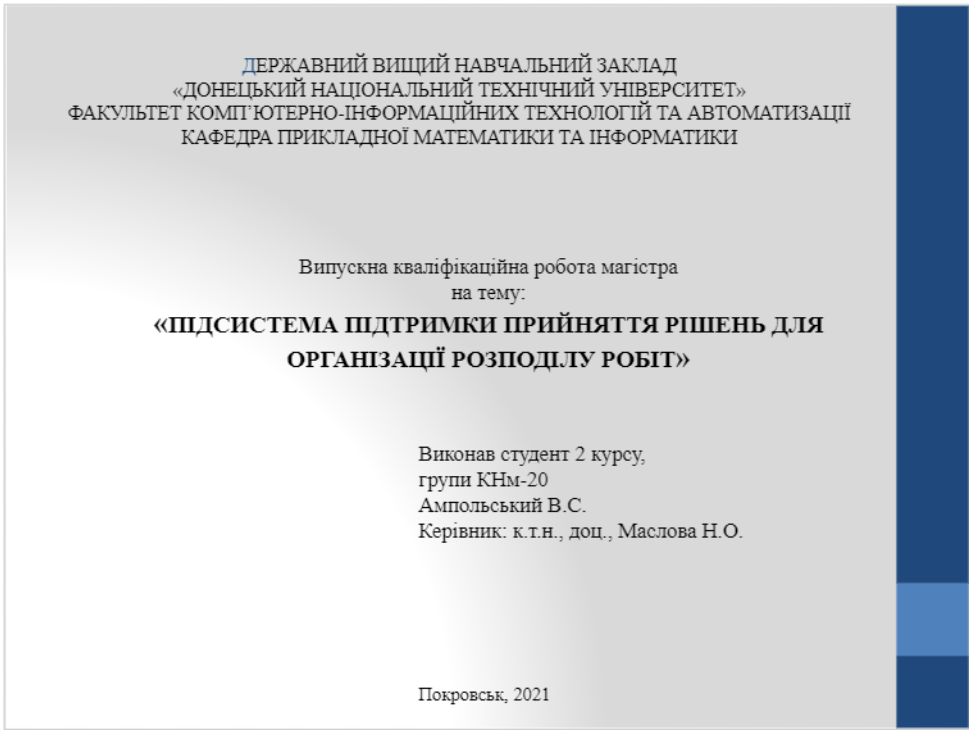

Рисунок В.1 – Перший аркуш презентації

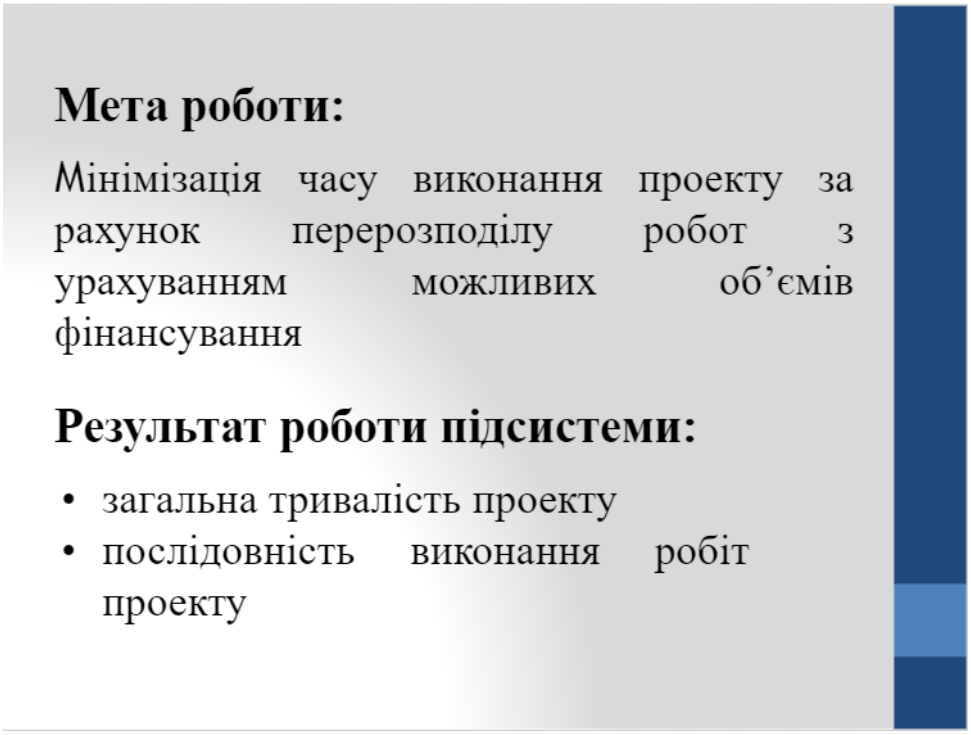

Рисунок В.2 – Другий аркуш презентації

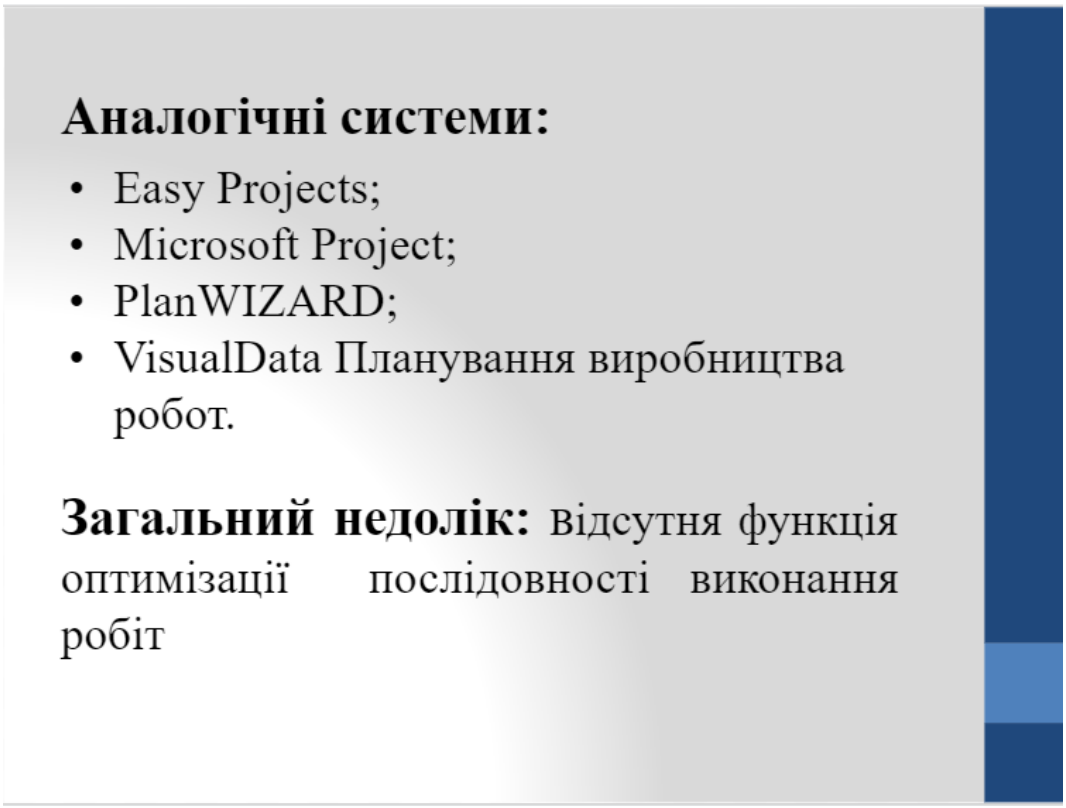

Рисунок В.3 – Третій аркуш презентації

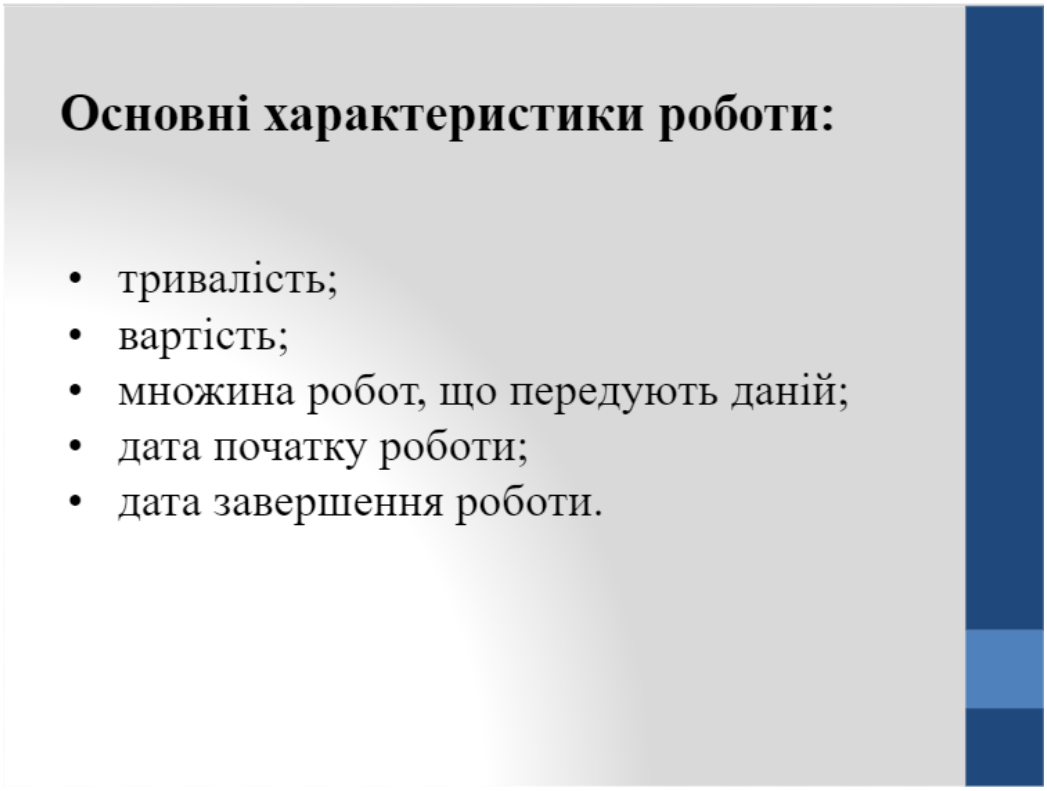

Рисунок В.4 – Четвертий аркуш презентації

Цільова функція:

$$P^t = f(T,t,M) \rightarrow \min$$

P_t – тривалість усього проекту,
T – множина торговельних точок,
t – період інвестування,
M – об’єм інвестування

Рисунок В.5 – П’ятий аркуш презентації

Алгоритм відбору робіт для виконання

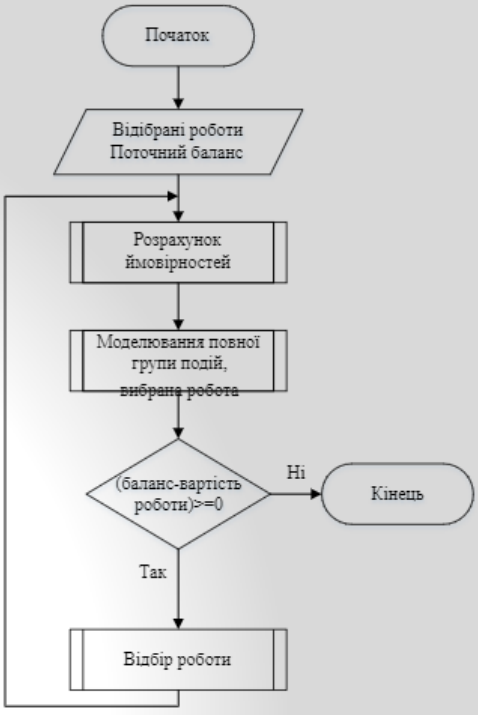

Рисунок В.6 – Шостий аркуш презентації

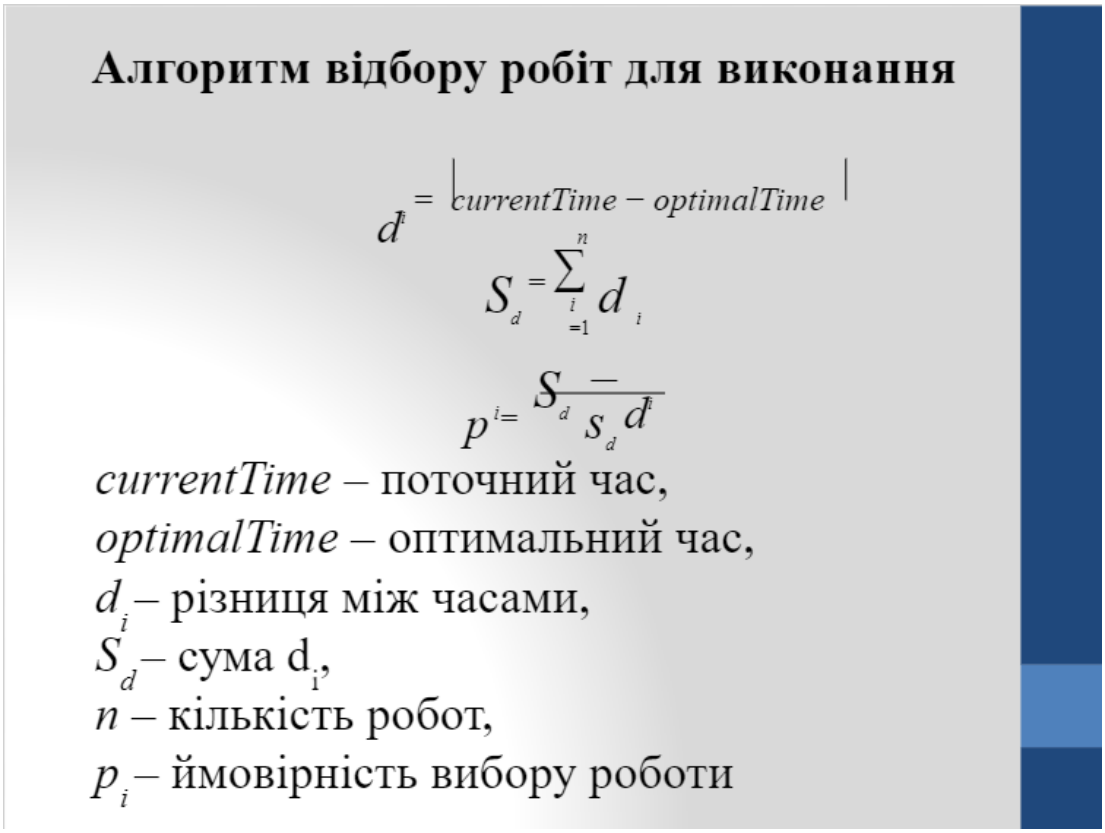

Рисунок В.7 – Сьомий аркуш презентації

Рисунок В.8 – Восьмий аркуш презентації

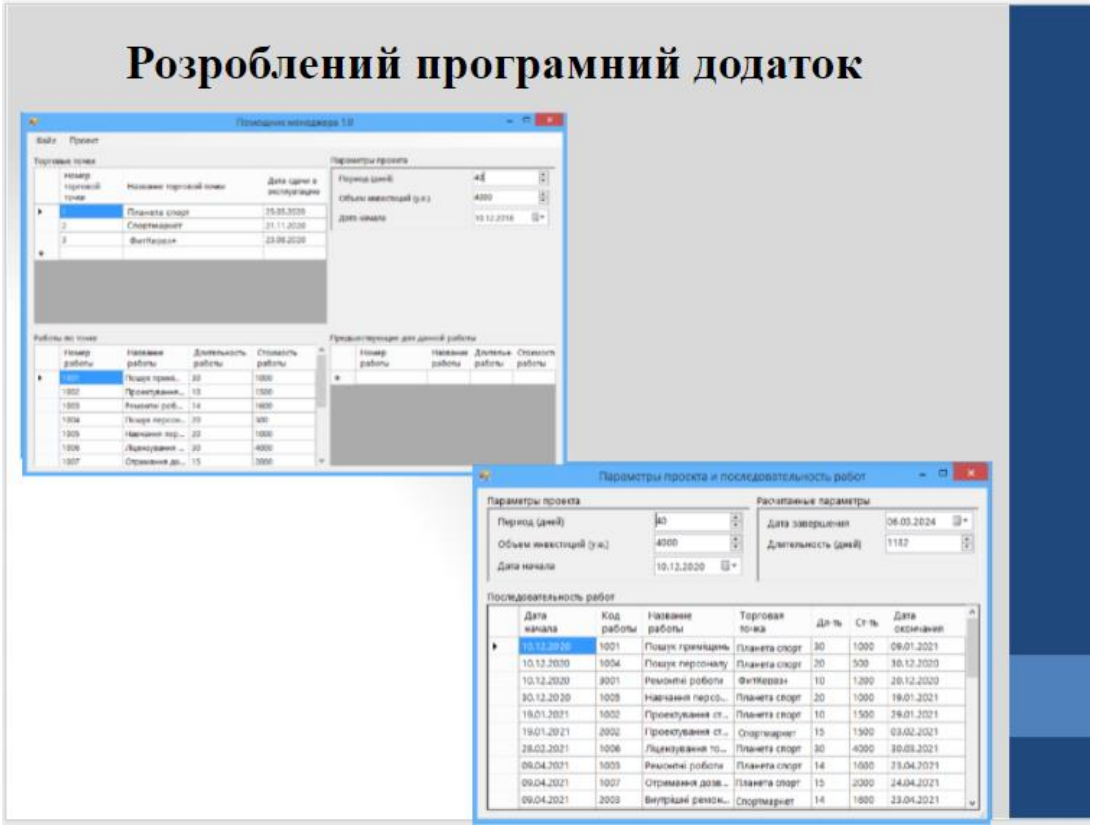

Рисунок В.9 – Дев’ятий аркуш презентації

Рисунок В.10 – Десятий аркуш презентації

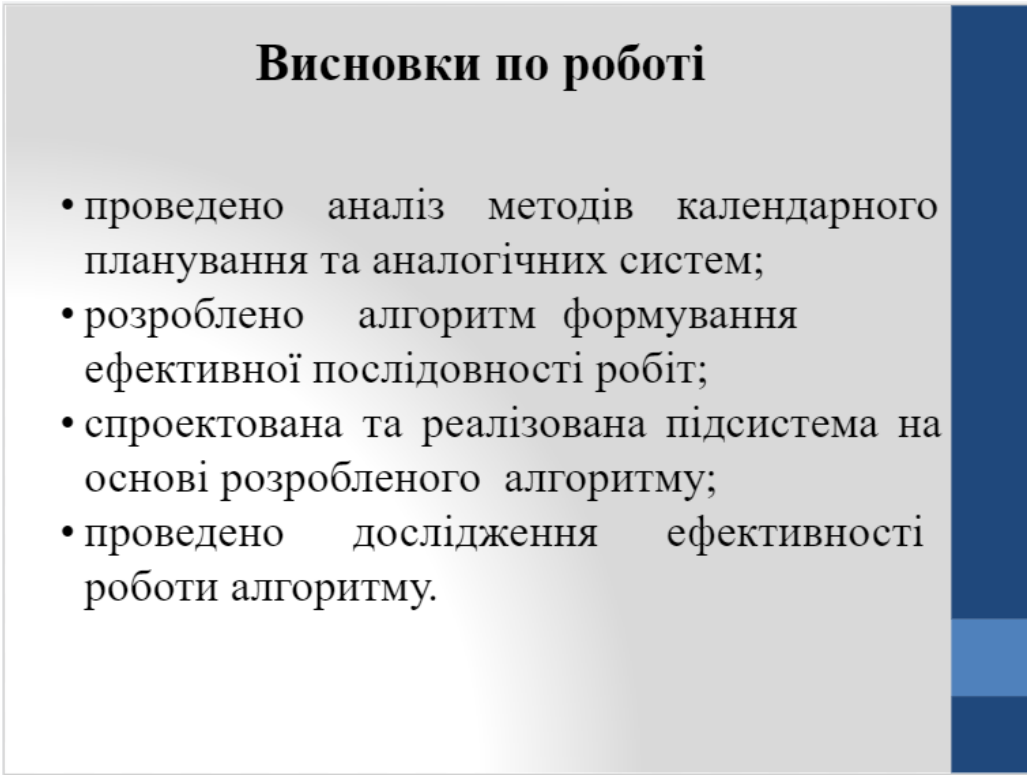

Рисунок В.11 – Одинадцятий аркуш презентації