МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут
комп'ютерних наук та управління проектами
_____
(повне найменування інституту, назва факультету)
Програмного забезпечення автоматизованих систем
_____
(повна назва кафедри)

Пояснювальна записка

до кваліфікаційної (магістерської) роботи

за темою **Розробка ймовірнісної моделі управління змістом програмного забезпечення для обліку енергії та розробка програми для її реалізації**

Виконав: студент 6 курсу, групи 6151м
спеціальності
121 «Інженерія програмного забезпечення»
_____
(шифр і назва спеціальності)

_____Черв'яков В С._____
(підпис, прізвище та ініціали)
Керівник _____Пономаренко Т.В._____
(підпис, прізвище та ініціали)
Рецензент _____Приходько С.Б._____
(підпис, прізвище та ініціали)
Завідувач кафедри _____Приходько С.Б.___
(підпис, прізвище та ініціали)

м. Миколаїв – 2020 р.

Міністерство освіти і науки України
Національний університет кораблебудування
імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра програмного забезпечення автоматизованих систем
Освітній ступень Магістр
Галузь 12 «Інформаційні технології»
(шифр і назва)
Спеціальність 121 «Інженерія програмного забезпечення»
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри

" 26 " 10 2020 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ (МАГІСТЕРСЬКУ) РОБОТУ СТУДЕНТУ

Черв'якова Владислава Сергійовича

1.Тема магістерської роботи Розробка ймовірнісної моделі управління змістом програмного забезпечення для обліку енергії та розробка програми для її реалізації

керівник роботи Пономаренко Т.В.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від " 26 " жовтня 2020 року №1037-уч

2. Строк подання студентом роботи 01.12.2020 року

3. Вихідні дані до роботи

4.Зміст магістерської роботи (МР):

• Титульний аркуш, завдання на кваліфікаційну (магістерську) роботу, реферат (українською, англійською), зміст, перелік умовних позначень, символів, одиниць та термінів (за необхідності).

• Вступ (Актуальність теми. Зв'язок роботи з науковими програмами, планами, темами. Мета і завдання дослідження. Об'єкт дослідження. Предмет дослідження. Методи дослідження. Наукова новизна одержаних результатів. Практичне значення одержаних результатів. Особистий внесок здобувача. Апробація результатів досліджень**.** Публікації.)

• Огляд літератури за темою, обґрунтування необхідності проведення досліджень за обраною темою, вибір напрямків досліджень, мета дослідження, основні задачі дослідження

• Викладення результатів власних досліджень з висвітленням того нового, що пропонується

• Проект програмного забезпечення

• Результати досліджень та розробки проекту програмного забезпечення

• Організаційно-економічний розділ

Розділи з охорони праці та охорони навколишнього середовища

РЕФЕРАТ

Черв’якова Владислава Сергійовича

«Розробка ймовірнісної моделі управління змістом програмного забезпечення для обліку енергії та розробка програми для її реалізації»

Кваліфікаційна робота на здобуття освітнього рівня магістра зі спеціальності 121 – «Інженерія програмного забезпечення». Національний університет кораблебудування імені адмірала Макарова. Миколаїв, 2020 р.

Обсяг роботи: 130 стор., 4 табл., 10 рис., 16 використаних джерел, 5 додатків.

Актуальність теми роботи: Необхідність підвищення достовірності управління змістом розробки проєкту, шляхом створення зручного наочного інструментарію для менеджерів при управлінні змістом проєктів з врахуванням його статистичної бази та перспектив розвитку, враховуючи велику ступінь невизначеності, що росте з кожним днем в умовах пандемії, ставить задачу створення більш надійної бази для планування, аналізу та управління проєктами з розробки програмного забезпечення з застосуванням не тількі сценарного підходу, а й доповненням моделі інформацією про статистичну історію настання подій проекту з формуванням ймовірністного графу розвитку змісту проекту.

Мета та завдання дослідження. Метою роботи є підвищення достовірності прогнозування розвитку змісту проєктів програмного забезпечення за допомогою створення ймовірнісної графової моделі змісту програмного проєкту з обліку енергії. Для досягнення мети роботи необхідно вирішити такі завдання: провести аналіз змісту проєкту з обліку енергії; виконати аналіз стратегічних методів планування та їх вплив на розвиток сценаріїв подій; розробити програмне забезпечення для управління змістом проєкту на основі ймовірністної моделі.

Завдання дослідження: підвищення достовірності прогнозування розвитку змісту проєктів програмного забезпечення за допомогою застосування створеної ймовірнісної моделі управління змістом при розробці програмного забезпечення обліку енергії; розробити програму для управління змістом проєктів програмного забезпечення.

Об'єктом дослідження є процес управління змістом програмних проєктів.

Предметом дослідження є ймовірністна графова модель управління змістом проєкту програмного забезпечення з обліку енергії.

Методи дослідження. У вирішенні поставлених завдань використано методи теорії графів, методи статистичного аналізу даних, методи підтримки прийняття рішень та метод прямого процесу планування (як один із методів стратегічного планування).

Наукова новизна одержаних результатів. Створено ймовірнісну модель управління змістом програмного забезпечення, на основі застосування теорії ймовірності до управління змістом проектів, що дозволить підвищити достовірність прогнозування подальшого розвитку змісту проєкту враховуючи не тільки експертну оцінку, але й статистичну базу минуліх подій проекту з врахуванням прогнозів що не справдилися.

Практичне значення одержаних результатів. Розроблено програмне забезпечення для побудови ймовірнісної моделі змісту проєктів зі створення програмного забезпечення для обліку енергії.

Ключові слова: ЙМОВІРНІСНА ГРАФОВА МОДЕЛЬ, УПРАВЛІННЯ ЗМІСТОМ, ЙМОВІРНІСНІ ГРАФИ.

ABSTRACT

of Vladislav Sergeevich Chervyakov

«Development of the probabilistic model of the energy metering software scope management and developing the software for its implementation»

The qualification work in obtaining a master's degree in specialty 121 - "Software Engineering". Admiral Makarov National University of Shipbuilding. Mykolayiv, 2020.

The qualification work is presented on the 130 pages of typewritten text, contains 4 tables, 10 figures, 5 appendices and 16 references.

Relevance of the topic of the work: the need to increase the efficiency of content management of the EnergyDB project, by creating convenient visual tools for managers in managing the content of projects that are multi-alternative and stochastic, taking into account its statistical base and development prospects, given the growing uncertainty in a pandemic situation.

The goal and objectives of the study. The goal of the study is to increase the efficiency of software content management by creating a probabilistic graph model of the content of the software project for energy accounting. To achieve the goal of the work it is necessary to solve the following tasks: to analyze the mathematical model of analysis of the content of innovative projects; perform an analysis of strategic planning methods and their impact on the development of event scenarios; develop software for project content management based on a scenario approach.

Objectives of the study: to analyze existing models of content management in the development of energy accounting software; develop a program to manage the content of software projects.

The object of the study is the process of of the scope managment of the energy software projects.

The subject of the study is probabilistic graph models of the scope management of energy accounting software projects.

The methods of the study. The methods of graph theory, methods of statistical data analysis, methods of decision support and the method of direct planning process (as one of the methods of strategic planning) are used in solving the problems.

The scientific novelty of obtained results. Mathematical model of scenario analysis for project content management was further developed, based on the application of strategic planning methods, which will take into account the influence of actors on the development of scenarios, which in turn has a more positive effect on planning and forecasting innovation projects.

The practical significance of obtained results. A software tool has been developed to use a scripting approach in managing the content of software projects.

Approbation of study results. The results of the research were published at the All-Ukrainian scientific-practical applicants for higher education and young scientists, Kropyvnytskyi, November 25-27. 2020 "Computer Engineering and Cybersecurity: Achievements and Innovations".

Publications. The results of the work were published in the materials of the All-Ukrainian scientific-practical Internet conference, the materials of the II All-Ukrainian. scientific-practical conf. applicants for higher education and young scientists, Kropyvnytskyi, November 25-27, 2020 / Ministry of Education and Science of Ukraine, Gos. Science. Institution "Institute for Modernization of Educational Content", Central Ukrainian nat. tech. University. - Kropyvnytskyi: CNTU, 2020. - P. 47.

Keywords: PROBABILISTIC GRAPH MODEL, CONTENT MANAGEMENT, PROBABILISTIC GRAPHS.

ЗМІСТ

ВСТУП

Проєкти програмного забезпечення в умовах світової кризи та постійних змін економічного стану враховуючи Brexit, локдауни та різні форми карантинних заходів що застосовуються, додають ще більше багатоальтернативності та стохастичності при управлінні змістом. Звідси виникає складність прогнозування, оцінки майбутньої конкурентоспроможності та ринкової адаптації проєктів. Важливою умовою здійсненності такого проєкту є вибір надійної бази прогнозування та аналізу інновації, а також розробка нових методів управління враховуючи високу ступінь невизначеності подій.

Значний вклад у розробку методів управління проектами і в тому числі управління змістом проектів зробили С.Д. Бушуєв, Ю.М. Тесля, К.В. Кошкін, В.А. Рач, І.В. Кононенко, Р.А. Фатхутдинов, Чернов С.К. Науковою базою дослідження стали роботи В.М. Глушкова, М.З. Згуровського, Н.Д. Панкратової, П.І. Бідюка, О.І. Ларичева, Б.Г. Литвака, І.І. Коваленка тощо. Ці роботи створили необхідні теоретичні передумови.

Сучасним інструментом планування, аналізу та управління інноваційними проєктами є сценарний підхід до work based structure, заснований на побудові дерева цілей або прогнозного графа, що відображає як альтернативні шляхи розвитку подій, потреб, так і ймовірнісну оцінку можливостей розвитку проєктів.

Створюючи початковий план проекту менеджер стикається з великою кількістю альтернативних сценаріїв зміни розвитку проєкту в контексті досягнення генеральної цілі. На розвиток сценаріїв та на досягнення їх цілей впливають фактори оточення, та стейкхолдери.

Актуальність теми роботи**:** Необхідність підвищення достовірності управління змістом розробки проєкту, шляхом створення зручного наочного

інструментарію для менеджерів при управлінні змістом проєктів з врахуванням його статистичної бази та перспектив розвитку, враховуючи велику ступінь невизначеності, що росте з кожним днем в умовах пандемії, ставить задачу створення більш надійної бази для планування, аналізу та управління проєктами з розробки програмного забезпечення з застосуванням не тількі сценарного підходу, а й доповненням моделі інформацією про статистичну історію настання подій проекту з формуванням ймовірністного графу розвитку змісту проекту.

Метою роботи є підвищення достовірності прогнозування розвитку змісту проєктів програмного забезпечення за допомогою створення ймовірнісної графової моделі змісту програмного проєкту з обліку енергії. Для досягнення мети роботи необхідно вирішити такі завдання: провести аналіз змісту проєкту з обліку енергії; виконати аналіз стратегічних методів планування та їх вплив на розвиток сценаріїв подій; розробити програмне забезпечення для управління змістом проєкту на основі ймовірністної моделі.

Завданням дослідження є підвищення достовірності прогнозування розвитку змісту проєктів програмного забезпечення за допомогою застосування створеної ймовірнісної моделі управління змістом при розробці програмного забезпечення обліку енергії; розробити програму для управління змістом проєктів програмного забезпечення.

Об'єктом дослідження є процес управління змістом програмних проєктів.

Предметом дослідження є ймовірністна графова модель управління змістом проєкту програмного забезпечення з обліку енергії.

Методи дослідження. У вирішенні поставлених завдань використано методи теорії графів, методи статистичного аналізу даних, методи підтримки прийняття рішень та метод прямого процесу планування (як один із методів стратегічного планування).

Наукова новизна одержаних результатів. Створено ймовірнісну модель управління змістом програмного забезпечення, на основі застосування теорії ймовірності до управління змістом проектів, яка дозволить підвищити достовірність прогнозування подальшого розвитку змісту проєкту враховуючи не тільки екпертну оцінку, але й статистичну базу минуліх подій проекту з врахуванням прогнозів що не справдилися.

Практичне значення одержаних результатів. Розроблено програмне забезпечення для побудови ймовірнісної моделі змісту проєктів зі створення програмного забезпечення для обліку енергії.

Апробація результатів досліджень. Результати досліджень були оприлюднені на Всеукраїнській науково-практичній здобувачів вищої освіти й молодих учених, м. Кропивницький, 25–27 листоп. 2020 р. “Комп’ютерна інженерія і кібербезпека : досягнення та інновації”.

Публікації. Основні результати кваліфікаційної роботи викладено у 1 науковій праці- тезах конференції.

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ УПРАВЛІННЯ ЗМІСТОМ

1.1 Опис управління змістом програмного забезпечення для обліку енергії

Процесс управління змістом програмного забезпечення включає документування, збір та аналіз потреб всіх зацікавлених в досягненні генеральної цілі проекту сторін. Як досягти цілі і коли це робити це дуже нагальне питання для кожного проектного менеджера. Іншим аспектом управління змістом є визначення тих задач які можуть не знадобитися і на які не бажано витрачати ресурси проекту. Визначення змісту зажди потребує докладного опису всіх дрібних елементів які треба врахувати при проектній розробці. Для цієї мети широко використовується ієрархічна структура робіт, яка базується а декомпозиції глобальної цілі на підцілі і елементарні роботи які можна довірити виконавцям з метою ефективного управління і моніторингу стану проекту.

Управління змістом складається з ряду планів: базовому плану по змісту, плану управління змінами, плану управління конфігурацією та плану управління вимогами, а також матриці відстеження вимог, матриці відповідальності тощо. Все це вимагає багато зусиль і часу але є вельми необхідним для успішності проекту. Проекти з розробки програмного забезпечення завжди відрізнялися великим ступенем невизначеності, тому застосування автоматизованих статистичних методів аналізу завди зустрічається спеціалістами з проектного менеджменту з великим ентузіазмом.

Аналіз відхилень від основних планів розвитку змісту, вимірювання виконання ходу проекту є основними завданнями, що виконує проектний менеджер, тому інструменти наочної візуалізації є завжди затребуваними та актуальними.

Важливою умовою досягнення цілей є прийняття рішень про необхідність коригувальних або запобіжних дій, що заважають виконанню завдань, тому системи підтримки прийняття рішень, що мають розгалуджену систему нотифікування менеджера мають великий попит на ринку програмного забезпечення.

Використання ймовірнісних моделей сьогодні дуже популярне в галузі енергозбереження, енергоефективності та різних систем інформаційного моніторингу і спеціалісти з менеджменту цієї галузі поставили завдання на створення інструменту для управління розвитку змістоу проекту з енергозбереження поставили за завдання реалізацію інструменту на базі ймовірнісного графу, що буде допомагати їх не тільки враховувати ризики у виробництві, але й ефективно керувати розвитком змісту проекту. В якості коефіцієнтів відносної важливості будуть використовуватися значення ймовірностей настання подій на дереві цілей, та для його аналізу можуть бути використані основні формули теорії ймовірностей.

Результати кожної події повинні складати вичерпні множини, тобто повинні виконуватися умови:

$$P_{1,1} + P_{1,2} = 1;$$
$$P_{2,1} + P_{2,2} + P_{2,3} = 1;\ P_{2,4} + P_{2,5} = 1;\ P_{3,1} + P_{3,2} = 1;\ P_{3,3} = 1;$$
$$P_{3,4} + P_{3,5} + P_{3,6} + P_{3,7} = 1.$$

Повна ймовірність кожного результату визначається як добуток всіх ймовірностей, зазначених на гілках дерева, починаючи від даного результату і закінчуючи коренем дерева (основною метою).

Моделі, засновані на альтернативних графах, можуть застосовуватися не тільки для аналізу початкових стадій процесів розробки продукції, а й для будь-яких інших, в яких необхідно здійснити аналіз комплексу альтернативних ситуацій.

Таким чином, розглянутий підхід заснований на уявленні про процес створення нової продукції у вигляді багатоваріантного альтернативного ймовірнісного графа *G (J, U*), де *J* - множина вершин, *U* - множина дуг, в якому в тій чи іншій комбінації застосовуються вісім типів вершин [7], відображаємо різні ситуації в розроблюваних сценаріях. Ймовырнісна модель дозволяє імітувати процес оцінки та прийняття рішення в місцях альтернативного розгалуження процесу, визначити повну ймовірність кожного з передбачених варіантів розробки, час і витрати, пов'язані з реалізацією того чи іншого сценарію проєкту. Крім того, вона дає можливість визначати функції поділу ресурсів для досягнення цілей з урахуванням відносних переваг кожного з варіантів його реалізації.

1.2 Постановка задачі дослідження та основних вимог до розробки програмного забезпечення управління змістом програмного забезпечення для обліку енергії

Виходячи з результатів аналізу є нагальною необхідність і доцільність розробки програмного інструменту для управління змістом проєктів на базі ймовірнісної моделі, який являтиме собою систему підтримки прийняття рішень. Програмний продукт повинен менеджерові зручно демонструвати проблемні моменти та наочно демонструвати напрямки на які треба звернути увагу при керуванні змістом проекту. В результаті аналізу, програма повинна видати 2 найймовірніші спрощені графи подій.

В програмі потрібно реалізувати зручний користувацький інтерфейс, за допомогою якого можна буде відобразити ймовірнісні сценарії виконання подій у вигляді ймовіврнісних графів, оцінити події за критеріями та виконати їх аналіз.

Редактор повинен дозволяти користувачеві маніпулювати графічними об’єктами для побудови графів, тобто змінювати їх розмір, місце розташування і властивості, якими володіє об’єкт. Граф складається з 8 типів вершин, які пред-

ставляють собою події з вірогідністю виконання та ребер – зв’язків, які зв’язують події між собою та мають вагу: «та», «або» чи виключне «або». Граф будуватиметься у вигляді ієрархії, тому події будуть розташовуватись на рівнях, а зв’язки будуть поєднувати події, які знаходяться на сусідніх рівнях.

Для виконання етапів аналізу планування від досягнутого буде використовуватись метод попарних порівнянь Т. Сааті.

Програма повинна надавати користувачеві можливість виконання перерахованих нижче функцій: створення бази для нового ймовірнісного графа з проектного беклогу системи керування проектами Mantis; додавання/видалення завдання-події; редагування атрибутів події з можливістю визначення експертом ймовірностей тої чи іншої події; групування подій за ознаками; потрібно створити подію перерахунку ймовірнісної моделі кожної ночі за cron, щоб не створювати зайвого навантаження на систему вдень коли кількість користувачів є великою.

Більш детальну інформацію можна знайти в технічному завданні, представленому у додатку А.

2 РОЗРОБКА ЙМОВІРНІСНОЇ МОДЕЛІ УПРАВЛІННЯ ЗМІСТОМ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОБЛІКУ ЕНЕРГІЇ.

Розглянемо приклад одного з варіантів побудови альтернативних графів. На першому етапі створення сценарію інноваційної розробки на змістовному рівні визначається структурна схема, в рамках якої досліджуваний об'єкт розчленовується на укрупнені елементи (групи робіт). Характер такого розбиття специфічний для різних видів розробок і визначається типом створюваного об'єкта.

Структурна схема будується зазвичай у вигляді графа Gs (Js, Us) типу дерево, де Js означає вузли, блоки та інші компоненти нової продукції, а Us - функціональні зв'язки між ними. При цьому виділяються вершини а∈ А∈ J_s, в яких можливі альтернативні рішення. Для чого використовується один з методів групових експериментальних оцінок, наприклад, метод комісій. Даний метод відноситься до колективних експертними оцінками, і відпрацювання думок фахівців в цьому випадку проводиться в основному самими експертами, які в процесі безпосереднього спілкування виробляють структуру розробки в цілому.

На наступному етапі побудови структурної схеми основним завданням є визначення можливо більшого набору альтернативних напрямків, і тому перевага віддається методам індивідуальної експертизи.

Отже, для всіх вершин графа а ∈ А визначається множина допустимих альтернатив, і кожна з них відображається дугою (а, е), е ∈ Js. Для кожної альтернативи (а, е) будується підграф Ge (JE, Ue) її реалізації. Підграф Ge (JE, Ue) може бути альтернативним стохастичним графом. Стохастичний альтернативний граф G (J, U), що відображає процес в цілому, виходить за допомогою об'єднання на основі графа Gs всіх графів типу Ge і послідовної заміною дуг (а, е) Us набором підграфов Ge, що відображають передбачувані для вершин альтернативи:

$$G(J,U) = G_s(J_s,U_s) \cup G_e(J_e,U_e).$$

Щоб побудувати граф треба виявити якнайбільше подій які призведуть нас до генеральної цілі проекту та занести їх у проектний беклог, в нашому випадку це буде програмне забезпечення з управління проектами Mantis; визначити характеристики завдань (назви, дії-нащадки по досягненню, ваги з врахуванням терміну реалізації, ймовірності здійснення подій); встановлення зв'язків між вершинами; визначення характеристик зв'язків - ваг ребер і дуг, визначення часу запуску аналізу для визначення альтернативних сценаріїв розвитку змісту в результаті аналізу.

Для правильного формулювання цілей дослідження і подальшого формулювання критеріїв попередньо необхідно проаналізувати дерево цілей. Для цього необхідно виконання процедур перерахованих нижче.

1. Використовуючи інформацію про виникнення проблеми, дані змістовного характеру, ієрархічну модель об'єкта (системи), провести генерацію цілей для об'єкта і кожного з його елементів.
2. Поставити цілі у відповідність елементам ієрархічної моделі об'єкта, тобто рознести мети за рівнями ієрархії і елементам дерева. При цьому верхнім рівням в дереві є рівень проєкту, що включає досліджуваний проєкт в якості підсистеми. Конкретніше число рівнів визначається специфікацією завдання і обсягом наявної інформації.

З позицій теоретико-множинного підходу дерево цілей це зв'язний граф, який в свою чергу характеризується зв'язковою множиною подій (цілей)

$$\overline{E} = \{e_1, e_2, ..., e_i, ..., e_m\},$$

Враховуючи, що граф має ієрархічну структуру, то вихідну множину станів можна представити у вигляді зв'язних підмножин, що представляють завдання на кожному рівні ієрархії:

$$\overline{E} = \{\overline{E}_0, \overline{E}_1, ..., \overline{E}_i, ...\overline{E}_k\},$$

де $\overline{E}_0 = \{e_1^0, e_2^0, ..., e_i^0, ..., e_n^0\}$, $\overline{E}_1 = \{e_1^1, e_2^1, ..., e_i^1, ..., e_m^1\}$, …, $\overline{E}_k = \{e_1^k, e_2^k, ..., e_i^k, ..., e_l^k\}$.

Градуювання розглянутих характеристик у відповідності з різними сценаріями, наведена в таблиці 2.1. Шкала градуйована в цілих числах від -5 до +5. Нуль відповідає збереженню існуючого положення, позитивні цілі числа показують різні ступені «збільшення», а негативні відповідно «зменшення». Ваги сценаріїв і узагальнена вага розраховуються нижче.

Таблиця 2.1 Сім сценаріїв і градуювання їх характеристик за шкалою -5 ... +5

	№ сценарія	1	2	3	4	5	6	7	Загальна вага
	Вага	0.11	0.15	0.26	0.1	0.21	0.11	0.03	
	Характеристика								
Економічні	1	1	1	4	-5	4	2	-4	1,74
	2	0	1	3	-5	2	1	-4	0,84
	3	0	0	2	0	1	2	0	0,95
Інституційні	1	-1	-1	2	-4	2	0	0	0,28
	2	0	0	2	-2	1	1	0	0,64
	3	0	0	2	-2	1	0	0	0,53
Технологічні	1	-1	-1	3	-4	2	-1	0	0,54
	2	-1	-1	2	-4	1	0	0	0,03
	3	0	0	1	0	1	0	0	0,47
Соціальні	1	-1	-1	1	-4	0	0	0	-0,4
	2	-1	-1	0	-2	0	0	0	-0,46
	3	0	0	0	0	0	0	0	0

Побудуємо матрицю попарних порівнянь щодо впливу чинників на розвиток холдингу (таблиця 2.2).

Таблиця 2.2 Матриця попарних порівнянь факторів

Фактори оточення	Економічні	Інституційні	Технологічні	Соціальні			d	Ω
Економічні	1	7	1/8	1			0,88	0,47
Інституційні	1/7	1	1/5	1/2			0,01	0,01
Технологічні	1	5	1	2			10,00	5,39
Соціальні	1	2	1/2	1			1,00	0,54

$D = 1,86$

Наступним етапом було знаходження важливості стейкхолдерів щодо факторів, що впливають на розвиток агенції. Це здійснюється шляхом множення матриці власних векторів акторів щодо кожного фактора третього рівня на власний вектор, отриманий для другого рівня.

Так як на замовника припадає 75% (42 +33) впливу на чотири основні чинники, що впливають на агенцію, було вирішено використовувати тільки ці дві діючі сторони для отримання ваг сценаріїв. Якщо використовувати більше діючих сторін, то обчислювальна процедура буде тією ж, що і показана нижче, тільки зросте обсяг обчислень.

Тепер знайдемо найважливіші цілі двох діючих сторін - замовника і агенції, що виконує роботи по розробці програмних продукту для компані. Для цього помножимо власний вектор цілей на відповідну вагу діючих сторін, який був тільки що обчислений.

Для замовника:

$$0{,}42 \times \begin{bmatrix} 0{,}26 \\ 0{,}35 \\ 0{,}6 \\ 0{,}24 \\ 0{,}5 \\ 0{,}26 \end{bmatrix} = \begin{bmatrix} 0{,}11 \\ 0{,}15 \\ 0{,}26 \\ 0{,}01 \\ 0{,}21 \\ 0{,}11 \end{bmatrix} \begin{matrix} \text{Репутація на біржі} \\ \text{Нарощування кількості користувачів} \\ \text{Успішний вихід на ринок програмного продукту} \\ \text{Використання найновіших технологій} \\ \text{Добробут} \\ \text{Пошук інвесторів} \end{matrix}$$

Для агенції:

$$0{,}33 \times \begin{bmatrix} 0{,}26 \\ 0{,}35 \\ 0{,}6 \\ 0{,}24 \\ 0{,}5 \\ 0{,}26 \end{bmatrix} = \begin{bmatrix} 0{,}09 \\ 0{,}11 \\ 0{,}2 \\ 0{,}08 \\ 0{,}16 \\ 0{,}09 \end{bmatrix} \begin{matrix} \text{Людські ресурси} \\ \text{Технологія} \\ \text{Прибуток} \\ \text{Інтереси} \\ \text{Стабільність розвитку} \\ \text{Модернізація обладнання} \end{matrix}$$

Звідси можна побачити, що найбільш впливовими цілями для замовника є успіх на ринку і добробут, а для агенції - прибуток і стабільність у розвитку. Використовуючи ці чотири цілі і нормалізуючи їх ваги отримуємо вектор ваг.

$$\begin{bmatrix} 0{,}26 \\ 0{,}21 \\ 0{,}2 \\ 0{,}16 \end{bmatrix} \begin{matrix} \text{Добробут} \\ \text{Успішний вихід на ринок програмного продукту} \\ \text{Прибуток} \\ \text{Стабільність розвитку} \end{matrix}$$

Застосуємо цей вектор для отримання ваг сценаріїв. Сценарії були зважені стосовно чотирьох цілей. Для отримання ваг сценаріїв помножимо матрицю власних векторів сценаріїв на вектор цілей. Цей добуток являє собою узагальнені ваги сценаріїв.

Сценарій	Добробут	р П	Прибуток	Розвиток		Вектор ваг цілей		Вага сценаріїв
1	0,24	0,36	0,32	0,18		0,1		0,11
2	0,25	0,37	0,33	0,2		0,13		0,15
3	0,24	0,37	0,33	0,19	×	0,23	=	0,26
4	0,24	0,36	0,33	0,18		0,09		0,1
5	0,26	0,38	0,33	0,2		0,18		0,21
6	0,23	0,35	0,3	0,22		0,10		0,11
7	0,35	0,45	0,4	0,3		0,02		0,03

Потім використовуємо ці ваги, заповнивши рядок «ваги сценаріїв» таблиці 1.3 для отримання значень змінних, які записуються в правий стовпець, як було зазначено раніше.

Як видно, найбільшу вагу має третій сценарій. Тепер отримаємо узагальнений сценарій за шкалою вимірів. Ця шкала вийде шляхом підсумовування добутків ваг сценаріїв на відповідні значення характеристик: (+1) (0,11) + (+1) (0,15) + (4) (+0,26) + (-5) (0,1) + (+4) (0,21) + (+2) (0,11) + (-4) (+0,03) = 1,74.

Цей результат занесений в крайній праворуч стовпець таблиці. Аналогічно отримуємо інші елементи цього стовпця.

3. ПРОЄКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ УПРАВЛІННЯ ЗМІСТОМ ПРОГРАМНОГО ПРОЄКТУ ДЛЯ ОБЛІКУ ЕНЕРГІЇ

3.1 Ескізний проєкт

Вибір акторів та варіантів використання

1. Експерт – оцінює критерії;
2. Менеджер – особа, яка приймає рішення на основі аналізу ймовірісного графу, який був оцінений експертом.

3.1.2 Діаграма варіантів використання

Діаграма варіантів використання представлена на рисунку 3.1.

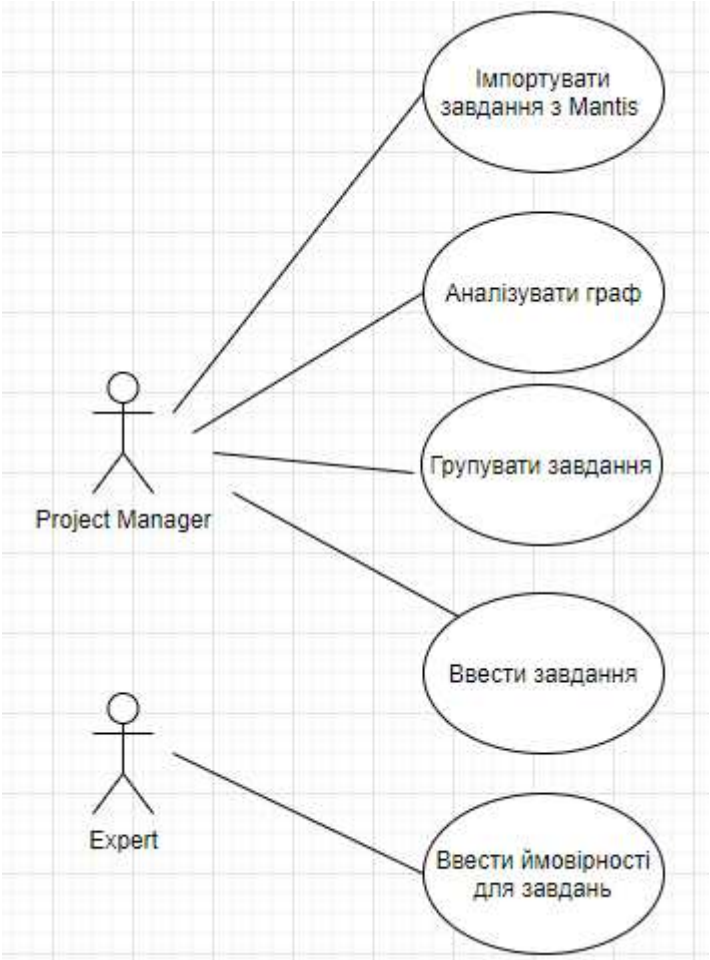

Рисунок 3.1 – Діаграма варіантів використання програмного забезпечення управління змістом.

Усі актори активні і відіграють важливу роль в системі. Також функції кожного з акторів може виконувати одна і та ж людина.

3.1.4 **Опис варіантів використання**

Виявлені варіанти використання зведені у таблиці 3.1.

Таблиця 3.1 – Виявлені варіанти використання.

Основні актори	**Найменування**	**Формулювання**
Project Manager	Побудувати сценарій	Дозволяє побудувати сценарій за допомогою інструментів.
Project Manager	Імпортувати завдання з Mantis (Import tasks from Mantis)	Завантажує сценарій з файлу, який був экспортований у csv форматі з системи Mantis.
Expert	Оцінити критерії (CRUD probability)	Дозволяє установити та редагувати ймовірності виконання кожного критерію для правильного аналізу графу.
Project Manager	Створити події (CRUD tasks)	Дозволяє створювати, редагувати та видаляти події
Project Manager	Створити розклад аналізу графу (Create the analizing schedule)	Дозволяє створити подію перерахунку ймовірнісної моделі за cron таблицею
Project Manager	Ан**алізу**вати граф (Analizing graph)	Запускає аналіз графу та видає найбільш успішні сценарії.
Project Manager	**Групувати завдання** (Group tasks)	Дозволяє визначити групу для події

Конкретизуємо кожний варіант використання:

1.

Найменування: Побудувати сценарій.

Основна дійова особа: Project Manager.

Зв'язок з іншими варіантами використання: Включає варіант використання «Зберегти сценарій»;

Короткий опис:

Варіант використання «Побудувати сценарій» дозволяє створити сценарій за допомогою виконання маніпуляцій з елементами сценарію. Елементами є: подія (ціль, критерій, альтернатива), зв'язок, групування подій та рівень. Ці елементи можна до-давати на поле, переміщати за допомогою маніпулятору миші, редагувати та видаляти. Верхівкою графу є ціль, яка відображає головну ціль проєкту. Альтернативи – це елементи нижнього рівня і представляють собою множину варіантів досягнення цілі. Елементи які знаходяться між альтернативами і ціллю – критерії. Елементи, які мають однин батьківський вузол називаються групою зрівняння і можуть буті згруповані у разі необхідності. Кожен вузол має пріоритет від 0 до 1. Пріоритет цілі – 1.0, пріоритети критеріїв виставляють експерти, а пріоритет альтернатив вираховується. Між вузлами на сусідніх рівнях можна встановити зв'язок з вагою «та», «або» чи виключне «або».

Після створення сценарію він зберігається.

Початковий стан: Прецедент починається, коли в програмі було створено чисте поле для побудови сценарію.

Потоки подій:

Базовий потік – Побудова сценарію:

1) Project Manager обирає пункт «Створити сценарій»:
2) Відкривається чисте поле для побудови сценарію;
3) Project Manager додає рівні, події, зв’язки та групи подій;
4) Система відображає додані Project Managerем елементи на екрані;
5) Include «Зберегти сценарій».

Альтернативні потоки

1) Якщо у п. 3 базового потоку указати вже існуюче ім’я вузлу, то буде видане повідомлення про помилку, пропозиція для зміни імені та збереження події буде неможливе.

2) Якщо при створенні сценарію створити 2 зв’язки у яких указані однакові події, то один з них буде видалено.

Кінцевий стан: Побудований граф, збережений на накопичувачі.

2.

Найменування: Редагувати сценарій.

Основна дійова особа: Project Manager.

Зв'язок з іншими варіантами використання: Включає варіанти використання «Завантажити сценарій» та «Зберегти сценарій»

Короткий опис:

Дозволяє відредагувати раніше створений сценарій, за допомогою додавання нових або видалення чи редагування вже існуючих елементів сценарію.

Початковий стан: Прецедент починається, після завантаження сценарію Project Managerем.

Потоки подій:

Базовий потік – Створення нового сценарію:

1) Include «Завантажити сценарій»;

2) Project Manager додає, редагує чи видаляє рівні, події, зв’язки та групи подій;

3) Система відображає зроблені Project Managerем зміни на екрані;

4) Include «Зберегти сценарій».

Альтернативний потік

1) Якщо у п. 3 базового потоку указати вже існуюче ім’я вузлу, то буде видане повідомлення про помилку, пропозиція для зміни імені та збереження події буде неможливе.

2) Якщо при створенні сценарію створити 2 зв’язки у яких указані однакові події, то один з них буде видалено.

Кінцевий стан: Відредагований граф збережений на накопичувачі.

3.

Найменування: Зберегти сценарій.

Основна дійова особа: Project Manager, Експерт.

Зв'язок з іншими варіантами використання: Включається варіантами використання «Побудувати сценарій», «Редагувати сценарій» та «Оцінити критерії».

Короткий опис:

Зберігає поточний сценарій на накопичувач у вигляді файлу у певному форматі.

Початковий стан: Прецедент починається, після побудови графу або після оцінки критеріїв.

Потоки подій:

Базовий потік – Зберігання сценарію:

1) Project Manager обирає пункт «Зберегти сценарій»;
2) Відкривається діалогове вікно, для вибору місця збереження файлу;
3) Project Manager обирає каталог;
4) Вказує назву файлу, відповідно, для збереження;
5) Обирає пункт зберегти;
6) Діалогове вікно закривається після успішного збереження.

Альтернативний потік

1) Project Manager обирає пункт «Зберегти сценарій»;
2) Відкривається діалогове вікно, для вибору місця збереження файлу;
3) Project Manager обирає каталог;
4) Вказує назву файлу, відповідно, для збереження;
5) Обирає пункт зберегти;
6) Якщо в обраному каталозі є файл з такою, назвою, то система видає попередження, та пропонує замінити файл;
7) Project Manager обирає пункт замінити файл;
8) Діалогове вікно закривається після успішного збереження.

Кінцевий стан: На накопичувачі створений файл, у вигляді якого збережений граф.

Найменування: Завантажити сценарій.

Основна дійова особа: Project Manager, Експерт.

Зв'язок з іншими варіантами використання: Розширяє варіант використання «Оцінити критерії» і включається варіантами використання «Виконати аналіз сценарію» та «Редагувати сценарій».

Короткий опис:

Завантажує, збережений раніше на накопичувачі, сценарій використання, та відображає його на новому полі.

Початковий стан: Прецедент починається, після того як Project Manager обере пункт «Завантажити сценарій»

Потоки подій:

Базовий потік – Завантаження сценарію:

1) Project Manager обирає пункт «Завантажити сценарій»;
2) Відкривається діалогове вікно, для вибору файлу;
3) Project Manager обирає файл;
4) Обирає пункт завантажити;
5) Діалогове вікно закривається після успішного завантаження;
6) Завантажений сценарій відображається на новому полі побудови сценарію.

Альтернативні потоки

1) Якщо обраний файл виявляється не файлом даної програми, то після виконання п. 4 видається відповідне повідомлення про помилку і потік повертається до пункту 3.

2) Якщо завантаження файлу відбувається під час побудови іншого графу, то після п. 4 система видає попередження і пропонує зберегти поточний граф. Після збереження або відмові відбувається перехід до п. 5.

Кінцевий стан: В редакторі створено нове поле, на якому відображений граф з файлу.

5. *Найменування*: Оцінити події за критеріями.

Основна дійова особа: Експерт.

Зв'язок з іншими варіантами використання: Розширяється прецедентом «Завантажити сценарій» та включає «Зберегти сценарій».

Короткий опис:

Дозволяє установити ймовірності виконання кожної альтернативи для правильного аналізу графу.

Початковий стан: Прецедент починається, після того як Експерт завантажить сценарій або побудує його.

Потоки подій:

Базовий потік – Оцінка подій за критеріями:

1) Extend «Завантажити сценарій»;
2) Експерт обирає подію та викликає команду «оцінити альтернативи»;
3) Відкривається форма для оцінки альтернатив (нащадків);
4) Експерт додає критерії для оцінки альтернатив;
5) Критерії зберігаються в обраної події;
6) Експерт обирає команду «порівняти критерії»;
7) Відкривається таблиця для порівняння;
8) Експерт заповнює таблиці порівнянь критеріїв та альтернатив;
9) Виконує команду «порівняти»;
10) Система виконує аналіз;
11) Задає вагу альтернативам;
12) Відображає вагу для кожної альтернативи на графічній панелі;
13) Експерт виходить з форми оцінки альтернатив;
14) Include «Зберегти сценарій».

Кінцевий стан: На накопичувачі збережено сценарій з оціненими подіями експертом.

6.

Найменування: Виконати аналіз сценарію.

Основна дійова особа: Project Manager.

Зв'язок з іншими варіантами використання: Включає варіант використання «Завантажити сценарій».

Короткий опис:

Виконує аналіз графу та видає найбільш успішні сценарії.

Початковий стан: Прецедент починається, після завантаження сценарію.

Потоки подій:

Базовий потік – Виконання аналізу сценарію:

1) Include «Завантажити сценарій»
2) Project manager обирає пункт «Виконати аналіз»;
3) Система аналізує сценарій, та відображає декілька найбільш успішні спрощені графи та один найуспішніший;

Альтернативні потоки

1) Якщо в сценарії не оцінені критерії або оцінені не вірно, то після п. 2 система видає попередження і пункт 2 не виконується;
2) Якщо в сценарії є подія без зв’язку, то система видає відповідне попередження, пропонує продовжити чи відмінити аналіз.

Кінцевий стан: Система виводить на екран 3 найуспішніші спрощені графи.

7. *Найменування*: Затвердити спрощений сценарій змісту проєкту.

Основна дійова особа: Project Manager.

Зв'язок з іншими варіантами використання: Включає варіант використання «Виконати аналіз сценарію».

Короткий опис:

Затверджує спрощений граф з представлених після аналізу.

Початковий стан: Прецедент починається, після виконання аналізу сценарію.

Потоки подій:

Базовий потік – Затвердження спрощеного сценарію змісту проєкту:

1) Include «Виконати аналіз сценарію»;
2) Project Manager обирає один із спрощених сценаріїв представлених після аналізу;
3) Виділяє його, як основний потік подій для даного проєкту;
4) Система фіксує основний потік подій для проєкту;
5) Extend «Зберегти сценарій».

Кінцевий стан: На накопичувачі збережено сценарій з виділеним основним потоком подій.

3.1.4 Розробка прототипу інтерфейсу

На основі опису варіантів використання був розроблений прототип інтерфейсу, який представлений на рисунках 3.2– 3.6.

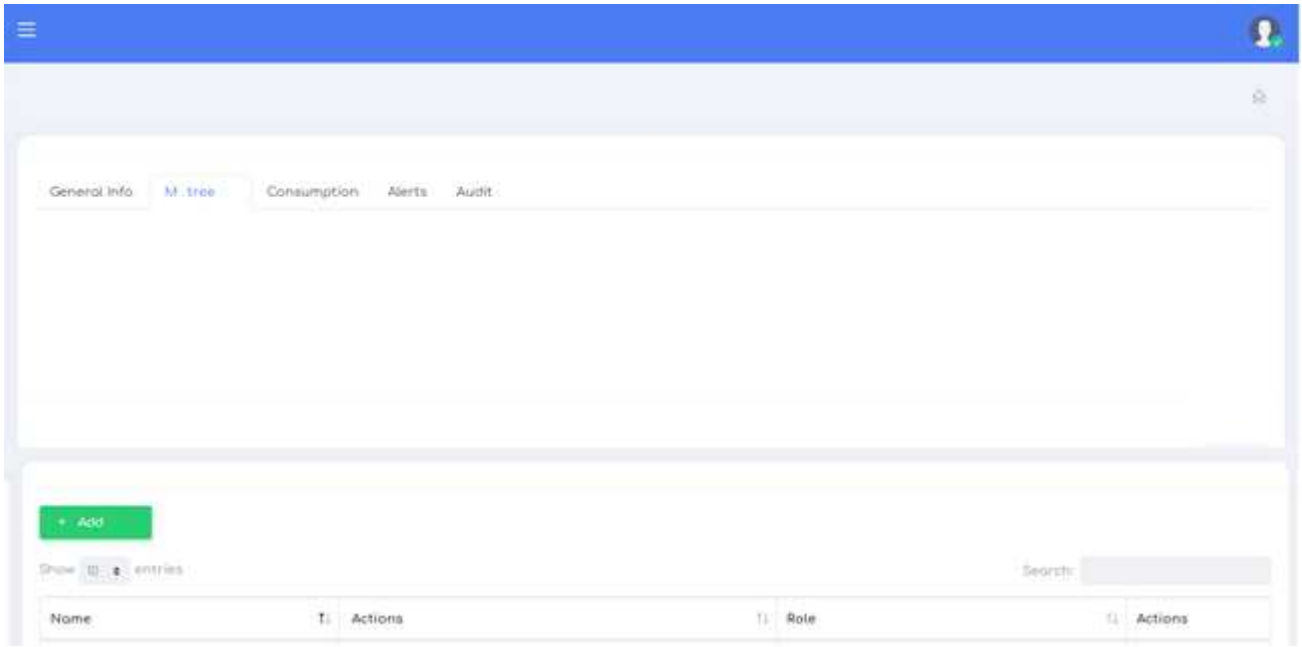

Рисунок 3.2 – Головне вікно програми

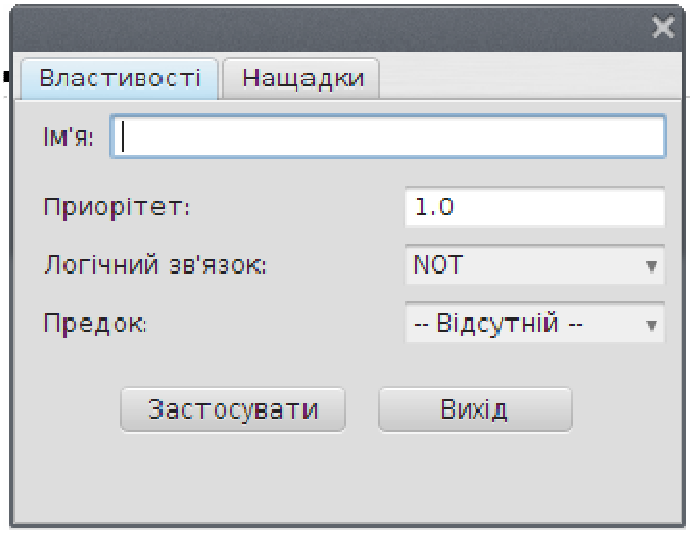

Рисунок 3.3 – Властивості події

3.2 Технічний проєкт

3.2.1 Вибір архітектурного стилю

Програмне забезпечення, яке розробляється найбільш відповідає архітектурному стилю MVC (модель – представлення-контроль) – ціль якої полягає в розподілі бізнес-логіки (моделі) від її візуалізації (представлення). За рахунок такого розділення підвищується можливість повторного використання. То ж данні для розроблюваного ПЗ розділяються на три частини:

- Модель, яка представляє данні у вигляді набору елементів на основі яких будується граф та методи для роботи з цим графом.

- Представлення – відповідає за відображення (візуалізацію) програми у вигляді форми з графічними елементами, на якій відображаються різноманітні інструменти для роботи з графами та відповідно самі графи.

- Контролер – забезпечує зв’язок між користувачем та системою: контролює введення даних користувачем та використовує модель і представлення для побудови та аналізу графів.

Архітектура програмного забезпечення у вигляді діаграми UML представлена на рисунку 3.7.

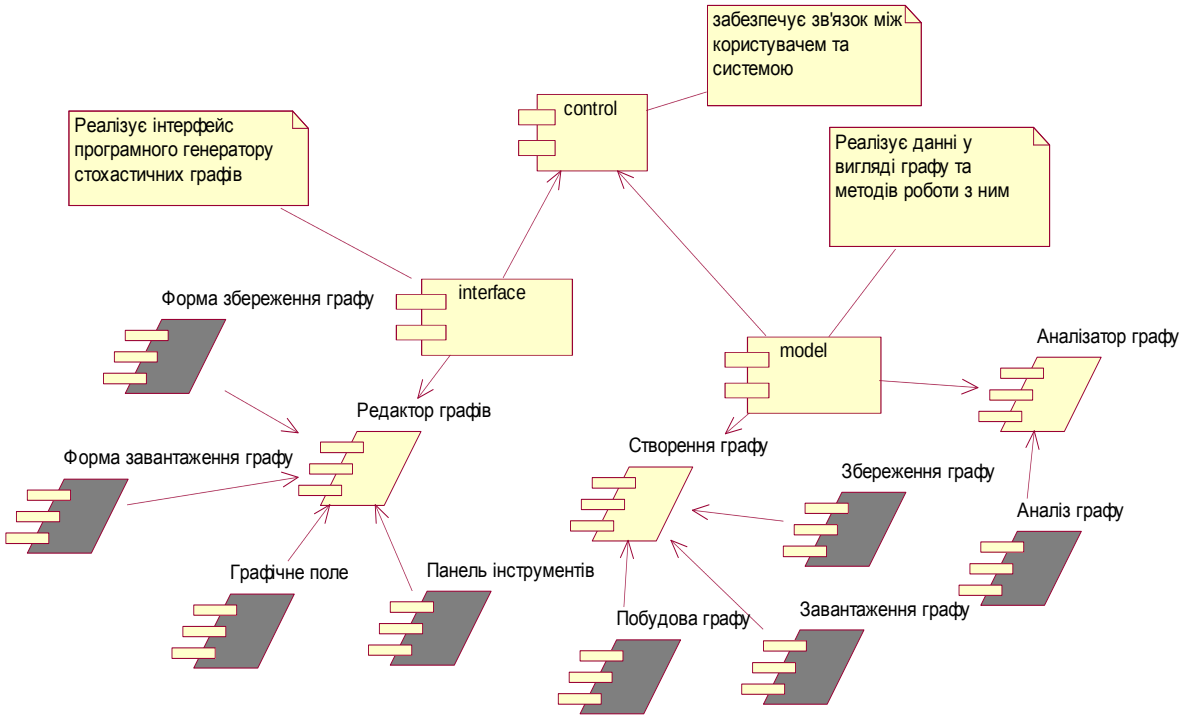

Рисунок 3.7 – Діаграма компонентів програмного забезпечення управління змістом

3.2.2 Модель аналізу

Модель аналізу для реалізації варіанта використання «Створити граф» розроблена і представлена у вигляді діаграми класів на рисунку 3.8 та у вигляді діаграми взаємодії на рисунку 3.9.

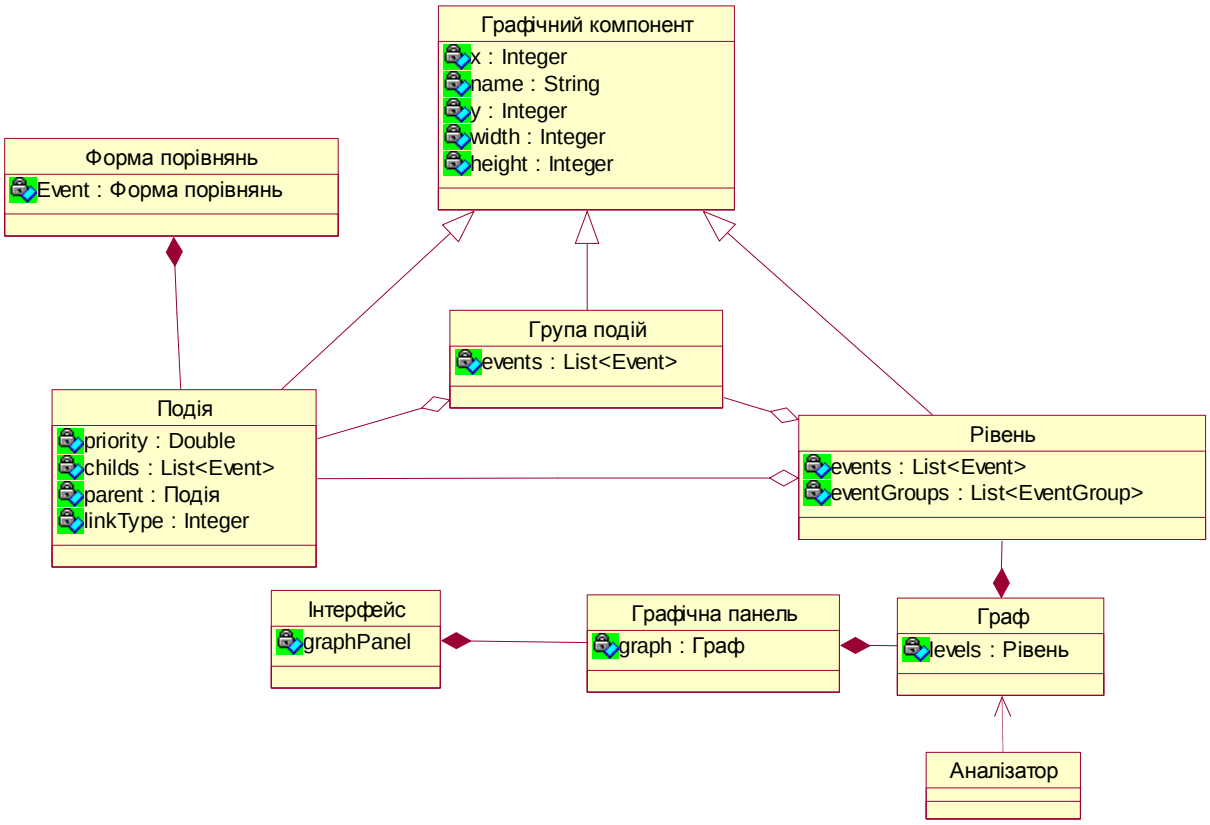

Рисунок 3.8 – Діаграма класів програмного забезпечення управління змістом проекту з аналізу енергії.

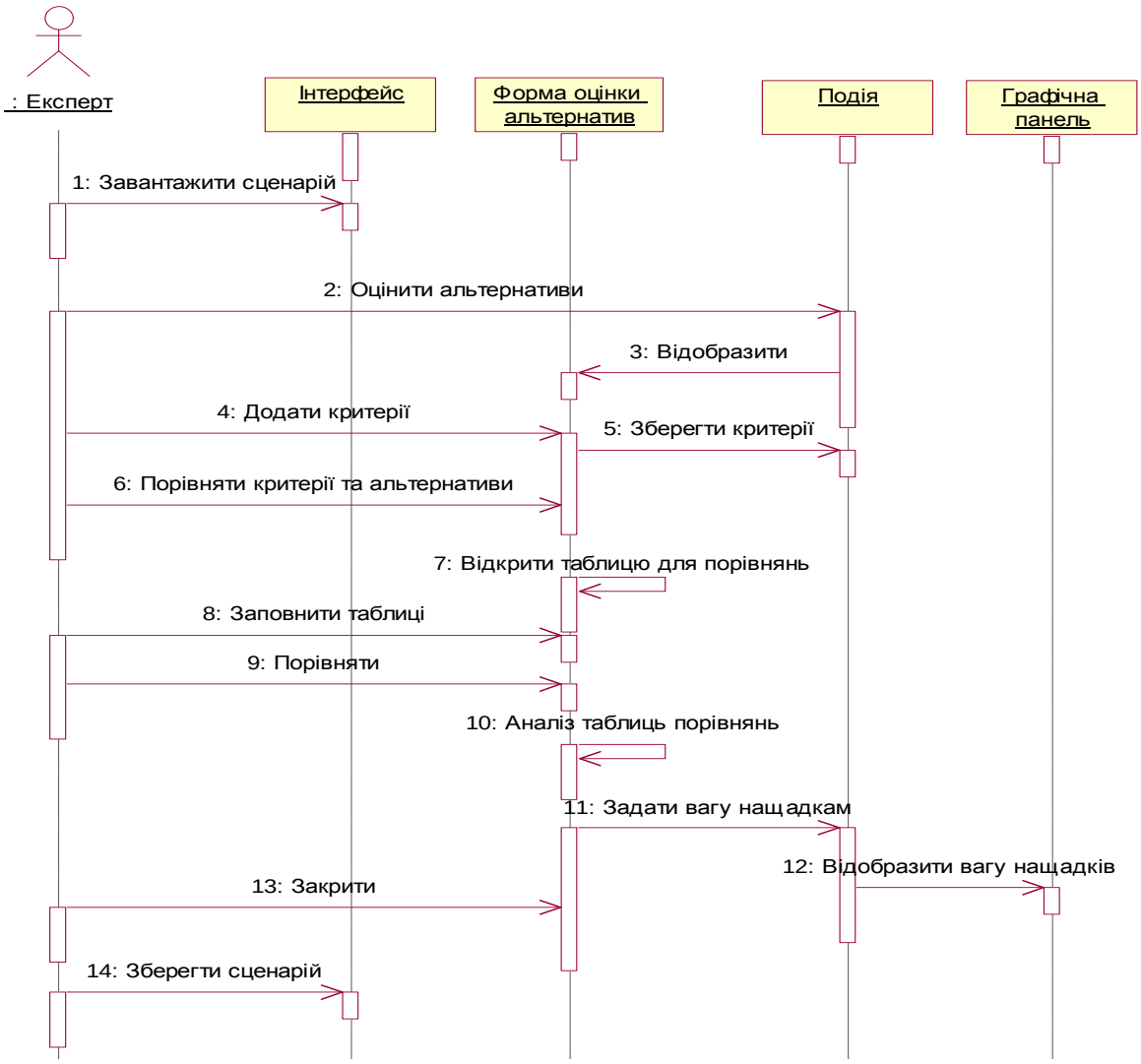

Рисунок 3.9 – Діаграма взаємодії програмного забепечення з управління змістом.

3.2.3 Проєктна модель

Проєктна модель реалізації відповідно до моделі аналізу розроблена і представлена у вигляді діаграми класів на рисунку 3.10. Та у вигляді діаграми взаємодії представлені варіанти використання «Оцінити критерії», «Завантажити сценарій» та «Виконати аналіз» на рисунку 3.11, 3.12, 3.13.

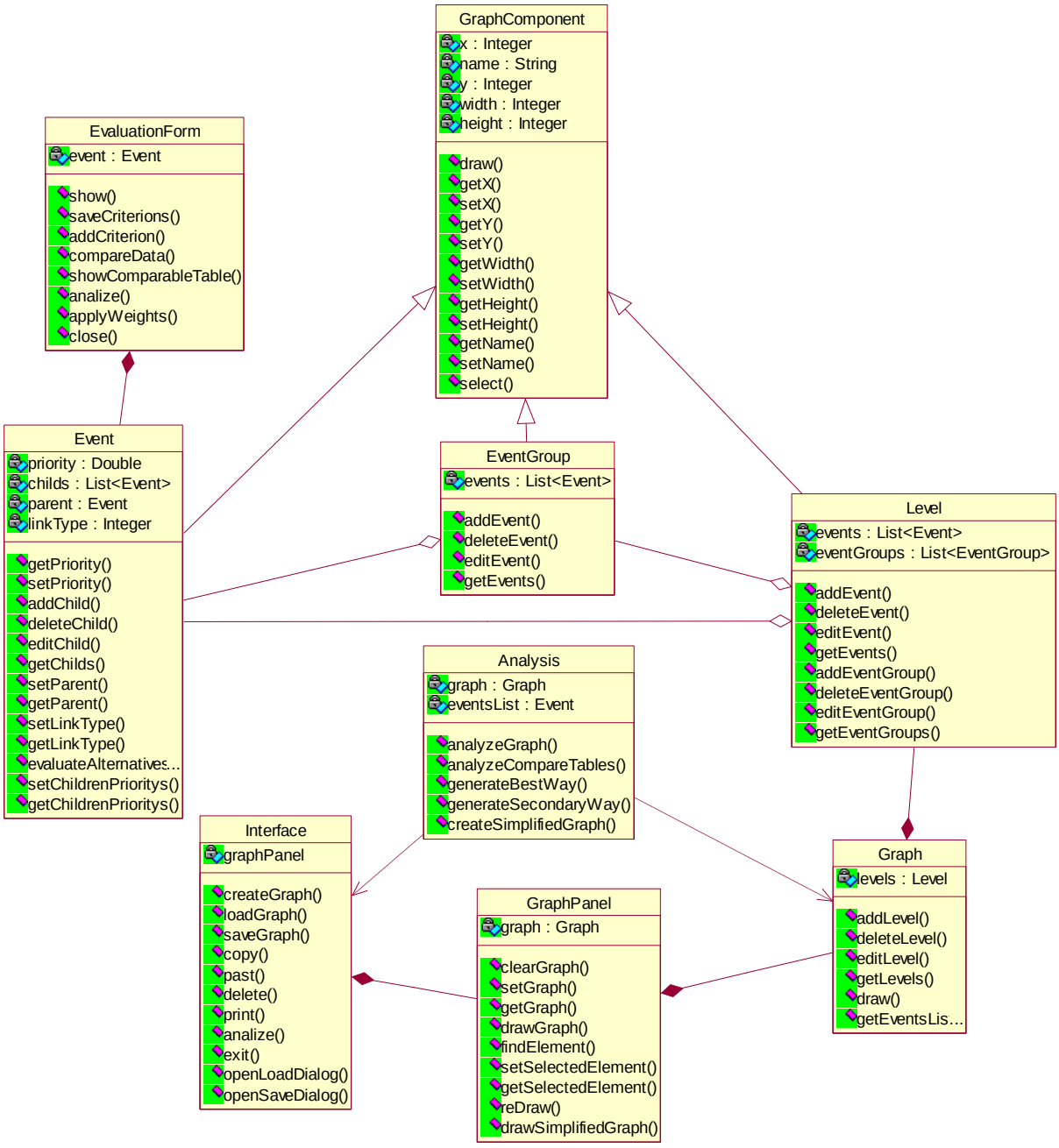

Рисунок 3.10 – Діаграма класів програмного забезпечення з управління змістом

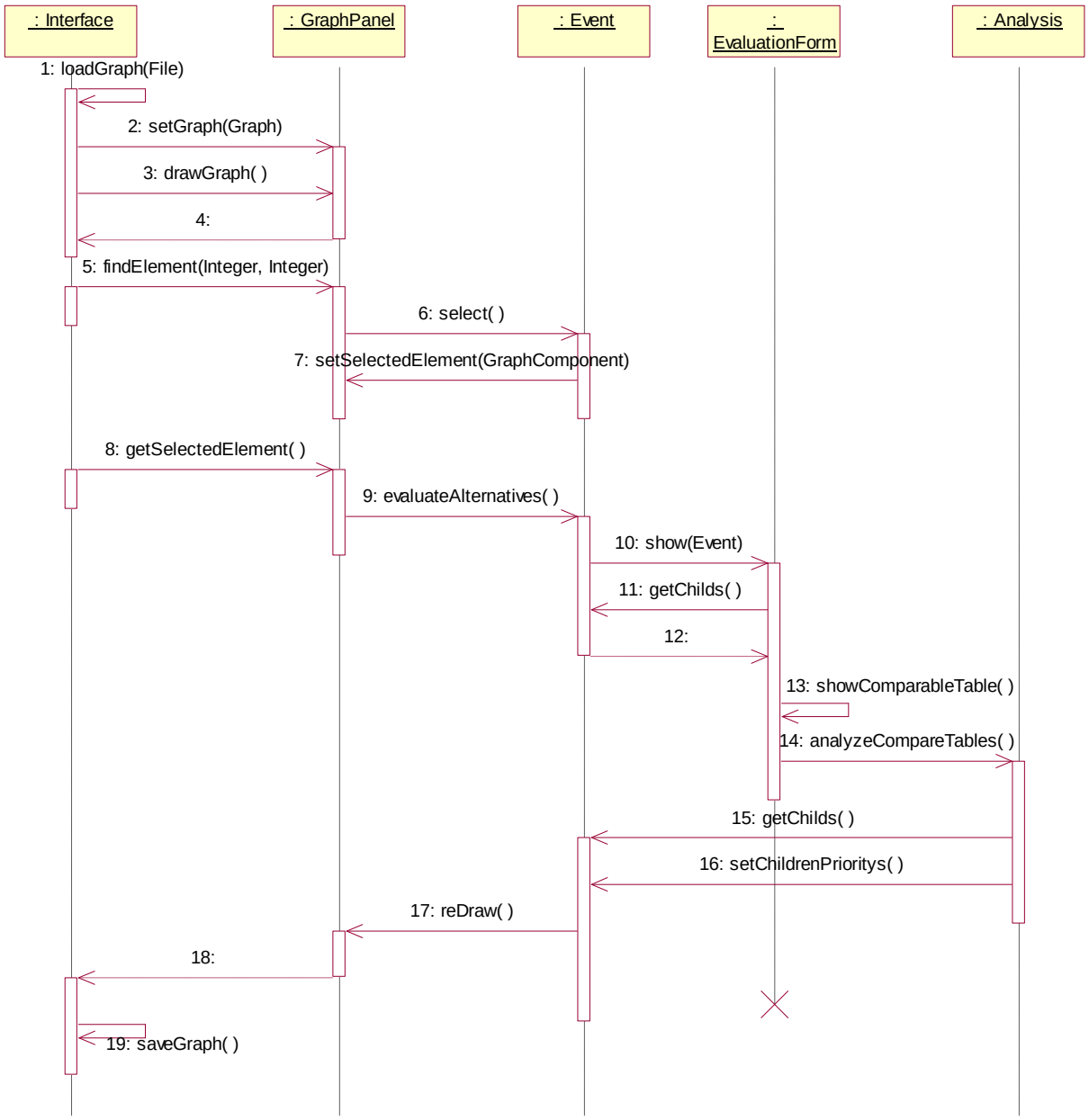

Рисунок 3.11 – Діаграма використання програмного забезпечення з управління змістом

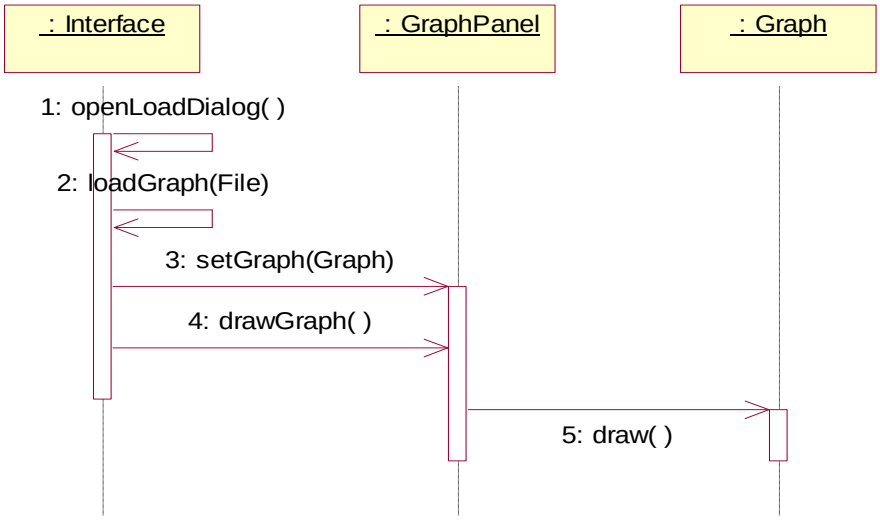

Рисунок 3.12 – Діаграма взаємодії для варіанта використання «Завантажити сценарій».

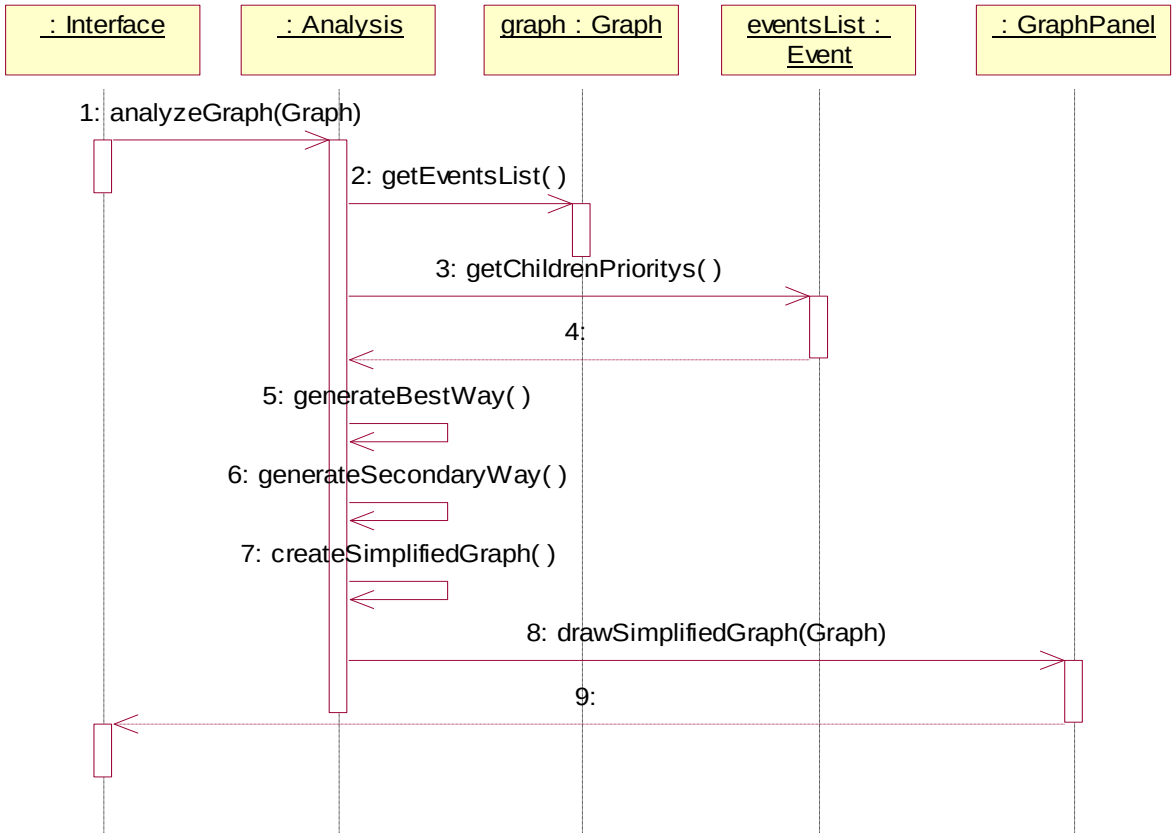

Рисунок 3.13 – Проєктна модель варіанта використання «Виконати аналіз» у вигляді діаграми взаємодії.

3.3 Робочий проєкт

Кодування програми проводиться на основі специфікації програмних модулів, представлених у підрозділі технічного проєкту. Повний текст програми наведений в Додатку Б.

4 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ УПРАВЛІННЯ ЗМІСТОМ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОБЛІКУ ЕНЕРГІЇ

Результатом розробки стало програмне забезпечення побудови ймовірнісного графу змісту програмного забезпеченняя на базі csv файлів, що експортовано з системи управління проектами Mantis із зазначенням події, її експертної оцінки та інформації про те які роботи було виконано та якими членами команди для її досягнення, які ризики її спіткали та які ресурси було використано для її досягнення що за

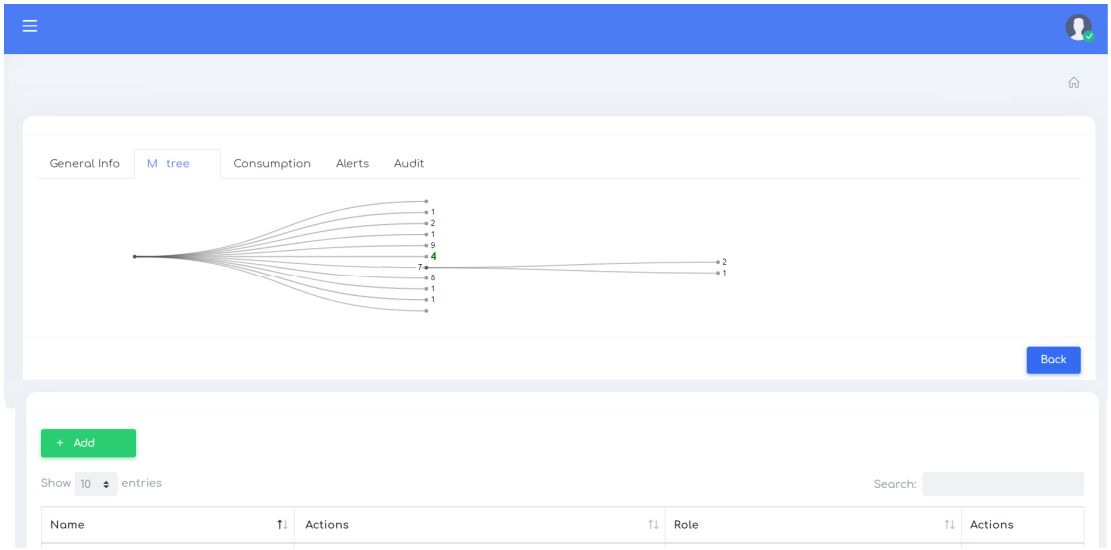

Рисунок .4.1 Ймовірнісний граф розвитку змісту проекту EnergyDB

5 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ РОЗРОБКИ І ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ АНАЛІЗАТОРУ ГРАФОВИХ МОДЕЛЕЙ

Найбільш важливим моментом для розробника, з економічної точки зору, є процес формування вартості програмного продукту. Він являє собою дуже специфічний товар з безліччю особливостей.

Створення програмного продукту вимагає одноразових витрат на його розробку, придбання необхідних технічних засобів і поточних витрат на функціювання продукту. Економія від функціювання програмного продукту визначається з урахуванням витрат на його експлуатацію. Відношення цієї економії до витрат на створення програмного продукту характеризує економічну ефективність капітальних вкладень. Економічні показники визначаються по оптовим цінам, тарифам і ставкам заробітної плати, що діють на момент розрахунку.

5.1 Розрахунок витрат на створення й експлуатацію програми

Витрати на розробку продукту складаються з витрат на зарплату розробника, на амортизацію ЕОМ, на якій виконується розробка, на експлуатацію цієї ЕОМ, на засоби розробки та витрат на матеріали і комплектуючі.

Розробка програмного забезпечення виконується програмістом, місячний оклад якого складає 10 000 грн. Додаткова заробітна плата складає 20% від основної. Виходячи з цього, основна і додаткова заробітна плата розробника системи складає 12000 грн/міс, а вартість ПК складає 6300 грн (на базі Intel Core i5). При

вартості кіловат-години електроенергії рівної 0,65 грн, розраховується вартість розробки програми. Витрати на допоміжні матеріали приведені в табл. 5.1.

Таблиця 5.1 — Витрати на допоміжні матеріали

Пункт витрат	Сума
Папір	80,00
Заправлення картриджа до принтера	125,00
CD	9,20
Непередбачені витрати	200,00
Всього	414,20

Вартість програми розрахуємо по формулі:

$$C_{пр} = (З_{зп} + З_{сз} + З_{зг} + З_{е}) * Т + З_{м} \quad (5.1)$$

Таблиця 5.2 - Витрати на розробку програмного продукту

Найменування витрат	Позначення	Кількість
Тривалість розробки	**Т**	2 міс.
Основна і додаткова заробітна плата	$\mathbf{З_{зп}}$	12 000 грн
Відрахування на соціальні заходи	$\mathbf{З_{сз}}$	1 800 грн
Загальногосподарські витрати	$\mathbf{З_{зг}}$	1 000 грн
Витрати на основні і допоміжні матеріали	$\mathbf{З_{м}}$	414,20 грн
Витрати на електроенергію	$\mathbf{З_{е}}$	158,40 грн

Відрахування на соціальні заходи ($З_{сз}$) складає 15% від основної і додаткової заробітної плати; загальногосподарські витрати ($З_{зг}$) складає 10% від основної заробітної плати.

При споживанні потужності 0,5 кВт, тривалості роботи за місяць, що дорівнює 22 * 8 = 176 годин, вартості кіловат-години електроенергії 1,8 грн,

$$З_е = 176 * 1,8 * 0,5 = 158,4 \text{ грн.}$$

За формулою 4.1 вартість програми дорівнює:

Спр = (12 000 + 1 800 + 1 000 + 158,40) * 2 + 414,20 = 30 331 грн.

Амортизаційні відрахування на устаткування складають 60% балансової вартості в рік:

$$А_{об} = 6\ 300 * 0,6 = 3\ 780 \text{ грн.}$$

У масштабах підприємства річні витрати на основні та допоміжні матеріали визначаються в розмірі 5% вартості основного устаткування:

$$В_м = 6\ 300 * 0,05 = 315 \text{ грн.}$$

Річний обсяг робіт ПК у годинах визначається в такий спосіб:

$$Ф_м = 264,5 * Т_з \quad (5.2)$$

де Тз — середнє місячне навантаження устаткування (близько 4 годин);

264,5 — середня кількість робочих днів у році.

Отже, річний обсяг роботи ПК складає:

$$Ф_м = 264,5 * 4 = 1058 \text{ годин.}$$

Витрати на електроенергію $З_е$ складуть:

$$З_е = 1058 * 0,5 * 1,8 = 952,2 \text{ грн.}$$

Експлуатаційні витрати для ПК за рік складуть:

$$З_{зр} = 3\ 780 + 315 + 952,2 = 5\ 047,2 \text{ грн.}$$

Отже, у перший рік витрати на створення й експлуатацію програми складуть:

$$З_{се} = 30\ 128,6 + 5\ 047,2 = 35\ 175,8 \text{ грн.}$$

5.2 Економічна ефективність розробки і впровадження програми

Основним показником економічної ефективності функціювання програмного продукту є підвищення ефективності керування інформацією у вигляді зниження витрат на керування при одночасному збільшенні швидкості і якості одержання потрібного результату.

Крім багатьох інших негативних ефектів, ручна обробка інформації спричиняє наступні негативні економічні ефекти:

3 високі витрати на складування паперових документів;

4 підвищені витрати на канцтовари;

5 витрати, пов'язані з роботою виявлення раніше допущених помилок (людський фактор).

До числа основних факторів, що визначають приріст прибутку в зв'язку з упровадженням програми, відносяться підвищення продуктивності праці та вивільнення робочого часу.

Крім того, не піддається прямій грошовій оцінці підвищення оперативності керування, якість одержуваних результатів, поліпшення організації праці і т.д.

Обов'язковою умовою визначення економічної ефективності програми є порівнянність усіх показників у часі, за цінами й іншими нормами, використовуваними для визначення показників, за змістом і колом елементів витрат.

Визначимо пряму економічну ефективність, ґрунтуючись на тому, що впровадження програмного продукту вивільняє 0,3 працівника (за експертною оцінкою фахівців підприємства).

Зарплата 0,3 працівника в рік складає:

$$10\,000 * 12 * 0{,}3 = 36\,000 \text{ грн.}$$

Річний економічний ефект розраховується по формулі:

$$Э_{год} = \Delta C_n – E_n * k \quad (5.3)$$

де ΔC_n – вивільнені кошти після впровадження програмного продукту (36 000 грн) мінус експлуатаційні витрати (4 438,85 грн);

E_n – коефіцієнт ефективності (дорівнює коефіцієнту амортизації – 0,6);

k – одноразові витрати на впровадження продукту (30 331 грн).

$$Э_{год} = 31\,561{,}15 – 0{,}6 * 30\,331 = 13\,362{,}55 \text{ грн.}$$

Строк окупності програми розраховується по формулі:

$$T = \frac{k}{\Delta C_n} = \frac{30331}{31561{,}15} = 0{,}96 \approx 1\,рік$$

Отже строк окупності програмного продукту складає приблизно 1 рік.

Висновки за розділом 5.

У цьому розділі були здійснені розрахунки витрат економічної ефективності та терміну окупності програми для управління змістом програмного забезпечення. Виходячи з розрахунків, впровадження програми окупить себе протягом 12 місяців. Таким чином, його розробка є економічно обґрунтованою.

6 ОХОРОНА ПРАЦІ

Кожен має право на належні, безпечні і здорові умови праці. Це гарантує нам Конституція України (ч. 4 ст. 43). Більш детальні вимоги щодо охорони праці, зокрема охорони праці офісних працівників, містять Кодекс законів про працю, Закон України «Про охорону праці», а також інші підзаконні нормативно-правові акти. У відповідності до вимог ст. 153 Кодексу законів про працю України та ст. 6 Закону України «Про охорону праці» на всіх підприємствах, в установах, організаціях створюються безпечні і нешкідливі умови праці. Забезпечення безпечних і нешкідливих умов праці покладається на власника або уповноважений ним орган. А згідно з ч. 1 ст. 13 Закону України «Про охорону праці» роботодавець зобов'язаний створити на робочому місці в кожному структурному підрозділі умови праці відповідно до нормативно-правових актів, а також забезпечити додержання вимог законодавства щодо прав працівників у галузі охорони праці.

Робочі місця офісних працівників, обладнані персональними комп'ютерами (далі – робочі місця), повинні відповідати вимогам «Правил охорони праці під час експлуатації електронно-обчислювальних машин», затверджених Наказом Державного комітету України з промислової безпеки, охорони праці та гірничого нагляду від 26.03.2010 року № 65 (Правила), та «Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин», затверджених постановою Головного державного санітарного лікаря України від 10.12.98 N 7 (ДСанПіН 3.3.2-007-98). Правила поширюються на всіх суб'єктів господарювання незалежно від форм власності, які у своїй діяльності здійснюють роботу, пов'язану з персональними комп'ютерами, у тому числі на тих, які мають робочі місця, обладнані персональними комп'ютерами і периферійними пристроями. Зазначені нормативно-правові акти встановлюють санітарно-гігієнічні вимоги до приміщення, в якому розташоване

робоче місце, власне до робочого місця, освітлення, рівнів вібрації і шуму, мікроклімату в приміщенні тощо.

Персональні комп'ютери, периферійні пристрої, інше устаткування (апарати управління, контрольно-вимірювальні прилади, світильники), електропроводи та кабелі за виконанням і ступенем захисту мають відповідати класу зони, мати апаратуру захисту від струму короткого замикання та інших аварійних режимів. Під час монтажу та експлуатації ліній електромережі необхідно повністю унеможливити виникнення електричного джерела загоряння внаслідок короткого замикання та перевантаження проводів, обмежувати застосування проводів з легкозаймистою ізоляцією і, за можливості, застосовувати негорючу ізоляцію. Лінія електромережі для живлення персональних комп'ютерів і периферійних пристроїв виконується як окрема групова трипровідна мережа шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Не допускається використовувати нульовий робочий провідник як нульовий захисний провідник. Нульовий захисний провідник прокладається від стійки групового розподільного щита, розподільного пункту до розеток електроживлення. Не допускається підключати на щиті до одного контактного затискача нульовий робочий та нульовий захисний провідники. Площа перерізу нульового робочого та нульового захисного провідника в груповій трипровідній мережі має бути не менше площі перерізу фазового провідника. Усі провідники мають відповідати номінальним параметрам мережі та навантаження, умовам навколишнього середовища, умовам розподілу провідників, температурному режиму та типам апаратури захисту. У приміщенні, де одночасно експлуатуються понад п'ять персональних комп'ютерів і периферійних пристроїв, на помітному та доступному місці встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення. Персональні комп'ютери і периферійні пристрої повинні підключатися до електромережі тільки за допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення. У штепсельних

з’єднаннях та електророзетках, крім контактів фазового та нульового робочого провідників, мають бути спеціальні контакти для підключення нульового захисного провідника. Їхня конструкція має бути такою, щоб приєднання нульового захисного провідника відбувалося раніше, ніж приєднання фазового та нульового робочого провідників. Порядок роз’єднання при відключенні має бути зворотним. Не допускається підключати персональні комп’ютери та периферійні пристрої до звичайної двопровідної електромережі, в тому числі з використанням перехідних пристроїв. Електромережі штепсельних з’єднань та електророзеток для живлення персональних комп’ютерів та периферійних пристроїв потрібно виконувати за магістральною схемою, по 3-6 з’єднань або електророзеток в одному колі. Штепсельні з’єднання та електророзетки для напруги 12В та 42В за своєю конструкцією мають відрізнятися від штепсельних з’єднань для напруги 127В та 220В. Штепсельні з’єднання та електророзетки, розраховані на напругу 12В та 42В, мають візуально (за кольором) відрізнятися від кольору штепсельних з’єднань, розрахованих на напругу 127В та 220В. Індивідуальні та групові штепсельні з’єднання та електророзетки необхідно монтувати на негорючих або важкогорючих пластинах. Електромережу штепсельних розеток для живлення персональних комп’ютерів і периферійних пристроїв при розташуванні їх уздовж стін приміщення прокладають по підлозі поруч зі стінами приміщення, як правило, в металевих трубах і гнучких металевих рукавах, а також у пластикових коробах і пластмасових рукавах з відводами відповідно до затвердженого плану розміщення обладнання та технічних характеристик обладнання. При розміщенні в приміщенні до п’яти персональних комп’ютерів і периферійних пристроїв допускається прокладання трипровідникового захищеного проводу або кабелю в оболонці з негорючого чи важкогорючого матеріалу по периметру приміщення без металевих труб та гнучких металевих рукавів. Не допускається в одній трубі прокладати ланцюги до 42В та вище 42В. При організації робочих місць операторів електромережу штепсельних розеток для живлення персональних комп’ютерів, периферійних пристроїв і у центрі приміщення прокладають у каналах або під знімною під-

логою в металевих трубах або гнучких металевих рукавах. При цьому не допускається застосовувати провід і кабель в ізоляції з вулканізованої гуми та інші матеріали, які містять сірку.

Режими праці та відпочинку

При організації праці, що пов'язана з використанням персональних комп'ютерів, для збереження здоров'я працюючих, запобігання професійним захворювання і підтримки працездатності слід передбачити внутрішньозмінні регламентовані перерви для відпочинку. Внутрішньозмінні режими праці і відпочинку мають передбачати додаткові нетривалі перерви в періоди, що передують появі об'єктивних і суб'єктивних ознак стомлення і зниження працездатності. За основну роботу з персональним комп'ютером слід вважати таку, що займає не менше 50% часу впродовж робочої зміни. Протягом дня мають передбачатися:

ТЕМА :перерви для відпочинку і вживання їжі (обідні перерви);

ТЕМА :перерви для відпочинку і особистих потреб (згідно з трудовими нормами);

ТЕМА :додаткові перерви, що вводяться для окремих професій з урахуванням особливостей трудової діяльності.

Тривалість обідньої перерви визначається чинним законодавством про працю і Правилами внутрішнього трудового розпорядку. Встановлюються такі внутрішньозмінні режими праці та відпочинку при роботі з ЕОМ при 8-годинній денній робочій зміні в залежності від характеру праці:

- для розробників програм слід призначати регламентовану перерву для відпочинку тривалістю 15 хвилин через кожну годину роботи за персональним комп'ютером;
- для операторів персональних комп'ютерів слід призначати регламентовані перерви для відпочинку тривалістю 15 хвилин через кожні дві години;

- для операторів комп’ютерного набора слід призначати регламентовані перерви для відпочинку тривалістю 10 хвилин після кожної години роботи за персональним комп’ютером.

У всіх випадках, коли виробничі обставини не дозволяють застосувати регламентовані перерви, тривалість безперервної роботи з персональним комп’ютером не повинна перевищувати 4 години. При 12-годинній робочій зміні регламентовані перерви повинні встановлюватися в перші 8 годин роботи аналогічно перервам при 8-годинній робочій зміні, а протягом останніх 4-х годин роботи, незалежно від характеру трудової діяльності, через кожну годину тривалістю 15 хвилин. З метою зменшення негативного впливу монотонності є доцільним застосовувати чергування операцій усвідомленого тексту і числових даних (зміна змісту роботи), чередування вводу даних та редагування текстів. Для зниження нервово-емоційного напруження, стомлення зорового аналізатору, поліпшення мозкового кровообігу, подолання несприятливих наслідків гіподинамії, запобігання втомі доцільні деякі перерви використовувати для виконання комплексу вправ. В окремих випадках – при хронічних скаргах працюючих на зорове стомлення, незважаючи на дотримання санітарно-гігієнічних вимог до режимів праці і відпочинку, а також застосування засобів локального захисту очей – допускаються індивідуальних підхід до обмеження часу робіт з персональним комп’ютером, зміни характеру праці, чергування з іншими видами діяльності, не пов’язаними з персональним комп’ютером.

5.2 Розрахунок системи штучного освітлення офісного приміщення

Вхідні дані. Довжина A = 10 м, ширина B = 7 м, висота H = 3,5 м. Висота робочої поверхні h_p = 1 м. Для освітлення приймаємо світодіодний офісний світильник L-office 25/3025/32. Коефіцієнт відбиття стелі ρ_n = 70%, стін ρ_c = 50%, робочої поверхні ρ_p = 30%.

Розв'язання. Відстань від стелі до робочої поверхні

$$H_0 = H - h_p = 2{,}5 \text{ м}$$

Відстань від стелі до світильника $h_c = 0$ м.

Висота підвішування світильника над освітлюваною поверхнею

$$h = H_0 - h_c = 2{,}5 \text{ м.}$$

Висота підвішування світильника над підлогою

$$H_n = h + h_p = 3{,}5 \text{ м.}$$

Для досягнення найбільшої рівномірності освітлення приймаємо

$$L / h = 1{,}5.$$

Відстань між центрами світильників

$$\text{L} = 1{,}5 \text{ h} = 3{,}75 \text{ м.}$$

Індекс приміщення

$$i = \frac{A \cdot B}{h \cdot (A + B)} = 1{,}18.$$

При i = 1,18; ρ_n = 70%, ρ_c = 50%, ρ_p = 30% для світильників типу L-office 25/3025/32 коефіцієнт використання світлового потоку η = 0,37.

Кількість світильників, необхідних для створення заданого освітлення,

$$N = \frac{E_{min} \cdot S \cdot K_з \cdot Z}{n \cdot \Phi_л \cdot \eta} = \frac{300 \cdot 245 \cdot 1 \cdot 1{,}1}{3025 \cdot 0{,}37} \approx 73 \, шт.,$$

де E_{min} – рівень мінімального освітлення за нормами, Лк;

S – площа приміщення, м^2;

$K_з$ – коефіцієнт запасу;

Z – коефіцієнт мінімального освітлення;

$n\Phi_л$ – сумарний світловий потік ламп, встановлених в одному світильнику (n - кількість ламп; $\Phi_л$ – світловий потік однієї лампи, Лм).

Загальна потужність освітлюваної установки

$$P_з = P_л * N = 32 * 73 = 2336 \text{ Вт} = 2{,}3 \text{ кВт}.$$

5.3 Розробка заходів щодо зменшення впливу шкідливих та небезпечних факторів

5.3.1 Мікроклімат

Гігієнічні вимоги до мікроклімату виробничих приміщень нормуються СанПіН 2.2.4.548-96. Оптимальні норми мікроклімату приміщення наведено в табл. 6.1.

Таблиця 6.1 - Оптимальні норми мікроклімату приміщень

Період Року	Температура повітря, ^{0}С не більше	Відносить. вологість повітря,%	Швидкість руху повітря, м/с
Холодний	21 ... 23	40 ... 60	0,1
Теплий	22 ... 24	40 ... 60	0,2

Підтримування параметрів мікроклімату в приміщенні забезпечується опаленням і кондиціонуванням. Кліматичні умови, підтримуються в межах: температура - 15 ... 30 ^{0}С, відносна вологість повітря 20 ... 80%, концентрація пилу в повітрі не більше 0.5 мг / м^3. Рівні іонізації повітря в приміщенні наведено в табл. 6.2.

Таблиця 6.2 - Рівні іонізації повітря

Рівні	Число іонів в 1 см куб. повітря	
	n +	n-
Мінімальні	400	600
Оптимальні	1500-3000	30000-50000
Максимальні	50000	50000

Так як джерел виділення шкідливих речовин у приміщенні немає, то місцевої вентиляції не потрібно. У приміщенні щодня повинне проводитися вологе прибирання.

6.3.2 Освітлення

Природне і штучне освітлення у приміщеннях регламентується нормами СНиП 23-05-95 в залежності від характеру зорової, системи та види освітлення, фону, контрасту об'єкта з фоном.

При роботі з ЕОМ, як правило, застосовується природне освітлення. Бажано щоб світлові прорізи розташовувалися зліва від оператора ЕОМ, допускається і правосторонній природне освітлення. У тих випадках, коли одного природного освітлення не вистачає, встановлюється суміщене освітлення. При цьому додаткове штучне освітлення застосовується не тільки в темний, але і в світлий час доби.

Для забезпечення нормованих значень освітленості в приміщенні слід проводити чистку скляних рам і світильників не рідше двох разів на рік і проводити своєчасну заміну перегорілих ламп.

Для штучного освітлення приміщення слід використовувати головним чином люмінесцентні лампи. Найбільш прийнятними є люмінесцентні лампи білого і тепло - білого світла.

Для виключення засвічення екранів дисплеїв прямими світловими потоками світильники загального освітлення мають в своєму розпорядженні збоку від робочого місця, паралельно лінії зору оператора і стіні з вікнами.

Слід обмежувати відбиту блесткость на робочих поверхнях за рахунок правильного вибору типів світильників та розташування робочого місця по відношенню до джерел штучного освітлення.

Рекомендоване освітлення для роботи з екраном дисплея складає 200 лк, а при роботі з екраном в поєднанні з роботою над документами - 400 лк. Рекомендовані яскравості в полі зору операторів повинні лежати в межах 1:5 - 1:10.

Освітлення має бути достатньо рівномірно розподілено на робочих поверхнях і в навколишньому просторі, не повинно бути різких тіней, прямий і відображеної бляклості; освітлення повинно бути рівномірно в часі; напрямок випромінюваного освітлювальними приладами світлового потоку має бути оптимальним.

6.3.3 Шум

Засоби і методи захисту від шуму визначені в ГОСТ 12.1.029-80. Для зниження шуму слід:

- послабити шум самих джерел, зокрема, передбачити застосування їх конструкції акустичних екранів, кожухів і т.д.;
- знизити ефект сумарної дії на робочі місця відбитих звукових хвиль за рахунок звукопоглинання енергії прямих звукових хвиль поверхнями огороджувальних конструкцій;
- застосовувати раціональне розташування обладнання;

– використовувати архітектурно-планувальні та технологічні рішення, спрямовані на ізоляцію джерел шуму.

6.3.4 Заходи захисту від ураження електричним струмом

Важливе значення для запобігання електротравматизму має правильна організація обслуговування діючих електроустановок, проведення ремонтних, монтажних і профілактичних робіт.

Залежно від категорії приміщення необхідно застосовувати певні захисні заходи, що забезпечують достатню електробезпеку при експлуатації, технічному обслуговуванні та ремонті. У приміщеннях з підвищеною небезпекоюелектроприлади, переносні світильники повинні бути виконані з подвійною ізоляцією або напруга живлення не повинна перевищувати 42 В.

Під час роботи оператору забороняється:

1) стосуватися одночасно екрану монітора і клавіатури; торкатися до задньої панелі системного блоку при включеному живленні;
2) перемикати роз'єми інтерфейсних кабелів периферійних пристроїв при включеному живленні;
3) захаращувати верхні панелі пристроїв сторонніми предметами;
4) проводити відключення харчування під час виконання активної задачі;
5) проводити часті перемикання харчування;
6) допускати попадання вологи на поверхню системного блоку, монітора, робочу поверхню клавіатури, дисководу, принтера та інших пристроїв;
7) проводити самостійно розкриття та ремонт обладнання.

Операторові забороняється приступати до роботи при виявленні будь-якої несправності обладнання до її усунення.

6.3.5 Захист від статичної електрики

Засоби захисту від статичної електрики наведені в ГОСТ 12.4.124-83.

Основні заходи, які застосовуються для захисту від статичної електрики виробничого походження, включають методи, що виключають або зменшують інтенсивність генерації зарядів, і методи усувають утворюються заряди. Інтенсивність генерації зарядів можна зменшити відповідним підбором пар тертя або змішуванням матеріалів таким чином, що в результаті тертя один із змішаних матеріалів наводить заряд одного знака, а інший - іншого. В даний час створено комбінований матеріал з нейлону і дакрону, що забезпечує захист від статичної електрики за цим принципом.

Утворені заряди статичної електрики усувають найчастіше шляхом заземлення електропровідних частин виробничого обладнання. Опір такого заземлення повинен бути не більше 100 Ом. При неможливості влаштування заземлення практикується підвищення відносної вологості повітря в приміщенні. Можна збільшити об'ємну провідність діелектрика, для чого в нього вносять графіт, ацетиленові сажу, алюмінієву пудру, а в рідкі діелектрики - спеціальні добавки. Для ряду машин і агрегатів знайшли застосування нейтралізатори статичної електрики (коронного розряду, радіоізотопні, аеродинамічні і комбіновані). У всіх типах цих пристроїв шляхом іонізації повітря поблизу елемента конструкції, накопичує заряд статичної електрики, утворюються іони, в те числі зі знаком, протилежним знаку заряду, що і викликає його нейтралізацію.

До засобів індивідуального захисту від статичної електрики належать електростатичні халати і спеціальне взуття, підошва якої виконана з шкіри або електропровідної гуми, а також антистатичні браслети.

6.3.6 Заходи щодо запобігання виникнення пожежі

Загальні вимоги до пожежної безпеки нормуються ГОСТ 12.1.004-91.

За категорією приміщення належить до пожежонебезпечної категорії В, оскільки містить речовини (масла) здатні горіти.

Основні засоби гасіння пожежі:

2 Вода:

Водою не можна гасити електроустановки під напругою.

3. Вуглекислий сніг

Утворюється з рідкої вуглекислоти, при її виході з балона. Температура снігу -80 ^{0}С. Застосовується для гасіння електроустановок під напругою, пожеж в закритих приміщеннях і на відкритих майданчиках при невеликих розмірах вогнища горіння.

- Піна

Піна застосовується в основному для гасіння горючих рідин.

2 Порошкові засоби

Створюються на основі неорганічних солей лужних металів, з додаванням соди, піску. Порошок є єдиними засобами гасіння лужних металів і з'єднань. Добре збивають полум'я, але не завжди повністю гасять, тому застосовуються спільно з іншими засобами пожежогасіння.

Приміщення має бути в обов'язковому порядку обладнано ручними засобами пожежогасіння.

Також дуже важливий при роботі в офісі активний відпочинок, який має полягати у виконанні комплексу гімнастичних вправ, спрямованих на зняття нервового напруження, м’язове розслаблення, відновлення функцій фізіологічних систем, що порушуються протягом трудового процесу, зняття втоми очей, поліпшення мозкового кровообігу і працездатності. За умови високого рівня напруженості робіт з персональним комп’ютером показане психологічне розвантаження у спеціально обладнаних приміщеннях (в кімнатах психологічного розвантаження) під час регламентованих перерв або в кінці робочого дня. Таким чином, мною де-

тально було розглянуто вимоги, які пред’являє чинне законодавство України до охорони праці офісних співробітників.

7 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Охорона довкілля — система заходів щодо раціонального використання природних ресурсів, збереження особливо цінних та унікальних природних комплексів і забезпечення екологічної безпеки. Це сукупність державних, адміністративних, правових, економічних, політичних і суспільних заходів, спрямованих на раціональне використання, відтворення і збереження природних ресурсів землі, обмеження негативного впливу людської діяльності на навколишнє середовище. Включає охорону атмосферного повітря, вод суші та вод Світового океану, земель, флори і фауни, геологічного середовища.

Мета охорони навколишнього середовища — протидія негативним змінам у навколишньому середовищі, які мали місце в минулому, відбуваються зараз або можуть бути.

Актуальність охорони навколишнього середовища, що перетворилася в глобальну проблему, пов'язана головним чином зі зростанням антропогенного впливу. Це зумовлено демографічним вибухом, урбанізацією, що прискорюється, і розвитком гірничих розробок і комунікацій, забрудненням навколишнього середовища відходами, надмірним навантаженням на орні землі, пасовища, ліси, водойми.

Заходами, спрямованими на охорону довкілля можуть бути:

- обмеження викидів в атмосферу та гідросферу з метою поліпшення загальної екологічної обстановки;
- створення заповідників, заказників і національних парків з метою збереження природних комплексів;
- обмеження лову риби, полювання з метою збереження певних видів;

– обмеження несанкціонованого викидання сміття. використання методів екологічної логістики для тотального очищення від несанкціонованого засмічення території регіону.

7.1 Забруднення навколишнього середовища промисловими підприємствами

7.1.1 Основні джерела забруднення атмосферного повітря на промислових підприємствах

Викиди шкідливих речовин в атмосферу можна поділити на чотири групи: тверді, рідкі, теплові та парогазоподібні. Причини утворення твердих речовин (виробничий пил) залежать від типу виробничого процесу та його характеру:

1) механічне оброблення різних речовин (буріння, розрівнювання, заповнення, подрібнення, розмелювання, полірування тощо);

2) транспортування сипких матеріалів (навантажувально-розвантажувальні процеси, просіювання, змішування тощо).

Одним із значних джерел викидів твердих речовин в атмосферу є металургійна промисловість, зокрема виробництва сирого чавуну (агломерація і доменні печі), сталі (кисневі конвертори та тандем-печі або двополюсні печі), феросплавів, ливарні дільниці та вагранки, коксові установки або генератори. Найбільшим джерелом виділення пилу на металургійних підприємствах є електродугові печі.

Рідкі забруднення (туман, краплі) утворюються: а) при конденсації випарів; б) при розпилюванні або розтіканні рідин; в) у результаті хімічних або фотохімічних реакцій. Теплові викиди трапляються під час спалювання, обпалю-

вання, сушіння, плавлення, конденсування, карбонізації, газифікації, дистиляції тощо.

Утворення паро- і газоподібних забруднень характерне для різних промислових підприємств, технологічні процеси яких відрізняються за характером, токсичністю, ступенем виділення шкідливих речовин в атмосферу. Переважна більшість технологічних процесів відзначається хімічними реакціями (окислення, відновлення, заміщення, розкладання), а також електрохімічними (електроліз) та фізичними (випарування, дистиляція, азеотропна дистиляція) процесами.

Найбільшу частину паро- і газоподібних викидів становлять продукти окислення, що утворюються переважно в процесах горіння, коли під час окислення вуглецю виділяється діоксид та оксид вуглецю, при окисленні сірки — діоксид сірки, а при високотемпературному окисленні азоту в печах — оксид і діоксид азоту.

Електрохімічні процеси є джерелом суттєвих забруднень як у металургії, так і в хімічній промисловості. Значними джерелами забруднення атмосфери в хімічній промисловості є також фізичні процеси, зокрема випарування та дистиляція (наприклад, викид вуглеводнів, хлорпохідних вуглеводнів та інших розчинників, що випаровуються в процесі виробництва, та використання цих продуктів). Дистиляція різних хімічних речовин, включно з смолами, а також деякі нафтоочисні та нафтохімічні процеси — ще одне джерело значних викидів шкідливих речовин в атмосферу.

7.1.2 Основні джерела та речовини, що забруднюють стічні води на промислових підприємствах

На території промислових підприємств утворюються стічні води трьох видів: побутові, поверхневі (зливні) та виробничі.

Побутові стічні води підприємств утворюються при експлуатації на їхній території душових кімнат, санвузлів, пральних приміщень, їдалень. Підприємства не відповідають за якість даних стічних вод і скеровують їх у міські (районні) станції очищення.

Поверхневі стічні води утворюються в результаті змивання дощовою (зливною), талою та поливальною водою домішок, що накопичуються на території, дахах і стінах виробничих будівель. Основними домішками цих вод є тверді частинки (пісок, камінь, стружка, ошурки, пил, сажа, рештки рослин, дерев тощо); нафтопродукти (масла, бензин, гас), що використовуються у двигунах транспортних засобів, а також органічні та мінеральні добрива, що застосовуються у заводських квітниках, скверах. Кожне підприємство несе відповідальність за забруднення водоймищ, тому важливим є визначення об'єму стічних вод конкретного типу.

Виробничі стічні води утворюються у результаті використання води у технологічних процесах (для охолодження технологічного обладнання, утворення технологічної пари в котельних установках, приготування і конденсації клеєвих розчинів).

7.1.3 Енергетичне забруднення довкілля

Промислові підприємства є потужними джерелами енергетичного забруднення довкілля. До енергетичних забруднень довкілля відносять шум, вібрацію, електромагнітні та іонізуючі випромінювання. Негативного впливу зазнають атмосферне повітря, гідросфера, літосфера, флора, фауна, а через них і людина.

Шум спричиняє шкідливу фізіологічну дію на людський організм, зумовлює професійні захворювання. Шкідлива фізіологічна дія шуму виявляється через ушкодження слухового апарату, травми центральної нервової системи, сповільнену психологічну реакцію, порушення функцій органів травлення. Шум призводить до порушення ритму серцебиття, підвищення кров'яного тиску,

погіршення функціонування органів дихання, збільшення об'єму внутрішніх органів, виникнення злоякісних пухлин, послаблення пам'яті тощо.

Шум шкідливо діє не лише на організм людини, але й пригнічує ріст та розвиток представників флори і фауни. Під впливом шуму сповільнюється ріст рослин, зменшується у 1,5—2 раза їх родючість, спостерігається надмірне виділення вологи через листя, руйнування рослинних клітин. Шум змушує лісових тварин залишати шумні ділянки лісу, навіть якщо вони багаті харчами, і мігрувати у віддаленіші райони. Внаслідок тривалого шумового забруднення знижується популяція диких тварин, змінюється ареал їх поширення. Негативно впливає шум і на мешканців водоймищ, що призводить до зміни балансу їхніх популяцій.

Серед промислових підприємств найбільш шумними є деревообробні, металургійні, машинобудівні, текстильні, автотранспортні, а також підприємства з випробовування турбореактивних двигунів, з виробництва металовиробів та ін.

Основними джерелами шумового забруднення довкілля на деревообробних і лісозаготівельних підприємствах є: деревообробне обладнання, вентиляторні та компресорні установки, автотранспортні засоби, трактори і бульдозери та ін. У всіх випадках вібрація поширюється ґрунтом і досягає фундаментів житлових та громадських будівель, часто призводить до звукових коливань. Передача вібрації через фундаменти та ґрунт може спричинити нерівномірне осідання, руйнування інженерних споруд і житлових будівель.

Унаслідок широкого застосування джерел електромагнітної енергії в різних галузях промисловості різко зріс загальний електромагнітний фон Землі. Основним джерелом електромагнітних полів (ЕМП) антропогенного походження у великих містах з високорозвиненою промисловістю та потужними радіотехнічними об'єктами є: радіотехнічні об'єкти (РТО), телевізійні та радіолокаційні станції (РЛС), термічні цехи та дільниці машинобудівних підприємств, сушильні, личкувальні та фанерні цехи деревообробних підприємств та ін. Дія на довкілля

ЕМП промислової частоти найчастіше пов'язана з високовольтними лініями (ВЛ) електропередач. Джерелами постійних магнітних полів є промислові підприємства. Зони з підвищеними рівнями ЕМП, джерелами яких можуть бути РТО і РЛС, сягають 100—150 м.

Дія БМП на довкілля пов'язана з накопиченням заряду на предметах, що не мають зв'язку з землею. В цьому випадку можливий перехід електричного потенціалу накопичених зарядів на заземлені предмети (елементи систем опалення, водопроводу та каналізації). Цей розряд може спричинити переляк у людини, мимовільні рухи і, як наслідок, травми.

При тривалій постійній дії ЕМП радіочастотного діапазону на організм людини спостерігається порушення серцево-судинної діяльності та нервової систем. Суб'єктивно це виявляється у постійних головних болях, підвищеній втомі, слабкості, порушенні сну, підвищеній дратівливості, погіршенні пам'яті тощо.

Дія іонізуючого випромінювання на людину може відбуватися в результаті зовнішнього та внутрішнього опромінювань. Зовнішнє опромінювання спричиняють джерела рентгенівського, у-випромінювання та потоки протонів і нейтронів, що знаходяться поза організмом людини. Внутрішнє опромінювання викликають а- і р-частинки, які потрапляють в організм людини з радіоактивними речовинами через органи дихання та травний тракт.

Розробка засобів щодо зменшення забруднення

7.2.1 Основні принципи та способи вилучення пилу з атмосферного повітря

На промислових підприємствах практично неможливо уникнути пилоутворення. Пил — полідисперсна система з розміром частинок від 5 до 200 мкм і більше. Крім розміру (фракційності) частинок, важливо знати також інші характеристики пилу: походження (матеріал) пилинок, їхню поверхню, електрозаряд,

вологість і хімічні властивості. Залежно від цього вибирають методи та засоби захисту атмосфери від забруднення пилом.

При пиловловлюванні необхідно також знати фізико-хімічні характеристики пилу: дисперсний (фракційний) склад, густину, адгезійні властивості, змочуваність, електричний заряд частинок, питомий опір шарів частинок та ін. Для правильного вибору пиловловлювачів потрібні насамперед відомості про дисперсний склад пилу туману.

Відомо два основні способи вилучення пилу з повітряного (газового) потоку: сухий і мокрий. Для реалізації цих способів застосовують різні пилоочисні установки.

Потрібен ретельний аналіз основних принципів, явищ і рушійних сил, які впливають на ефективність роботи пиловловлювачів. В інженерній практиці відомі три основні принципи вилучення пилу з повітряного (газового) потоку: механічний, електричний та акустичний.

7.2.2 Основні методи очищення атмосферного повітря від шкідливих парів і газів

Очищення та знешкодження технологічних і вентиляційних викидів промислових підприємств від газо- і пароподібних домішок характеризується тим, що, по-перше, гази, які викидаються в атмосферу, надто різні за хімічним складом; по-друге, вони мають іноді достатньо високу температуру і містять значну кількість пилу, що суттєво ускладнює процес газоочищення, і потребують попередньої підготовки відвідних газів; по-третє, концентрація газоподібних і пароподібних домішок часто у вентиляційних і рідше в технологічних викидах є змінна та низька.

Для реалізації завдань захисту атмосфери від шкідливих викидів зараз застосовують шість основних методів: абсорбція; адсорбція; хемосорбція; термічна нейтралізація; каталітичне знешкодження; хімічне знешкодження.

7.2.3 Основні способи очищення стічних вод

Попередження забруднення виробничих стічних вод на промислових підприємствах може бути забезпечене організаційними та технічними заходами.

Організаційні заходи зводяться до попередження спуску стічних вод у водоймища без очищення. Технічні заходи передбачають очищення стічних вод різними способами, їхнє повторне використання для технічних потреб і поливання, створення оборотних і замкнених систем водокористування, вдосконалення технологічних процесів на промислових підприємствах з метою зменшення кількості забруднень у стічних водах, перехід на безвідходні та маловідходні технології, скорочення забруднення територій паливно-мастильними та лакофарбовими матеріалами, мінеральними та органічними добривами, тирсою та іншими виробничими відходами, які зі зливними стоками можуть потрапляти у водоймища.

Очищення виробничих стічних вод на промислових підприємствах може здійснюватися за такими напрямками:

— очищення стічних вод на заводських очисних спорудах;

— очищення стічних вод після забруднення на заводських, а потім на міських очисних спорудах з подальшим спуском у водоймища;

— безперервне очищення виробничих стічних вод і розчинів на локальних очисних спорудах протягом визначеного часу, після чого вони потрапляють в обіг, і лише після з'ясування неможливості регенерації усереднюються і передаються на заводські очисні споруди та утилізуються.

Основні способи очищення виробничих стічних вод поділяються на: механічні, фізичні, фізико-механічні, хімічні, фізико-хімічні, біологічні та комплексні.

7.2.4 Захист ґрунтів і земельних ресурсів від шкідливих викидів

Ґрунти й земельні ресурси, як і водні джерела та атмосфера, потребують захисту від впливу шкідливих хімічних і фізичних факторів.

З метою зниження негативного впливу пестицидів, які часто застосовуються на підприємствах лісового господарства, рекомендують такі заходи:

— підвищення активності пестицидів (щоб знизити їхню діючу концентрацію до рівнів, нешкідливих для людей і тварин);

— створення нових менш токсичних хімічних препаратів третього й четвертого поколінь, речовин вузько спрямованої дії, а також речовин з коротким терміном життєздатності (піретрини та їхні аналоги);

— поєднання хімічних засобів прискорення росту лісових саджанців з агротехнічними, селекційними та організаційно-господарськими;

— збризкування замість розпилювання (різко скорочується радіус поширення і забруднення), використання гранул замість борошна;

— заборона викидів у ґрунти і водоймища неочищених виробничих стічних вод;

З озеленення територій промислових підприємств та населених житлових масивів для поглинання ними шкідливих промислових викидів і виділення цілющих фітонцидів.

7.2.5 Основні засоби захисту довкілля від шумового забруднення

Засоби захисту довкілля від шуму, які найчастіше застосовують на машинобудівних, металургійних, деревообробних, текстильних та інших підприємствах, можна поділити на засоби колективного та індивідуального захисту. Домінуючими вважають засоби колективного захисту від шуму на шляху його поширення (рис. 7.1). Методи та засоби захисту від вібрації представлено на рисунку 7.2.

Рисунок 7.1. Засоби колективного захисту від шуму на шляху його поширення

Рисунок 7.2. Класифікація методів і засобів захисту від вібрації

Зниження шуму в джерелі його виникнення полягає у зміні конструкції інструментів та інших обертових мас обладнання, їхньому балансуванні тощо. Він широко застосовується для найбільш шумного обладнання, що характеризується

великою частотою обертання. До такого обладнання належать деревообробні верстати, вентиляторні установки та ін.

Метод звукопоглинання найчастіше застосовують у виробничих приміщеннях. Звукопоглинання це зменшення енергії звукових хвиль, що відбиваються від зустрічних перепон через перетворення звукової енергії в теплову. Звукопоглинання застосовують тоді, коли неможливо досягнути зниження шуму в джерелі його виникнення.

Високий рівень шуму знижується за допомогою глушників, встановлених у каналах, трубопроводах, повітропроводах. Залежно від принципу дії глушники поділяють на абсорбційні, реактивні (рефлексні) та комбіновані. Шум в абсорбційних глушниках знижується поглинанням звукової енергії у звукопоглинальних матеріалах глушників, а в реактивних глушниках — у результаті відбиття звуку зворотно до джерела. Комбіновані глушники володіють властивістю як поглинати, так і відбивати звук.

Використання віброгасіння пов'язане зі збільшенням реактивної частини імпедансу коливної системи. Віброгасіння реалізується при збільшенні ефективної жорсткості та маси корпуса машин або станин верстатів при їх об'єднанні в єдину замкнену систему з фундаментом за допомогою анкерних болтів або цементної основи. З цією ж метою малогабаритне інженерне обладнання житлових будинків (вентилятори, насоси) встановлюють на опорні плити та віброгасні основи.

На етапі експлуатації промислових комплексів обладнання в основному встановлюють без фундаменту на віброізолювальних опорах. Такий метод дозволяє забезпечити будь-який ступінь віброізоляції обладнання. Встановлення на віброізолювальні опори технологічного й інженерного обладнання здешевлює його монтаж, виключає псування обладнання і знижує рівень шуму, що супроводжує інтенсивні вібрації.

7. ЗЗаходи щодо запобігання забруднення навколишнього середовища

Персональні комп'ютери, ноутбуки та інша інформаційна техніка, як відомо широко використовується в галузі наукових досліджень, промисловості, а також у повсякденному домашньому користуванні. Але будь-яка техніка стрімко застаріває, їй на зміну приходять нові, більш потужні, більш сучасні ПК та оргтехніка. Поступово виникає проблема, що робити зі старою технікою, морально застарілою або з тих чи інших причин, що вийшла з ладу, яка захаращує підсобні приміщення та склади.

Утилізація оргтехніки та комп'ютерів це процес, який проводиться в кілька етапів. Найперше - списання обладнання безпосередньо з підприємства. Етап другий це розбір техніки і сортування отриманих матеріалів. Якщо деталі здатні служити вихідною сировиною, наприклад, кінескоп, деталі, в складі яких є дорогоцінні метали, то їх відправляють на очищення. Як нам відомо до складу комп'ютера входить безліч металів таких як золото, срібло, алюміній, мідь та інших. Ще етапом по утилізації персональних комп'ютерів можна досягнути завдяки вторинній переробці. Сутність даного процесу полягає в тому, що можна витягти з цієї сировини частку корисних та рідких матеріалів таких як іридію, міді та інших. Цей процес відтворити набагато легше аніж наприклад видобути тонну міді, яка міститься в тисячотонних гірських породах.

Одне з нововведень для утилізації друкованих плат - придумали можливість спеціального розчину який розчинюють у гарячій воді. Дія якого зумовлює відшарування електронних компонентів. Таким чином 90% компонентів нових друкованих плат можна використовувати знову, тоді як у випадку звичайним методам – тільки 2%. Практично жодне підприємство не зможе самостійно утилізувати комп'ютери та оргтехніку, так як цей процес вимагає сучасного обладнання та специфічних знань. Тому довірити таку роботу можна тільки професіоналам, які мають великий досвід у даній сфері.

Проблема утилізації використаних комп’ютерів, периферійного обладнання, стає гострішою з кожним роком. Обсяги виробництва продуктів інформаційно телекомунікаційних технологій та частота їх заміни на нові моделі примушують компанії замислюватись над проблемою біодеградації. Успіхи в цій галузі допоможуть, серед іншого, компаніям-виробникам зменшити податки, котрі вони сплачують зараз за утилізацію застарілих моделей. Останнє тим більше важливо, оскільки робить екологізацію економічно вигідною, тож спрямовує у цю сферу дедалі більше зусиль дослідників та довгострокових капіталовкладень.

Таким чином, подальше поширення інформаційних технологій не збільшить, а навпаки – зменшить техногенне навантаження на довкілля. В кінцевому результаті виконаної роботи можна стверджувати, що вдосконалення сучасних інформаційних технологій, слід направляти не тільки для того щоб створювати людині максимально комфортні умови життя теперішнього часу. Але і для досягнення безвідходного процесу утилізації відпрацьованої техніки, не завдаючи шкоду навколишньому середовищу.

Висновки

В спеціальному розділі з охорони навколишнього середовища було розглянуто питання щодо забруднення навколишнього середовища інформаційною технікою зокрема і персональними комп’ютерами, та розроблено методи та заходи стосовно раціональної утилізації відпрацьованого обладнання не завдаючи шкоду природі.

ВИСНОВКИ

Кваліфікаційна (магістерська) робота присвячена розробці ймовірністної моделі управління змістом програмного забезпечення для обліку енергії та розробці програми для її реалізації.

Було проведено аналіз існуючих моделей управління змістом та обгрунтовано актуальність розробки додаткового інструменту для менеджементу змісту проекту для обліку енергії. У результаті вирішення поставлених завдань, проведено аналіз змісту проєкту з обліку енергії; виконано аналіз стратегічних методів планування та їх вплив на розвиток сценаріїв розивитку змісту проекту з обліку енергії; розроблено програмне забезпечення для управління змістом проєкту на основі ймовірністної моделі, що дозволить підвищити достовірність прогнозування розвитку змісту проекту з обліку енергії.

Була виконана розробка ескізного, технічного та робочого проектів програмного забезпечення та розроблено програмне забезпечення для побудови ймовірністного графу подій проекту, що дозволить підвищити достовірність прогнозування розвитку змісту проеку з обліку енергії з використанням досвіду настання подій в минулому цього проекту.

За результатами побудовано ймовірнісну графову модель у вигляді ймовірнісного графу змісту проекту з обліку енергії на базі минулих подій проекту що накопичені протягом останіх трьох років у системі управління проектами Mantis, звідки вони були експортомані у файл формату csv, після чого файл було імпортовано в створену систему та доповнено новими подіями з їх ймовірностями.

Запропонований підхід можна використовувати у якості ефективного інструменту підтримки прийняття рішень при управлінні змістом проектів. Також даний інструмент може використовуватись особами, що приймають рішення, на підприємствах, які мають потребу в плануванні чи прогнозуванні розвитку подій тих чи інших проектів та працюють з програмами управління проектами, що мають змогу зберігати статистику в файлах формата csv.

Октім того, було виконано розділи з аналізу економічної ефективності розробки, оформлення праці та навколишнього середовища.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Коваленко И. И. Методы экспертного оценивания сценариев / И. И. Коваленко, А. В. Швед – Николаев : Изд-во. ЧДУ им. Петра Могилы, 2012. – 156 с.

2 Графодинамичесвое моделирование структур организационных систем / И. И. Коваленко, М. В. Донченко, А. В. Швед, И. А. Кобылинский. – Н. : Илион, 2012. – 59с.

3 Коваленко И. И. Представление знаний на основе теорий грубых множеств / И. И. Коваленко, Т. В. Пономаренко, А. В. Швед. – Николаев : Илион, 2013. – 52 с.

4 Гожий О. П. Информационная технология генерации альтернативных стохастических графов / О. П. Гожий, І. І. Коваленко, Т. В. Пономаренко – Миколаїв : – Наукові праці МДТУ ім. П. Могили, 2006 – С. 30-36

5 Малышев И. А. Разработка интеллектуальной системы поддержки принятия экономических решений на основе методов теорий нечетких множеств / И. А. Малышев. - [Б. м. : б. в.], 2006 – 156 с.

6 Пономаренко Т. В. Экспертное планирование от достигнутого перспектив развития судостроительного холдинга в следующем десятилетии / Т. В. Пономаренко, А. А. Маслов – Миколаїв :- Макаровські читання, 2014 – С. 70-71

7 Руководство к Своду знаний по управлению проектами (Руководство PMBOK) Третье издание, Project Management Institute, 2004. — 401 с.

8 Полковников А. Управление проектами - выбор, внедрение и использование ПО, PC WEEK/RE, 1996, №34-35.

9 Matt Light, Daniel B. Stang. Gartner RAS Core Research. Magic Quadrant for IT Project and Portfolio Management, 2007. 11 p.

10 Lewis Cardin. Forrester Research. Forrester Wave™: Project Portfolio Management Tools, Q4 2007. 14 p.

11 Модин А. А. Исследование и анализ потоков информации на промышленном предприятии. — М.: Энергия, 1970. — 304 с.

12 Крылова Г.Д. Основы стандартизации, сертификации, метрологии: Учебник для вузов. — М.: Аудит, ЮНИТИ, 1998. — 479 с.

13 Богданов В. Управление проектами в Microsoft Project 2007. Учебный курс [В соответствии с PMBOK 2004]. — СПб.: Питер, 2007. — 592 с.

14 Шлее М. Qt4. Профессиональное программирование на C++. — СПб.: БХВ-Петербург, 2007. — 880 с.

15 “Основи охорони праці” під редакцією д.т.н. професора М.П. Купчика, д.т.н. професора М.П. Гандзюка. — Київ, 2000. — 410 с.

16 Тубальцев А.М. Розрахунок систем штучного освітлення: Методичний посібник. — Миколаїв: УДМТУ, 2001. — 84 с.

ДОДАТОК А - ТЕХНІЧНЕ ЗАВДАННЯ

Назва – програмне забезпечення для управління змістом проектів з побудови сценаріїв у вигляді ймовірнісних графів.

Область застосування: може бути застосованим при управлінні змістом проектів та інших задачах в яких потрібна підтримка прийняття рішення.

3) Підстави для розробки

Документом, на підставі якого ведеться розробка програмного продукту, є наказ 129уч від 26 жовтня 2020 року

2. Призначення розробки

2.1. Функціональне призначення

Даний програмний продукт призначений для побудови сценаріїв і формування спрощеного сценарію подій шляхом формування множини альтернативних сценаріїв в інтерактивному режимі за допомогою сценарного підходу та виконання аналізу можливих комбінацій логічних функцій «та», «або» та виключне «або».

2.2 Експлуатаційне призначення

Розроблене ПЗ повинно спростити процес прийняття рішення особі що приймає рішення щодо розвитку змісту проектів шляхом автоматизованого аналізу можливих комбінацій вершин-подій та ребер-зв’язків побудованих сценаріїв.

3 Вимоги до програмного забезпечення

3.1 Вимоги до функціональних характеристик

Програмне забезпечення є інструментом *автоматичного формуван*ня множини альтернативних сценаріїв в інтерактивному режимі за допомогою сценарного підходу, виконання аналізу можливих комбінацій подій та прийняття рішення.

3.1.1 Вимоги до складу виконуваних функцій

Програма повинна надавати користувачеві можливість виконання перерахованих нижче функцій:

- створення нового сценарію – створює нове поле для складання графу, який відображає події і зв’язки між ними;
- зберігання сценарію – зберігає сценарій в файл;
- завантаження сценарію – завантаження збереженого сценарію з файлу;
- додавання події – додає на поле нову подію і при цьому користувач може ввести назву і вірогідність події;
- видалення події – видалить подію;
- редагування події – дозволить змінити ім’я або вірогідність виконання події;
- додавання зв’язку – додає до сценарію зв'язок, який зв’язує дві події на сусідніх рівнях між собою, а його вага встановлюється батьківською подією, яка являє собою набір з множини «та» (яка повертає значення кон’юнкції між подіями), «або» (яка повертає значення диз’юнкції між подіями), чи виключне «або» (що повертає істину, якщо виконується тільки одна з подій) і визначає значимість події;
- видалення зв’язку – видаляє зв'язок зі сценарію;
- отримання спрощеного графу – аналізує розроблений сценарій і відображає спрощений граф;
- переміщення події за допомогою маніпулятора миші або клавіатури;
- додавання рівня – додає на поле новий рівень, на якому можуть знаходитись події.
- групування подій – об’єднує події обрані користувачем на одному рівні в групу, якій встановити назву.
- переміщення групи подій – переміщує групу подій за допомогою маніпулятору миші.

- видалення групи подій – видаляє групу подій.
- додавання критеріїв – додає критерії для конкретної події.

8) Попарне зрівняння критеріїв – зрівнює критерії на однієї події між собою за оцінками поставленими експертом, та надає кожному критерію ступінь важливості.
9) Оцінка подій за критеріями – виконує попарне зрівняння подій за критеріями, за оцінками поставленими експертом.

3.1.2 Вимоги до організації вхідних та вихідних даних

Вхідною інформацією для програмного продукту будуть:

На етапі створення сценарію додаються події і зв’язки між подіями;

На етапі додавання події вказується ім’я події, вірогідність її виконання та виключне виконання.

На етапі додавання зв’язку між подіями вказуються події між якими діє цей зв'язок та вказується вага зв’язку: виконується (логічне «та»), не виконується (логічне «або») або виконується/не виконується (логічне «та/або»).

Вихідною інформацією після аналізу графу буде спрощеній граф.

3.1.3 Вимоги до тимчасових характеристик

Вимоги до тимчасових характеристик програми не пред'являються

3.2 Вимоги до надійності

3.2.1 Вимоги до забезпечення надійного (стійкого) функціонування програми

Надійне (стійке) функціонування програми має бути забезпечене виконанням сукупності організаційно-технічних і програмних заходів, перелік яких наведено нижче:

3 здійснити контроль за введенням вхідних даних користувачем:

3.1 при введені вірогідності виконання події, дозволити користувачу вводити тільки числа і якщо число менше 0 або більше 1 повідомити користувача про неправильність введених даних і заблокувати кнопку збереження.

3.2 якщо користувач ввів ім’я події, яке вже використовується, то відати йому відповідне повідомлення і не дозволити зберегти подію з таким ім’ям.

4 автозбереження сценаріїв під час роботи користувача;

3.2.2 Час відновлення після відмови

Час відновлення після відмови, викликаного збоєм електроживлення технічних засобів (іншими зовнішніми чинниками), не фатальним збоєм (не крахом) операційної системи, не повинно перевищувати 2-х хвилин за умови дотримання експлуатаційних технічних і програмних засобів.

Час відновлення після відмови, викликаного несправністю технічних засобів, фатальним збоєм (крахом) операційної системи, не повинен перевищувати часу, необхідного на усунення несправностей технічних засобів і переустановлення програмних засобів.

3.2.3 Відмови за некоректних дій оператора

Відмови програми можливі внаслідок некоректних дій користувача при взаємодії з ПЗ. Програма повинна виконувати перевірку введених даних перед виконанням дій. У разі, коли ведені дані відповідні вимогам, програма повинна вивести повідомлення про помилку, і запропонувати змінити вхідні дані.

Крім того, у випадках де можна скористатися вже введеними даними, програма повинна пропонувати їх у вигляді довідника.

3.3 Умови експлуатації

3.3.1 Кліматичні умови експлуатації

Кліматичні умови експлуатації, при яких повинні забезпечуватися задані характеристики, повинні задовольняти вимогам, що пред'являються до технічних засобів в частині умов їх експлуатації.

3.3.2 Вимоги до видів обслуговування

Програма не вимагає проведення будь-яких видів обслуговування.

3.3.3 Вимоги до чисельності та кваліфікації персоналу

Для роботи програми достатньо одного користувача – експерта системи, який повинен орієнтуватися в графічному інтерфейсі програми, та бути експертом в тій предметній області на основі якої створюється сценарій.

3.4 Вимоги до складу і параметрів технічних засобів

Даний програмний продукт вимагає технічні засоби з параметрами не нижче наступних:

Процесор: не менше IntelPentium2 MHz.

Оперативна пам'ять: не менше 1 Gb;

Відео пам'ять: не менше 128Mb;

Місце на жорсткому диску: не менше 500Mb;

Маніпулятори: миша, клавіатура.

3.5 Вимоги до інформаційної та програмної сумісності

3.5.1 Вимоги до інформаційних структур і методів розв'язання

Вимоги до інформаційних структур (файлів) на вході і виході, а також до методів рішення не пред'являються.

Програмний продукт повинен функціонувати в операційних системах Windows і Unix і мати інтуїтивно зрозумілий інтерфейс.

3.5.2 Вимоги до вихідних кодів і мов програмування

Вихідні коди програми повинні бути реалізовані на мові програмування Java.

3.5.3 Вимоги до програмних засобів, які використовуються програмою

Системні програмні засоби, що використовуються програмою, повинні бути представлені ліцензійної локалізованої версією операційної системи.

3.6 Вимоги до маркування та упаковки

Програма поставляється у вигляді програмного продукту- на дистрибутивному (зовнішньому оптичному) носії (компакт-диску).

3.6.1 Вимога до маркування

Програмний продукт повинен мати маркування з позначенням товарного знаку компанії-розробника, типу (найменування), номера версії, порядкового номера, дати виготовлення та номера.

Маркування повинно бути нанесене на програмне виріб у вигляді наклейки, виконаної поліграфічним способом з урахуванням вимог ГОСТ 9181-74.

3.6.2 Вимоги до маркування та упаковки

Упаковка програмного виробу повинно здійснюватися в пакувальну тару підприємства-виробника.

3.7 Вимоги до транспортування і зберігання

Допускається транспортування програмного продукту всіма видами транспорту (в тому числі в опалювальних герметичних відсіках літаків без обмеження відстаней). При перевезенні в залізничних вагонах вид відправки - дрібний мало-вантажний.

При транспортуванні і зберіганні програмного продукту має бути передбачений захист від попадання пилу і атмосферних опадів. Не допускається кантування програмного виробу. Температура навколишнього повітря при транспортуванні від мінус 5 до плюс 50.

4 Вимоги до програмної документації

Склад програмної документації повинен містити:

4 Опис програми;

5 Інструкція користувача;

6 Вихідні тексти програми;

7 Програма та методика випробувань;

8 Технічне завдання.

5 Техніко-економічні показники.

Розрахунок економічності розробки та введення даної системи наводиться в економічному підрозділі курсової роботи.

6 Стадії та етапи розробки

Стадії й етапи розробки представлені в таблиці А1

Таблиця А1 – Стадії й етапи розробки

№	Стадії розробки	Етапи робіт	Зміст робіт	
1	2	3	5	
1	Технічне завдання	1.1 Обґрунтування необхідності розробки програми	Постановка задачі, збір початкових матеріалів, вибір і обгрунтування критеріїв ефективності та якості розроблюваної програми, обґрунтування необхідності проведення науково-дослідних робіт	1.9.20-4-9.20
		1.2 Науково-дослідні роботи	Визначення структури вхідних та вихідних даних, попередній вибір методів рішення завдання, обґрунтування доцільності застосування раніше розроблених програм, визначення вимог до технічних засобів, обґрунтування принципової можливості розв'язання поставленої задачі	5.9.20-10.9.20
		1.3 Розробка та затвердження технічного завдання	Визначення вимог до програми, розробка техніко-економічного обґрунтування розробки програми, визначення стадій, етапів і термінів розробки програми та документації до неї, вибір мов програмування, визначення необхідності проведення науково-дослідних робіт на подальших стадіях	11.9.20
2	Ескізний проект	2.1 Розробка ескізного проекту	Попередня розробка структури вхідних і вихідних даних, уточнення методів рішення завдання, розробка загального опису алгоритму рішення задачі, розробка техніко-економічного обґрунтування	12.9.20-15.9.20
		2.2 Затвердження ескізного проекту	Розробка пояснювальної записки, узгодження і затвердження ескізного проекту	16.9.20 - 25.9.20

3	Техніч-ний проект	3.1 Розробка технічного проекту	Уточнення структури вхідних і вихідних даних, розробка алгоритму рішення завдання, визначення форми представлення вхідних і вихідних даних, визначення семантики і синтаксису мови даних, розробка структури програми, остаточне визначення конфігурації технічних засобів	26.9.20 - 10.10.20
		3.2 Затвердження технічного проекту	Розробка плану заходів з розробки та впровадження програми, розробка пояснювальної записки, узгодження та затвердження технічного проекту	11.10.20- 20.10.20
1	2	3	4	5
4	Робочий проект	4.1 Розробка програми	Програмування і налагодження програми	21.10.20- 28.10.20
		4.2 Розробка програмної документації	Розробка програмних документів відповідно до вимог	1.11.20 - 10.11.20
		4.3 Випробування програми	Розроблення, погодження та затвердження програми і методики випробувань, проведення попередніх державних, міжвідомчих, прийомо-здавальних та інших видів випробувань, коректування програми і програмної документації за результатами випробувань	15.11.20- 20.11.20
5	Впрова-дження	5.1 Підготовка і передача програми	Підготовка і передача програми і програмної документації, оформлення і затвердження акта про передачу програми, передача програми в фонд алгоритмів і програм	21.11.20

7 Порядок прийому і контролю

Для контролю та прийому повинен бути наданий опис програми, а також програма та методика випробувань.

Порядок контролю і приймання даної розробки здійснюється представником замовника в присутності представника розробника згідно з програмою та методикою випробувань.

Якщо програма не пройшла випробування, виконавець зобов'язаний виправити помилки та недоліки в строк, не більше ніж 1 місяць з дня випробування.

За результатами прийому складається акт, який підписується представником замовника і представником розробника і затверджується керівниками організації-замовника та організації-розробника.

У разі виявлення помилок при прийомі програмного виробу складається акт про виявлені помилки, який підписується представниками замовника і розробника і затверджується керівниками організації - замовника та організації - розробника. Розробник повинен протягом не більше 1-го місяця виправити зазначені зауваження, і сповіщений про повторне проведення перевірки, не пізніше ніж за 2 тижні до початку прийому програмного продукту.

ДОДАТОК Б - ТЕКСТ ПРОГРАМИ

Лістинг 1 – файл Event.java

```
package components;

import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.FontMetrics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.Serializable;
import java.util.HashSet;
import java.util.Iterator;
import java.util.List;
import java.util.Set;
import javax.swing.JList;
import javax.swing.border.LineBorder;

/**
 * Даний клас визначає подію або вершину графу. Подія має предка, тип зв’зку
 * (та, або, виключне або чи відсутній взязок ), ім’я, список нащадків та
 * пріоритет. Він успадковується від класу GraphComponent. Даний клас являється
 * найважливішим, тому що на основі об’єктів цього класу будується граф та
 * відповідно аналізується. При побудові графу стрворюються події, від них
 * нащадки і т.д. Якщо клас має предка, значить він має звязок з ним, тип якого
 * вказаний у предка. Отже отрімуємо дерево подій, яке можна проаналізувати та
 * отримати ряд подій, які приведуть до найкращого результату.
 *
 * @param priority приорітет події
 * @param linkType тип події (та, або, виключне або чи відсутній взязок )
 * @param parent предок
 * @param childs список нащадків
 * @param selectedLinkTypeActive визначає чи активований вибір типу подій
 * @param eventsGroup група в якій знаходиться ця подія
 * @author Vlad Ch
 */
public class Event extends GraphComponent implements Serializable,
Comparable<Event> {
```

```
    public static final String DEFAULT_NAME = "Нова подія";
    protected final int PARENT_MARGIN = 20;
    private int minWidth = 100;
    private int minHeight = 63;
    public static final int AND = 0;
    public static final int OR = 1;
    public static final int XOR = 2;
    public static final int NOT = 3;
    public static final int ARC_ANGLE = 20;
    public static final int SIMPLE_EVENT = 0;
    public static final int IMPORTANT_EVENT = 1;
    public static final int MOST_IMPORTANT_EVENT = 2;
    public static final int OPTIONS_LINE_TOP = 20;
    public static final int PADDING_RECT = 7;
    private final Color SIMPLE_EVENT_BACKGROUND_COLOR = Color.WHITE;
    private final Color SIMPLE_EVENT_BORDER_COLOR = new Color(100, 100,
100);
    private final Color IMPORTANT_EVENT_BACKGROUND_COLOR = new
Color(206, 248, 214);
    private final Color IMPORTANT_EVENT_BORDER_COLOR = new Color(37,
162, 162);
    private final Color MOST_IMPORTANT_EVENT_BACKGROUND_COLOR =
new Color(249, 237, 179);
    private final Color MOST_IMPORTANT_EVENT_BORDER_COLOR = new
Color(191, 34, 34);
    private final double PRIORITY_CONST = 1.0;
    private final int LINK_TYPE_CONST = 0;
    private double priority;
    private int linkType;
    private Event parent;
    private HashSet<Event> childs = new HashSet<>();
    private JList list = new JList(new String[]{"AND", "OR", "XOR", "NOT"});
    private int PADDING_LEFT_LINK_TYPE = 0;
    private int PADDING_TOP_LINK_TYPE;
    private boolean selectedLinkTypeActive = false;
    private EventsGroup eventsGroup = null;
    private int status = SIMPLE_EVENT;

    /**
     * Конструктор класу зі встановленням місця знаходження об’єка.
     *
     * @param x координата x
```

```
 * @param y координата y
 */
public Event(int x, int y) {
    super(x, y);
    this.priority = PRIORITY_CONST;
    this.linkType = LINK_TYPE_CONST;
    this.parent = null;
    this.setSize(minWidth, minHeight);
    this.setName(DEFAULT_NAME);
    setMinSize(minWidth, minHeight);
    setLinkType(NOT);
    initList();
}

/**
 * Підготовлює компонент для вибору типу події. Встановлює необхідні опції
 * для правильного відображення коомпоненту.
 */
public final void initList() {
    list.setBorder(new LineBorder(SIMPLE_EVENT_BORDER_COLOR));

    list.setVisible(false);
    list.setFixedCellHeight(OPTIONS_LINE_TOP);
    list.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseClicked(MouseEvent e) {
            selectLinkTypeDeactive();
        }
    });

    list.addKeyListener(new KeyAdapter() {
        @Override
        public void keyPressed(KeyEvent e) {
            if (e.getKeyCode() == KeyEvent.VK_ESCAPE) {
                list.setVisible(false);
                selectedLinkTypeActive = false;
            }
        }
    });
}

/**
 * Повертає компонент вводу типу події.
```

```
     *
     * @return компонент вводу типу події
     */
    public JList getList() {
        return list;
    }

    /**
     * Активує компонент для вибору типу події.
     */
    public void selectLinkTypeActive() {
        selectLinkTypeDeactive();
        list.setBackground(getBackgroundColor());
        list.setBorder(new LineBorder(getBorderColor()));
        list.setLocation(getX() + getWidth() / 2, getY() + getHeight() -
OPTIONS_LINE_TOP);
        list.setSize(getWidth() / 2, OPTIONS_LINE_TOP * 4);
        selectedLinkTypeActive = true;
        list.setVisible(true);
        list.setSelectedIndex(linkType);
        list.requestFocusInWindow();

    }

    /**
     * Дактивує компонент для вибору типу події.
     */
    public void selectLinkTypeDeactive() {
        if (isSelectLinkTypeActiveActive()) {
            linkType = list.getSelectedIndex();
            list.setVisible(false);
            selectedLinkTypeActive = false;
        }
    }

    /**
     * Перевіряє чи активний компонент для вибору типу події.
     *
     * @return tru якщо активний або false якщо не активний
     */
    public boolean isSelectLinkTypeActiveActive() {
        return selectedLinkTypeActive;
    }
```

```
@Override
public void setSelected(boolean selected) {
   super.setSelected(selected);
   if (!selected) {
      selectLinkTypeDeactive();
   }
   this.selected = selected;
}

/**
 * Встановлюється група в якій знаходиться подія.
 *
 * @param eventsGroup об’єкт типу EventsGroup
 */
public void setEventsGroup(EventsGroup eventsGroup) {
   this.eventsGroup = eventsGroup;
}

/**
 * Повертає групу в якій знаходиться подія.
 *
 * @return групу в якій знаходиться подія
 */
public EventsGroup getEventsGroup() {
   return eventsGroup;
}

/**
 * Повертає пріоритет події.
 *
 * @return пріоритет події
 */
public double getPriority() {
   return priority;
}

/**
 * Встановлює пріоритет події.
 *
 * @param priority пріоритет події
 */
public void setPriority(double priority) {
```

```
    this.priority = priority;
}

/**
 * Повертає тип зв’язку події.
 *
 * @return тип зв’язку події
 */
public int getLinkType() {
    return linkType;
}

/**
 * Перетворює тип зв’язку події із числового значення в строкове.
 *
 * @param linkType числове значення типу події
 * @return строкове значення типу події
 */
public String getLinkType(int linkType) {
    String link = "";
    switch (linkType) {
        case AND: {
            link = "AND";
            break;
        }
        case OR: {
            link = "OR";
            break;
        }
        case XOR: {
            link = "XOR";
            break;
        }
        case NOT: {
            link = "NOT";
            break;
        }
    }
    return link;
}

@Override
public void setLocation(int x, int y) {
```

```
    super.setLocation(x, y);
    if (isInputActive()) {
        inputActive();
    }
    if (isSelectLinkTypeActiveActive()) {
        selectLinkTypeActive();
    }
    if (eventsGroup != null) {
        eventsGroup.refresh();
    }
}

@Override
public final void setSize(int width, int height) {
    super.setSize(width, height);
    if (isInputActive()) {
        inputActive();
    }
    if (isSelectLinkTypeActiveActive()) {
        selectLinkTypeActive();
    }
    if (eventsGroup != null) {
        eventsGroup.refresh();
    }
}

@Override
public void move(int x, int y) {
    super.move(x, y);
    if (isInputActive()) {
        inputActive();
    }
    if (isSelectLinkTypeActiveActive()) {
        selectLinkTypeActive();
    }
}

/**
 * Встановлює тип зв’язку події.
 *
 * @param linkType тип зв’язку події
 */
public final void setLinkType(int linkType) {
```

```
        if (linkType > 3) {
            this.linkType = 0;
        } else {
            this.linkType = linkType;
        }
    }

    /**
     * Повертає предка.
     *
     * @return предка
     */
    public Event getParent() {
        return parent;
    }

    /**
     * Встановлює предка.
     *
     * @param parent предок
     */
    public void setParent(Event parent) {
        if (parent == null) {
            priority = 1;
        }
        this.parent = parent;
    }

    @Override
    public void draw(Graphics2D g2) {
        super.draw(g2);
        if (getWidth() < minWidth) {
            this.setSize(minWidth, getHeight());
        }

        if (getHeight() < minHeight) {
            this.setSize(getWidth(), minHeight);
        }

        g2.setColor(getBackgroundColor());
        g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
        g2.setStroke(new BasicStroke(getLineBold()));
```

```
        g2.fillRoundRect(this.getX(), this.getY(), this.getWidth(), this.getHeight(),
ARC_ANGLE, ARC_ANGLE);
        g2.setColor(getBorderColor());
        g2.drawRoundRect(this.getX(), this.getY(), this.getWidth(), this.getHeight(),
ARC_ANGLE, ARC_ANGLE);
        g2.drawLine(this.getX(), this.getY() + this.getHeight() - OPTIONS_LINE_TOP,
this.getX() + this.getWidth(), this.getY() + this.getHeight() - OPTIONS_LINE_TOP);
        g2.drawLine(this.getX() + this.getWidth() / 2, this.getY() + this.getHeight() -
OPTIONS_LINE_TOP, this.getX() + this.getWidth() / 2, this.getY() + this.getHeight());
        FontMetrics fm = g2.getFontMetrics();
        g2.setColor(Color.BLACK);
        List<String> text = getConvertText(this.getName(), getHeight() -
Event.PADDING_RECT - Event.OPTIONS_LINE_TOP, this.getWidth() -
PADDING_RECT * 2, fm);
        for (int i = 0; i < text.size(); i++) {
            g2.drawString(text.get(i), this.getX() + PADDING_RECT, this.getY() +
fm.getHeight() * (i + 1));
        }

        String pitorityToString = String.valueOf(priority);
        if (pitorityToString.length() > 5) {
            pitorityToString = pitorityToString.substring(0, 5);
        }
        g2.drawString(pitorityToString, (this.getWidth() / 2 -
fm.stringWidth(pitorityToString)) / 2 + this.getX(), this.getY() + this.getHeight() +
fm.getHeight() - OPTIONS_LINE_TOP - 2);
        PADDING_LEFT_LINK_TYPE = (this.getWidth() / 2 -
fm.stringWidth(getLinkType(linkType))) / 2 + this.getWidth() / 2 + this.getX();
        PADDING_TOP_LINK_TYPE = this.getY() + this.getHeight() + fm.getHeight() -
OPTIONS_LINE_TOP - 2;
        g2.drawString(getLinkType(linkType), PADDING_LEFT_LINK_TYPE,
PADDING_TOP_LINK_TYPE);

        g2.setColor(getLineColor());
        if (parent != null) {
            g2.drawLine(getX() + getWidth() / 2, getY(), getX() + getWidth() / 2,
parent.getY() + parent.getHeight() + PARENT_MARGIN);
            g2.drawLine(getX() + getWidth() / 2, parent.getY() + parent.getHeight() +
PARENT_MARGIN, parent.getX() + parent.getWidth() / 2, parent.getY() +
parent.getHeight() + PARENT_MARGIN);
            g2.drawLine(parent.getX() + parent.getWidth() / 2, parent.getY() +
parent.getHeight() + PARENT_MARGIN, parent.getX() + parent.getWidth() / 2,
parent.getY() + parent.getHeight());
```

```
        }
    }

    @Override
    public void mouseClicked(int x, int y) {
        if (x >= getX() + PADDING_RECT && y >= getY() + PADDING_RECT / 2
                && x <= getX() + getWidth() - PADDING_RECT * 2
                && y <= getY() + getHeight() - PADDING_RECT * 2 -
OPTIONS_LINE_TOP) {
            inputActive();
        } else {
            inputDeactive();
        }
        if (x > getX() + getWidth() / 2 && y > getY() + getHeight() -
OPTIONS_LINE_TOP && x < getX() + getWidth() && y < getY() + getHeight()) {
            selectLinkTypeActive();
        } else {
            selectLinkTypeDeactive();
        }
    }

    /**
     * Активує ввод імені події.
     */
    public void inputActive() {
        setBackgroundTextArea(getBackgroundColor());
        inputActive(getX() + 4, getY() + Event.PADDING_RECT / 2, getWidth() - 7,
getHeight() - Event.PADDING_RECT - Event.OPTIONS_LINE_TOP);
    }

    /**
     * Додає нового нащадка.
     *
     * @param event нащадок.
     */
    public void addChild(Event event) {
        childs.add(event);
        recoutPriority();
    }

    /**
     * Додає список нащадків.
     *
```

```
 * @param events список нащадків
 */
public void addAllChild(Set<Event> events) {
    childs.addAll(events);
    recoutPriority();
}

/**
 * Повертає список нащадків.
 *
 * @return список нащадків
 */
public Set<Event> getChilds() {
    return childs;
}

/**
 * Видаляє нащадка.
 *
 * @param event нащадок
 */
public void removeChild(Event event) {
    childs.remove(event);
    recoutPriority();
}

/**
 * Видаляє всіх нащадків.
 */
public void removeAllChilds() {
    for (Iterator<Event> it = childs.iterator(); it.hasNext();) {
        Event event = it.next();
        event.setParent(null);
    }
    childs.removeAll(childs);
}

private void recoutPriority() {
    for (Iterator<Event> it = childs.iterator(); it.hasNext();) {
        Event event = it.next();
        event.setPriority(((int) (1. / childs.size() * 1000)) / 1000.);
    }
}
```

```
private Color getBackgroundColor(){
   switch (status) {
      case SIMPLE_EVENT: {
         return SIMPLE_EVENT_BACKGROUND_COLOR;
      }
      case IMPORTANT_EVENT: {
         return IMPORTANT_EVENT_BACKGROUND_COLOR;
      }
      case MOST_IMPORTANT_EVENT: {
         return MOST_IMPORTANT_EVENT_BACKGROUND_COLOR;
      }
   }
   return Color.white;
}

private Color getBorderColor(){
   switch (status) {
      case SIMPLE_EVENT: {
         return SIMPLE_EVENT_BORDER_COLOR;
      }
      case IMPORTANT_EVENT: {
         return IMPORTANT_EVENT_BORDER_COLOR;
      }
      case MOST_IMPORTANT_EVENT: {
         return MOST_IMPORTANT_EVENT_BORDER_COLOR;
      }
   }
   return Color.black;
}

private Color getLineColor(){
   switch (status) {
      case SIMPLE_EVENT: {
         return Color.BLACK;
      }
      case IMPORTANT_EVENT: {
         return IMPORTANT_EVENT_BORDER_COLOR;
      }
      case MOST_IMPORTANT_EVENT: {
         return MOST_IMPORTANT_EVENT_BORDER_COLOR;
      }
   }
```

```
        return Color.black;
    }

    private float getLineBold(){
        switch (status) {
            case SIMPLE_EVENT: {
                return 1;
            }
            case IMPORTANT_EVENT: {
                return 1.6f;
            }
            case MOST_IMPORTANT_EVENT: {
                return 1.6f;
            }
        }
        return 1;
    }

    @Override
    public void changeLevel(Level level) {
        if (level.getTotalLevel() != 0 || (level.getTotalLevel() == 0 &&
level.getNumberEvents() == 0)) {
            getLevel().removeEvent(this);
            level.addEvent(this);
            setLevel(level);
        }
    }

    @Override
    public GraphComponent getClone() {
        Event event = new Event(getX(), getY());
        event.setName(getName());
        event.setParent(getParent());
        event.setPriority(getPriority());
        event.setSize(getWidth(), getHeight());
        event.setLinkType(getLinkType());
        event.setEventsGroup(eventsGroup);
        event.setLevel(getLevel());
        return event;
    }

    @Override
    public String toString() {
```

```
        return getName();
    }
    @Override
    public int compareTo(Event o) {
        if (o.getName().equals(this.getName())) {
            return 0;
        } else {
            return 1;
        }
    }

    public void setStatus(int status) {
        this.status = status;
    }
    public int getStatus() {
        return status;
    }
}
```

Лістинг 2 – файл Graph.java

```
package components;

import analize.Analize;
import java.awt.Graphics2D;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import javax.swing.JComponent;

/**
 * Цей клас представляє собою граф (сценарій подій), який складається з рівнів,
 * подій та груп подій. Також він реалізує інтерфейс Serializable, що дозволить
 * сценарій зберегти на накопичувачі у вигляді файлу.
 *
 * @param levels список рівнів
 * @param saving кількість збережень сценарію
 * @param projectName назва проекту
 * @author Vlad Ch
 */
public class Graph extends JComponent implements Serializable {
```

```
public static final int addEvent_Successfull = 0;
public static final int addEvent_Error_Exist_Event = 1;
public static final int addEvent_Error_FirstLevel = 2;
List<Level> levels = new ArrayList<>();
public int saving = 0;
public String projectName = "";
Analize analyzer = new Analize(this);

/**
 * Конструктор для класа Graph
 */
public Graph() {
    setLocation(0, 0);
    setSize((int) getMaximumSize().getWidth(), (int) getMaximumSize().getHeight());
}

/**
 * Встановлює кількість збережень сценарію.
 *
 * @param saving кількість збережень.
 */
public void setSaving(int saving) {
    this.saving = saving;
}

/**
 * Повертає кількість збережень сценарію.
 *
 * @return кількість збережень сценарію
 */
public int getSaving() {
    return saving;
}

/**
 * Встановлює назву проекту.
 *
 * @param projectName назва проекту
 */
public void setProjectName(String projectName) {
    this.projectName = projectName;
}
```

```
/**
 * Повертає назву проекту.
 *
 * @return назву проекту
 */
public String getProjectName() {
    return projectName;
}

public void draw(Graphics2D g2) {
    for (Iterator<Level> it = levels.iterator(); it.hasNext();) {
        Level level = it.next();
        level.draw(g2);
    }
}

/**
 * Додає подію в сценарій.
 *
 * @param event подія
 * @return true, якщо подія успішно додана, інакше false
 */
public boolean addEvent(Event event) {
    setEventDefaultName(event);
    Level level = getLevel(event.getY());
    if (level == null) {
        level = addLevel();
    }
    if (level.addEvent(event)) {
        level.alignHeight();
        add(event.getInputText());
        add(event.getList());
        return true;
    } else {
        return false;
    }
}

/**
 * Повертає рівень за значенням координати y.
 *
 * @param y координата y
```

```
 * @return рівень
 */
public Level getLevel(int y) {
    for (Iterator<Level> it = levels.iterator(); it.hasNext();) {
        Level level = it.next();
        if (y >= level.getY() && y <= level.getY() + level.getHeight()) {
            return level;
        }
    }
    return null;
}

/**
 * Видаляє рівень зі сценарію.
 *
 * @param level рівень
 */
public void removeLevel(Level level) {
    if (level != null) {
        level.removeAll();
        if (level.getLastLevel() != null) {
            level.getLastLevel().setNextLevel(level.getNextLevel());
        }
        if (level.getNextLevel() != null) {
            level.getNextLevel().setLastLevel(level.getLastLevel());
        }
        levels.remove(level);
        for (int i = 0; i < levels.size(); i++) {
            levels.get(i).setTotalLevel(i);
        }
    }
}

/**
 * Повертає кількість равнів.
 *
 * @return кількість равнів
 */
public int getNumberLevels() {
    return levels.size();
}

/**
```

```
     * Повертає рівень за номером у списку.
     *
     * @param i номер рівня у списку
     * @return рівень за номером у списку
     */
    public Level getLevelByIndex(int i) {
        if (levels.isEmpty()) {
            return null;
        } else {
            return levels.get(i);
        }
    }

    /**
     * Повертає подію за координатами.
     *
     * @param x координата x
     * @param y координата y
     * @return подію за координатами
     */
    public Event getEvent(int x, int y) {
        Level level = getLevel(y);
        if (level != null) {
            return level.getEvent(x, y);
        }
        return null;
    }

    public Event getRootEvent(){
        if (getLevelByIndex(0) == null || getLevelByIndex(0).getNumberEvents() == 0){
            return null;
        }
        return getLevelByIndex(0).getEvent(0);
    }

    /**
     * Повертає групу подій за координатами.
     *
     * @param x координата x
     * @param y координата y
     * @return групу подій за координатами x та y. Якщо координати x, y не
     * належать жодній групі, то повертається null.
     */
```

```
public EventsGroup getEventsGroup(int x, int y) {
    Level level = getLevel(y);
    if (level != null) {
        return level.getEventsGroup(x, y);
    }
    return null;
}

/**
 * Повертає максимальну ширину сценарію на основі елементів, які в ньому
 * знаходяться.
 *
 * @return максимальну ширину сценарію на основі елементів, які в ньому
 * знаходяться
 */
public int getMaxX() {
    int x = 0;
    for (Iterator<Level> it = levels.iterator(); it.hasNext();) {
        Level level = it.next();
        int maxX = level.getMaxX();
        if (x < maxX) {
            x = maxX;
        }
    }
    return x;
}

/**
 * Додати новий рівень у сценарій.
 *
 * @return доданий рівень
 */
public Level addLevel() {
    Level level;
    if (levels.isEmpty()) {
        level = new Level(null);
        levels.add(level);
    } else {
        level = new Level(levels.get(levels.size() - 1));
        levels.add(level);
    }
    return level;
}
```

```
/**
 * Підвищує пріоритет компонента сценарію
 *
 * @param jgc компонент сценарію
 */
public void moveUpJGraphComponent(GraphComponent jgc) {
   getLevel(jgc.getY()).moveUpJGraphComponent(jgc);
}

/**
 * Додає нову групу подій у сценарій. Події для утворення групи обираються з
 * прямокутника який описується параметрами: координатами x y, шириною та
 * висотою.
 *
 * @param x координата x
 * @param y координата y
 * @param width ширина
 * @param height висота
 * @return додану групу подій
 */
public EventsGroup addEventsGroup(int x, int y, int width, int height) {
   Level level1 = getLevel(y);
   Level level2 = getLevel(y + height);
   int n1 = getNumberLevels() - 1, n2 = getNumberLevels() - 1;
   if (level1 != null) {
      n1 = level1.getTotalLevel();
   }

   if (level2 != null) {
      n2 = level2.getTotalLevel();
   }

   if (n1 > n2) {
      int buf = n1;
      n1 = n2;
      n2 = buf;
   }
   if (level1 != null || level2 != null) {
      List<Event> events = null;
      int level = -1;
      for (int i = n1; i <= n2; i++) {
         List<Event> ev = getLevelByIndex(i).getEvents(x, y, height, width);
```

```
            if (!ev.isEmpty()) {
                if (level == -1) {
                    Event parent = ev.get(0).getParent();
                    for (Iterator<Event> it = ev.iterator(); it.hasNext();) {
                        Event event = it.next();
                        //if (event.getParentEvent() != parent || event.getEventsGroup() !=
null){
                        if (event.getEventsGroup() != null) {
                            return null;
                        }
                    }
                    events = ev;
                    level = i;
                } else {
                    return null;
                }
            }
        }

        if (events != null) {
            EventsGroup eg = new EventsGroup(events);
            setEventsGroupDefaultName(eg);
            getLevelByIndex(level).addEventsGroup(eg);
            getLevelByIndex(level).alignHeight();
            add(eg.getInputText());
            return eg;
        }
    }
    return null;
}

/**
 * Додає групу подій у сценарій.
 * @param eventsGroup група подій.
 */
public void addEventsGroup(EventsGroup eventsGroup) {
    Level level = getLevel(eventsGroup.getY());
    for (int i = 0; i < eventsGroup.getNumberEvents(); i++) {
        addEvent(eventsGroup.getEvent(i));
    }
    level.addEventsGroup(eventsGroup);
    level.alignHeight();
    add(eventsGroup.getInputText());
```

```
}

/**
 * Повертає компонент графу за координатами.
 * @param x координата x
 * @param y координата y
 * @return компонент графу за координатами.
 */
public GraphComponent getJGraphComponent(int x, int y) {
    Level level = getLevel(y);
    if (level != null) {
        return level.getJGraphComponent(x, y);
    }
    return null;
}

/**
 * Перевіряє чи існує подія з ім’ям name
 * @param name ім’я події
 * @return true, якщо така подія існує, інакше false
 */
private boolean isEventExist(String name) {
    for (Iterator<Level> it = levels.iterator(); it.hasNext();) {
        Level level = it.next();
        if (level.isEventExist(name)) {
            return true;
        }
    }
    return false;
}

/**
 * Перевіряє чи є група подій з ім’ям name
 *
 * @param name ім’я групи подій
 * @return true, якщо група подій з таким ім’ям існує, інакше false
 */
private boolean isEventsGroupExist(String name) {
    for (Iterator<Level> it = levels.iterator(); it.hasNext();) {
        Level level = it.next();
        if (level.isEventsGoupExist(name)) {
            return true;
        }
```

```
    }
    return false;
}

/**
 * Встановлює ім’я по замовчуванням для події.
 * @param event подія
 */
private void setEventDefaultName(Event event) {
    if (event.getName().contains(Event.DEFAULT_NAME)) {
        event.setName(Event.DEFAULT_NAME);
        int i = 0;
        while (isEventExist(event.getName())) {
            i++;
            event.setName(Event.DEFAULT_NAME + String.valueOf(i));
        }
    }
}

/**
 * Встановлює ім’я по замовчуванням для групи подій.
 * @param eventsGroup група подій
 */
private void setEventsGroupDefaultName(EventsGroup eventsGroup) {
    eventsGroup.setName(EventsGroup.DEFAULT_NAME);
    int i = 0;
    while (isEventsGroupExist(eventsGroup.getName())) {
        i++;
        eventsGroup.setName(EventsGroup.DEFAULT_NAME + String.valueOf(i));
    }
}

/**
 * Ініціалізує все рівні.
 */
public void init() {
    for (Level level : levels) {
        level.init();
    }
}

public Analize getAnalyzer() {
    return analyzer;
```

```
    }

    public void setAnalyzer(Analize analyzer) {
        this.analyzer = analyzer;
    }

    public void analize(){
        analyzer.analize();
    }

    public void clearEventsStatus(){
        for (Level level : levels) {
            level.clearEventsStatus();
        }
    }

}
```

Лістинг 3 – файл Criterion.java

```
package analize;

import static analize.ComparableEvent.getDoubleGradation;
import components.Event;
import java.io.Serializable;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;

/**
 * Даний клас відповідає за заповнення матриці попарних порівнянь альтернатив по
 * даному критерію та підрахунок відповідних локальних пріоритетів.
 *
 * @param priority локальний пріоритет критерію
 * @param name ім’я критерію
 * @param vector вектор локальних пріоритетів для альтернатив
 * @param comparationMatrix матриця попарних порівнянь для альтернатив
 *
 * @author Vlad Ch
 */
public class Criterion implements Comparable<Criterion>, Serializable {

    private double priority = 0;
    private String name = "";
    private Map<Event, Double> vector = new HashMap<>();
    private Map<Event, Map<Event, String>> comparationMatrix = new HashMap<>();

    /**
     * Конмтруктор класу.
     *
     * @param name ім’я критерію
     * @param events список альтернатив
     */
    public Criterion(String name, Set<Event> events) {
        this.name = name;
        init(events);

    }
```

```
    /**
     * Виконує ініціалізацію матриці попарніх порівнянь та вектор локальних
     * пріоритетів. Заповнює їх значеннями "1", "0" або тими які вже існують.
     *
     * @param events
     */
    public final void init(Set<Event> events) {
        Map<Event, Double> tmpVector = new HashMap<>();
        Map<Event, Map<Event, String>> tmpComparationMatrix = new HashMap<>();

        for (Event event : events) {
            if (vector.containsKey(event)) {
                tmpVector.put(event, vector.get(event));
            } else {
                tmpVector.put(event, 0.);
            }
            Map<Event, String> tmpMap = new HashMap<>();
            for (Event event1 : events) {
                if ((comparationMatrix.containsKey(event) &&
comparationMatrix.get(event).containsKey(event1))) {
                    tmpMap.put(event1, comparationMatrix.get(event).get(event1));
                } else if (event.equals(event1)) {
                    tmpMap.put(event1, "1");
                } else {
                    tmpMap.put(event1, "0");
                }
            }
            tmpComparationMatrix.put(event, tmpMap);
        }
        vector = tmpVector;
        comparationMatrix = tmpComparationMatrix;
    }

    /**
     * Повертає приорітет критерію.
     *
     * @return приорітет критерію
     */
    public double getPriority() {
        return priority;
    }

    /**
```

```
     * Встановлює приорітет критерію.
     *
     * @param priority приорітет критерію
     */
    public void setPriority(double priority) {
        this.priority = priority;
    }

    /**
     * Повертає назву критерію.
     *
     * @return назву критерію
     */
    public String getName() {
        return name;
    }

    /**
     * Встановлює назву критерію.
     *
     * @param name назва критерію
     */
    public void setName(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return name;
    }

    @Override
    public int compareTo(Criterion o) {
        if (o.getName().equals(name)) {
            return 0;
        } else {
            return 1;
        }
    }

    /**
     * Повертає матрицю попарних порівнянь.
     *
```

```
     * @return матрицю попарних порівнянь
     */
    public Map<Event, Map<Event, String>> getComparationMatrix() {
        return comparationMatrix;
    }

    /**
     * Повертає вектор локальних пріоритетів.
     *
     * @return вектор локальних пріоритетів
     */
    public Map<Event, Double> getVectorEvents() {
        return vector;
    }

    /**
     * Встановнює значення в матрицю попарних порівнянь.
     *
     * @param ev1 альтернатива в рядку
     * @param ev2 альтернатива в стовпці
     * @param val значення
     */
    public void setValueEvent(Event ev1, Event ev2, String val) {
        comparationMatrix.get(ev1).put(ev2, val);
    }

    /**
     * Повертає значення з матриці попарних порівнянь.
     *
     * @param ev1 альтернатива в рядку
     * @param ev2 альтернатива в стовпці
     * @return значення з матриці попарних порівнянь
     */
    public String getValueEvent(Event ev1, Event ev2) {
        return comparationMatrix.get(ev1).get(ev2);
    }

    /**
     * Розраховується вектор попарних порівнянь для альтернатив.
     */
    public void colculationAlternativePriority() {
        double[][] criterionMatrix = new double[vector.size()][vector.size()];
        int i = 0;
```

```
        for (Iterator<Event> it1 = vector.keySet().iterator(); it1.hasNext();) {
            Event ev1 = it1.next();
            int j = 0;
            for (Iterator<Event> it2 = vector.keySet().iterator(); it2.hasNext();) {
                Event ev2 = it2.next();
                criterionMatrix[i][j] =
getDoubleGradation(comparationMatrix.get(ev1).get(ev2));
                j++;
            }
            i++;
        }
        double[] v = ComparableEvent.getVectorV(criterionMatrix);
        i = 0;
        for (Iterator<Event> it = vector.keySet().iterator(); it.hasNext();) {
            Event ev = it.next();
            vector.put(ev, v[i]);
            //System.out.println(v[i] + " ");
            i++;
        }
    }

    /**
     * Повертає пріоритет події.
     * @param event подія
     * @return пріоритет події
     */
    public double getPriorityEvent(Event event) {
        return vector.get(event);
    }
}
```

Лістинг 4 – файл Analize.java

```
package analize;

import components.Event;
import components.Graph;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.LinkedList;
import java.util.List;
import java.util.Set;

/**
 * Цей клас відповідає за аналіз переданого в нього графу. Він зберігає матриці
 * попарних зрівнянь критеріїв та нащадків подій, та виконує аналіз графу на
 * основі пріоритетів подій та їх типів звязків для нащадків ("та", "або",
 * "виключне або" чи відсутнічть звязку) Виконує наступні функції:
 *
 * <ul>
 * <li>отримання об’єкта для порівння нащадків події;</li>
 * <li>виконання аналізу графу;</li>
 * <li>вибір нащадків події за пріоритетами нащадків та типу зв’язку;</li>
 * <li>отримання найважливішого нащадка</li>
 * <li>отримання важливих для аналізу нащадків</li>
 * <li>підрахування середнього пріоритету за весь шлях до події</li>
 * <li>встановлення статусу шляху до події</li>
 * <li>отримання найважливішої події зі списку</li>
 * </ul>
 *
 * @param comparableEvents список об’єктів порівнянь подій
 * @param graph граф, який підлягає аналізу
 *
 * @author Vlad Ch
 */
public class Analize implements Serializable {

    private List<ComparableEvent> comparableEvents = new ArrayList<>();
    private Graph graph;

    /**
     * Конструктор класу в який передається граф, який підлягає аналізу.
     *
```

```
 * @param graph
 */
public Analize(Graph graph) {
   this.graph = graph;
}

/**
 * Повертає об’єкт для порівняння події зі списку таких об’єктів. Якщо за
 * переданою подією event, такого об’єкта немає в списку, то створюється
 * новий об’єкт для заданої події. Він додається до списку та функція
 * повертає цей об’єкт.
 *
 * @param event подія, для якої потрібно отримати об’єкт для зрівняння
 * @return об’єкт для порівняння ComparableEvent для заданої події event
 */
public ComparableEvent getComarableEvent(Event event) {
   for (ComparableEvent comparableEvent : comparableEvents) {
      if (comparableEvent.getEvent().equals(event)) {
         return comparableEvent;
      }
   }
   ComparableEvent comparableEvent = new ComparableEvent(event);
   comparableEvents.add(comparableEvent);
   return comparableEvent;
}

/**
 * Виконує аналіз графу на основі пріоритетів та типу звязків подій
 * починаючи з корневої події. В функції будується черга з подій, які треба
 * проаналізувати. З початку в черзі тільки корнева подія. Далі в циклі,
 * доки черга не стане пустою, з неї обирається перша подія. Якщо подія не
 * має нащадків, то вона додається до списку важливих подій інакше до черги
 * додаються важливі нажадки події, які обираються фунцією chooseChilds(),
 * та подія видаляється з черги. Після виявлення важливих листів графу,
 * функція відмічає найважливійший та менш важливі шляхи (сценарії) для
 * досягнення цілі.
 * @return true, якщо аналіз успішно виконано, інакше false.
 */
public boolean analize() {
   Set<Event> importantEvents = new HashSet<>();
   LinkedList<Event> queue = new LinkedList<>();
   if (graph.getRootEvent() != null) {
      queue.add(graph.getRootEvent());
```

```
    }
    while (!queue.isEmpty()) {
        Event event = queue.getFirst();
        if (event.getChilds().isEmpty()) {
            importantEvents.add(event);
        } else {
            queue.addAll(chooseChilds(event));
        }
        queue.remove(event);
    }
    graph.clearEventsStatus();
    for (Event event : importantEvents) {
        setStatusPath(event, Event.IMPORTANT_EVENT);
    }
    Event bestEvent = getMostImportantEvent(importantEvents);
    if (bestEvent != null) {
        setStatusPath(bestEvent, Event.MOST_IMPORTANT_EVENT);
    } else {
        return false;
    }
    return true;
}

/**
 * Обирає важливих для аналізу нажадків подій на основі приоритетів нащадків
 * та типу звязку. Якщо тип звязку "та" - обираються всі нащадки, "або" -
 * важливі для аналізу нащадки, "виключне або" - найважливіший нащадок,
 * відсуній тип пустий список.
 *
 * @param event подія для якої обираються нащадки
 * @return список обраних нащадків
 */
public Set<Event> chooseChilds(Event event) {
    Set<Event> result = new HashSet<>();
    switch (event.getLinkType()) {
        case Event.AND: {
            result = event.getChilds();
            break;
        }
        case Event.OR: {
            result = getImportantChilds(event);
            break;
        }
```

```
        case Event.XOR: {
            Event ev = getMostImportantChild(event);
            if (ev != null)
            result.add(ev);
        }
    }
    return result;
}

/**
 * Повертає найважливішого нащадка події event.
 *
 * @param event подія, для якої потрібно знайти найважливішого нащадка
 * @return найважливішого нащадка події event
 */
public Event getMostImportantChild(Event event) {
    if (event == null){
        return null;
    }
    Event result = null;

    double priority = 0.;
    for (Event ev : event.getChilds()) {
        if (ev.getPriority() > priority) {
            priority = ev.getPriority();
            result = ev;
        }
    }
    return result;
}

/**
 * Повертає список важливих для аналізу нащадків. Обираються ті нащадки, які
 * мають пріортет не меньше ніж половина значення приорітету найважливішої
 * події.
 *
 * @param event подія для якої потрібно обрати важливих нащадків
 * @return список важливих для аналізу нащадків
 */
public Set<Event> getImportantChilds(Event event) {
    Set<Event> result = new HashSet<>();
    Event betterEvent = getMostImportantChild(event);
    if (betterEvent != null) {
```

```
        for (Event child : event.getChilds()) {
            if (Double.compare(betterEvent.getPriority() / 2., child.getPriority()) == -1) {
                result.add(child);
            }
        }
    }
    return result;
}

/**
 * Підраховує середній пріоритет від заданої події до цілі.
 *
 * @param event подія
 * @return середній пріоритет від заданої події до цілі
 */
public double averagePriority(Event event) {
    double result = 0.;
    int i = 1;
    while (event.getParent() != null) {
        result += event.getPriority();
        event = event.getParent();
        i++;
    }
    return result / i;
}

/**
 * Змінює статус подій, які зустрічаються на шляю від події event до цілі на
 * status. Таким чином можна буде візрізнити отримані після аналізу шляхи.
 *
 * Статус може бути:
 *
 * Event.SIMPLE_EVENT - проста подія;
 * Event.IMPORTANT_EVENT - важлива подія;
 * Event.MOST_IMPORTANT_EVENT - найважливыша подія;
 *
 * @param event подія для якої потрібно змінити статус
 * @param status статус
 */
public void setStatusPath(Event event, int status) {
    while (event.getParent() != null) {
        event.setStatus(status);
        graph.moveUpJGraphComponent(event);
```

```
            event = event.getParent();
        }
    }

    /**
     * Обирає подію з найбільшим пріоритетом зі списку.
     * @param events список подій
     * @return подію з найбільшим пріоритетом
     */
    public Event getMostImportantEvent(Set<Event> events) {
        Event result = null;
        double priority = 0.;
        for (Event event : events) {
            double averagePriority = averagePriority(event);
            if (averagePriority > priority) {
                priority = averagePriority;
                result = event;
            }
        }
        return result;
    }
}
```

ДОДАТОК В - ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ

1 Об’єкт випробувань

Об’єктом випробувань є програмне забезпечення для управління змістом. Програма призначена для побудови сценаріїв змісту у вигляді ймовірністного графу на базі файлів, що експортовано з системи управління проєктами.

2 Мета випробувань

Мета випробувань – перевірка працездатності програми та відповідності її характеристик та виконуваних функцій тим, що вказані у технічному завданні.

3 Вимоги до програми

Розглянута програма повинна забезпечувати виконання наступних функцій:

- створення нового сценарію – створює нове поле для складання графу, який відображає події і зв’язки між ними;
- зберігання сценарію – зберігає сценарій в файл;
- завантаження сценарію – завантаження збереженого сценарію з файлу;
- додавання події – додає на поле нову подію і при цьому користувач може ввести назву і вірогідність події;
- видалення події – видалить подію;
- редагування події – дозволить змінити ім’я або вірогідність виконання події;
- додавання зв’язку – додає до сценарію зв'язок, який зв’язує дві події на сусідніх рівнях між собою, також вказується його вага, яка являє собою набір з множини «та» (яка повертає значення кон’юнкції між подіями), «або» (яка повертає значення диз’юнкції між подіями), чи виключне «або» (що повертає істину, якщо виконується тільки одна з подій) і визначає значимість події;
- видалення зв’язку – видаляє зв'язок зі сценарію;

- редагування зв’язку – змінює вагу зв’язку або події які зв’язує цей зв'язок;
- отримання спрощеного графу – аналізує розроблений сценарій і відображає спрощений граф;
- друкування сценарію – друкує сценарій;
- перегляд поточного сценарію – дозволяє продивитись поточний сценарій в режимі перегляду;
- перегляд спрощеного сценарію – дозволяє продивитись спрощений сценарій в режимі перегляду;
- переміщення події за допомогою маніпулятора миші або клавіатури;
- переміщення зав’язку за допомогою маніпулятора миші або клавіатури, а також переміщення кінців зв’язку і прикріплення їх до подій;
- додавання рівня – додає на поле новий рівень, на якому можуть знаходитись події.
- групування подій – об’єднує події обрані користувачем на одному рівні в групу, якій встановити назву.
- додавання критеріїв – додає критерії для конкретного рівня.
- попарне зрівняння критеріїв – зрівнює критерії на одному рівні між собою за оцінками поставленими експертом, та надає кожному критерію ступінь важливості.
- оцінка подій за критеріями – виконує попарне зрівняння подій за критеріями, за оцінками поставленими експертом.

4 Вимоги до програмної документації

До складу програмної документації повинні входити:

9 Опис програми;

10 Вихідні тексти програми;

11 Програма та методика випробувань;

12 Технічне завдання.

5 Засоби та порядок випробування

Випробування повинно проводитись на комп’ютері з наступними вимогами:

1) Процесор: не менше IntelPentium2 MHz.
2) Оперативна пам'ять: не менше 1 Gb;
3) Відео пам'ять: не менше 128Mb;
4) Місце на жорсткому диску: не менше 500Mb;
5) Маніпулятори: миша, клавіатура;
6) ОС Windows або Linux.

Випробування включає в себе наступні етапи:

- Перевірка коректності запуску та ініціалізації програми;
- Перевірка правильної роботи редактору;
- Перевірка правильності відображення об’єктів на екрані;
- Перевірка правильності виконання аналізу;
- Перевірка коректності завершення роботи програми.

6 Методи випробувань

Для проведення випробування необхідно виконати наступні етапи:

1) Перевірка коректності запуску та ініціалізації програми.
   Після запуску програми повинен відкритися редактор графу та пусте поле для побудови графу.

2) Перевірка правильної роботи редактору.
   При побудові сценарію за допомогою редактору, його інструменти повинні коректно працювати:

«Подія» - повинен створювати нову подію у вказаному місці, з ім’ям(«нова подія..»), типом зв’язку(NOT) та пріоритетом(0) за замовченням.

«Групування подій» - повинен створити групу з виділених користувачем подій. Ім’я(«нова група») встановлюється за замовченням. Якщо події не виділені, то група не створюється.

«Зв’язок» - якшо це можливо, створює зв'язок між двома подіями, обраними користувачем.

«Рівень» - додає новий рівень на поле.

Графічні компоненти, такі як «Подія» та «Група подій» за певними діями користувача повинні коректно змінювати своє місце знаходження, рівень, розмір та ім'я

3) Перевірка правильності відображення об’єктів на екрані.

   Компоненти графу повинні коректно відображатись на екрані, тобто: кожен компонент повинен знаходитись на своєму рівні і не виходити за його рамки, правильно рисуватись без пробілів та викривлень. Якщо об’єкт знаходиться за рамками графічного поля, то повинен ввімкнутись скорінг цього поля, для того щоб мати до нього доступ.

4) Перевірка правильності виконання аналізу.

   Аналіз повинен забезпечити правильний розрахунок пріоритетів продій та знаходження найуспішнішого та успішних шляхів для досягнення цілі.

5) Перевірка коректності завершення роботи програми

Програма повинна завершитись коректно, без збоїв та запропонувати зберегти поточний граф, якщо він був змінений.

ДОДАТОК Г – ОПИС ПРОГРАМИ

«Програмне забезпечення для управління змістом проектів з побудови сценаріїв у вигляді ймовірнісних графів» - це програмне забезпечення призначенне для побудови сценаріїв та формування спрощеного сценарію подій шляхом формування множини альтернативних сценаріїв в інтерактивному режимі. Також даний продукт повинен спростити процес прийняття рішення ОПР, відносно розвитку змісту проектів шляхом автоматичного аналізу можливих комбінацій вершинподій і ребер (зв'язків) побудованих сценаріїв з врахуванням їх ймовірностей.

Інтерфейс програми дозволяє будувати сценарії за допомогою предоставленого інструметарію. У головному вікні програми є панель на якій відображаються поле для побудови графу, панель інструментів і головне меню. З панелі інструментів можна додавати об'єкти на поле, виконувати дії над ними (копіювати, вирізати вставити, оцінювання альтернатив, аналіз і т. д). При додаванні об'єкта на поле можна вказати його властивості. Наприклад для об'єкта «Подія» вказуючиється вага і назва, а для зв'язку - початкова подія, кінцеве подія. За допомогою миші об'єкти можна переміщати по полю і змінювати розмір.

В програмі реалізована можливість оцінки пріоритетів для подій за допомогою метода аналізу ієрархій Т. Сааті. Для обраної події можна додати критерії за якими можна попарно порівняти нащадків цієї події.

Після оцінки подій, можна виконати аналіз, який відобразить спрощений граф і тим самим дозволить ОПР прийняти рішення.

За допомогою головного меню можна створити новий граф, завантажити сценарій, зберегти сценарій на накопичувач, виконати оцінку, виконати аналіз та використовувати інструментарій для побудови графу, як і на панелі інструментів.

ДОДАТОК Д – ІНСТРУКЦІЯ КОРИСТУВАЧА

Інструкція користувача розроблена на основі головного меню програми.

1 Файл

1.1 Створити новий граф

Створює нове поле для побудови графу. За допомогою панелі інструментів можна додавати на поле об'єкти та виконувати над ними маніпуляції, а також хапустить процес імпорту подій.

1.2 Відкрити граф

Відкриває діалогове вікно за допомогою якого можна вибрати файл для завантаження. Після вибору файлу та натиску кнопки відкриття файлу проекту, граф відображаєся на новому полі та готовий для редагування.

1.3 Зберегти граф

Відкриває діалогове вікно для збереження проєкту. Після обрання каталогу та назви, прокт зберігається на накопичувачі у вигляді файлу.

Якщо даний проект вже був збережений раніше у файл, то він зберігається в автоматично в той же файл.

1.4 Зберегти як ...

Відкриває діалогове вікно для збереження проекту. Після обрання каталогу та назви, прокт зберігається на накопичувачі у вигляді файлу.

1.5 Вихід

Завершує роботу програми. Якщо граф відкритий в програмі не збережений, то програма запропонує зберегти його.

2 Правка

2.1 Вирізати

Вирізає виділений елемент на полі в буфер обміну.

2.2 Копіювати

Копіює виділений елемент в буфер обміну.

2.3 Вставити

Вставляє елемент на полі з буфера обміну.

2.4 Видалити

Видаляє виділений на полі елемент.

3 Інструменти

3.1 Подія

Вмикає режим додавання події. Після обрання цього пункту можна додати подію кліком лівої кнопки миші на полі. Подія з’явиться у місці кліку. Якщо подія додається не на рівень, то рівень додається автоматично.

3.2 Зв’язок

Вмикає режим додавання зв’язку між подіями. Для додавання зв’язку між двома подіями, потрібно натиснути ліву кнопку миші на першій події, та протягнути лінію зв’язку до другої події та відпустити кнопку миші. Якщо після відпускання кнопки миші за отриманими в цьому місці координатами, знаходиться подія сусіднього рівня, то зв’язок додається, інакше не додається.

3.3 Групування події

Вмикає режим групування подій. Щоб групувати події потрібно натиснути кнопкою миші на полі та за допомогою прямокутника, який з’явиться виділити

події, які потрібно групувати. Після обрання подій, та відпускання кнопки миша, буде створено групу подій.

3. 4 Додати рівень

Додає новий рівень на поле.

4 Аналіз

4.1 Оцінити альтернативи

Відкриває вікно для оцінки альтернатив (нащадків) для обраної події. Для заданої події можна додавати та видаляти критерії за допомогою відповідних кнопок. За допомогою кнопкри «порівняти критерії», відкривається матриця попарних порівнянь, для критеріїв, де можна ввести градації. Також при обранні критеріїв відкриваються відповідні матриці попарних порівнянь альтернатив за критеріями.

4.2 Виконати аналіз

Виконує аналіз поточного графа на основі ймовірностей подій та їх зв'язків. Після аналізу на полі буде представлений спрощений ймовірністний граф, який спростить менеджеру прийняття рішення.