



Міністерство освіти і науки України
Національний технічний університет України “Київський політехнічний
інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №2
З дисципліни «Технології розроблення програмного забезпечення»
Тема: **«ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ. СЦЕНАРІЙ
ВАРІАНТІВ ВИКОРИСТАННЯ. ДІАГРАМИ UML. ДІАГРАМИ КЛАСІВ.
КОНЦЕПТУАЛЬНА МОДЕЛЬ СИСТЕМИ»**
Flex Automatical tool

Виконав:
Студент групи ІА-22
Сидорін Д.О.

Перевірив:
Мягкий М. Ю.

Київ-2025

Зміст

Тема:.....	3
Мета:	3
Хід роботи	3
1. Схема прецедентів	4
2. Деталі сценаріїв використання	4
3. Схема класів	7
4. Структура бази даних.....	10
Висновки:	12

Тема:

Діаграма варіантів використання. Сценарії варіантів використання. Діаграми uml. Діаграми класів. Концептуальна модель системи

Мета:

Дослідити та застосувати UML-діаграми для моделювання варіантів використання і концептуальної структури даних системи, зосереджуючись на діаграмах класів, прецедентів та опису їхніх сценаріїв використання.

Завдання:

Розробити візуальний додаток для автоматизації, що включає функції для створення та управління кількома автоматичними завданнями (у вкладках). Користувач повинен мати можливість автоматично налаштовувати і виконувати дії, такі як завантаження нових файлів, оновлення статусів в месенджерах, виконання макросів. У додатку має бути можливість додавати категорії для завдань (терміновість, типи дій) та автоматично виконувати їх згідно з заданими правилами. Також повинна бути можливість візуалізації завдань у вигляді "карт пам'яті", де кожне завдання або макрос буде представлено у вигляді вузла з можливістю додавати вкладені файли, зображення, відео (з попереднім переглядом).

Хід роботи

1. Схема прецедентів

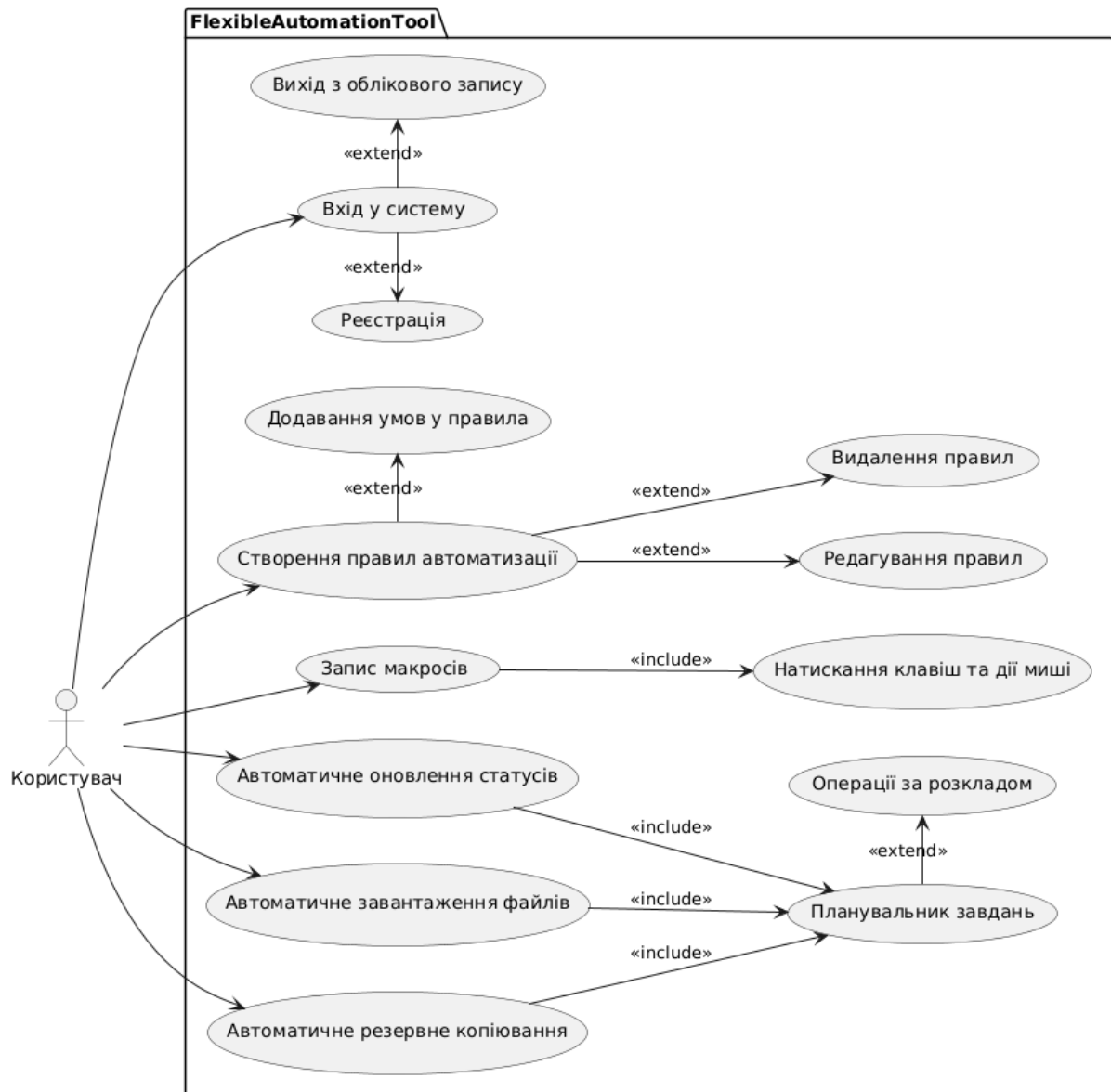


Рис 1. Діаграма прецедентів

2. Деталі сценаріїв використання

Прецедент: Вхід у систему

Короткий опис:

Користувач вводить свої облікові дані для доступу до системи.

Передумова:

Користувач має обліковий запис у системі.

Післяумова:

Користувач успішно входить в систему, отримуючи доступ до всіх функцій.

Основний сценарій:

1. Користувач відкриває програму.
2. Користувач вводить свій логін і пароль.
3. Система перевіряє введені дані.
4. Якщо дані правильні, користувач успішно входить у систему.
5. Користувач перенаправляється на головну сторінку.

Винятки:

- Якщо дані невірні, система виводить повідомлення про помилку.
- Якщо користувач забув пароль, система пропонує опцію для відновлення пароля.

2. Прецедент: Запис макросів

Короткий опис:

Користувач може записувати макроси для автоматизації задач.

Передумова:

Користувач увійшов в систему і має доступ до інтерфейсу для запису макросів.

Післяумова:

Записаний макрос зберігається в системі для подальшого використання.

Основний сценарій:

1. Користувач обирає функцію "Запис макросів".
2. Користувач натискає кнопку "Запис".
3. Система починає фіксувати натискання клавіш і дії миші.
4. Користувач виконує необхідні дії на екрані.
5. Користувач завершує запис макросу.
6. Система зберігає макрос у списку.

Винятки:

- Якщо користувач не виконав жодних дій під час запису, система попереджає, що макрос не може бути записаний.
- Якщо користувач скасовує запис, система не зберігає макрос.

3. Прецедент: Створення правил автоматизації

Короткий опис:

Користувач створює нові правила для автоматизації завдань.

Передумова:

Користувач увійшов в систему і має доступ до інтерфейсу для створення правил.

Післямова:

Нове правило автоматизації додається до системи.

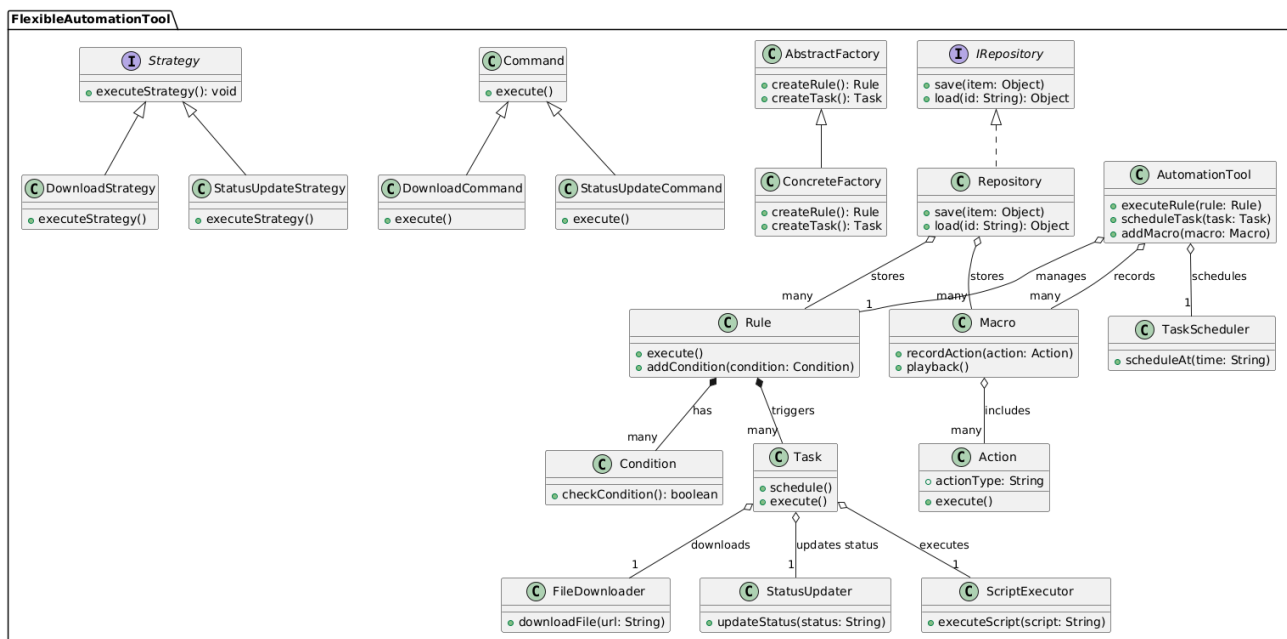
Основний сценарій:

1. Користувач обирає функцію "Створити правило".
2. Користувач вводить умови та дії для правила.
3. Користувач додає додаткові умови за потреби.
4. Користувач зберігає правило.
5. Система підтверджує створення нового правила.

Винятки:

- Якщо користувач не вводить необхідні умови, система не дозволяє зберегти правило і виводить повідомлення про помилку.
- Якщо система не може зберегти правило через технічні проблеми, виводиться повідомлення про помилку.

3. Схема класів



1. AutomationTool:

- Основний клас, який керує всіма процесами автоматизації. Він виконує правила (executeRule), планує завдання (scheduleTask) і додає макроси (addMacro).

2. Rule:

- Представляє правило автоматизації, яке включає умови та завдання. Може мати одну або кілька умов (Condition) і завдань (Task), які потрібно виконати, коли умови виконуються.

3. Condition:

- Клас, який перевіряє умови для виконання правил. Умова може бути будь-якою перевіркою (наприклад, час чи активність), яка має метод `checkCondition()` для визначення, чи виконуються умови.

4. **Task:**

- Визначає завдання, яке виконується в рамках правила. Завдання може бути планованим для виконання у певний час або може бути негайно виконано. Клас має методи для планування (`schedule`) і виконання (`execute`) завдання.

5. **Macro:**

- Клас для запису і відтворення макросів, що складаються з послідовності дій. Макроси можуть записувати натискання клавіш і дії миші за допомогою класу `Action`, а також відтворювати їх.

6. **Action:**

- Описує конкретні дії, які можуть бути записані в макросі. Кожна дія має тип (наприклад, натискання клавіші або рух миші) і метод `execute()`, що дозволяє виконати її.

7. **FileDownloader:**

- Клас для завантаження файлів через інтернет. Може бути використаний в рамках завдань, які включають завантаження файлів.

8. **StatusUpdater:**

- Клас для оновлення статусів в різних комунікаторах або програмах, наприклад, зміна статусу в Skype.

9. **ScriptExecutor:**

- Клас для виконання скриптів. Він дозволяє автоматично виконувати скрипти в рамках завдань автоматизації.

10. **TaskScheduler:**

- Клас, що відповідає за планування завдань на певний час або за розкладом.

11. **IRepository (Інтерфейс):**

- Інтерфейс для репозиторіїв, що дозволяє зберігати та завантажувати об'єкти (наприклад, правила і макроси) з бази даних чи іншого сховища.

12. **Repository:**

- Реалізація інтерфейсу `IRepository`, яка зберігає і завантажує об'єкти, такі як правила та макроси.

Взаємозв'язки:

□ **AutomationTool:**

- Клас **AutomationTool** керує виконанням **Rule** через метод `executeRule()`.
- Клас **AutomationTool** планує виконання **Task** через метод `scheduleTask()`.

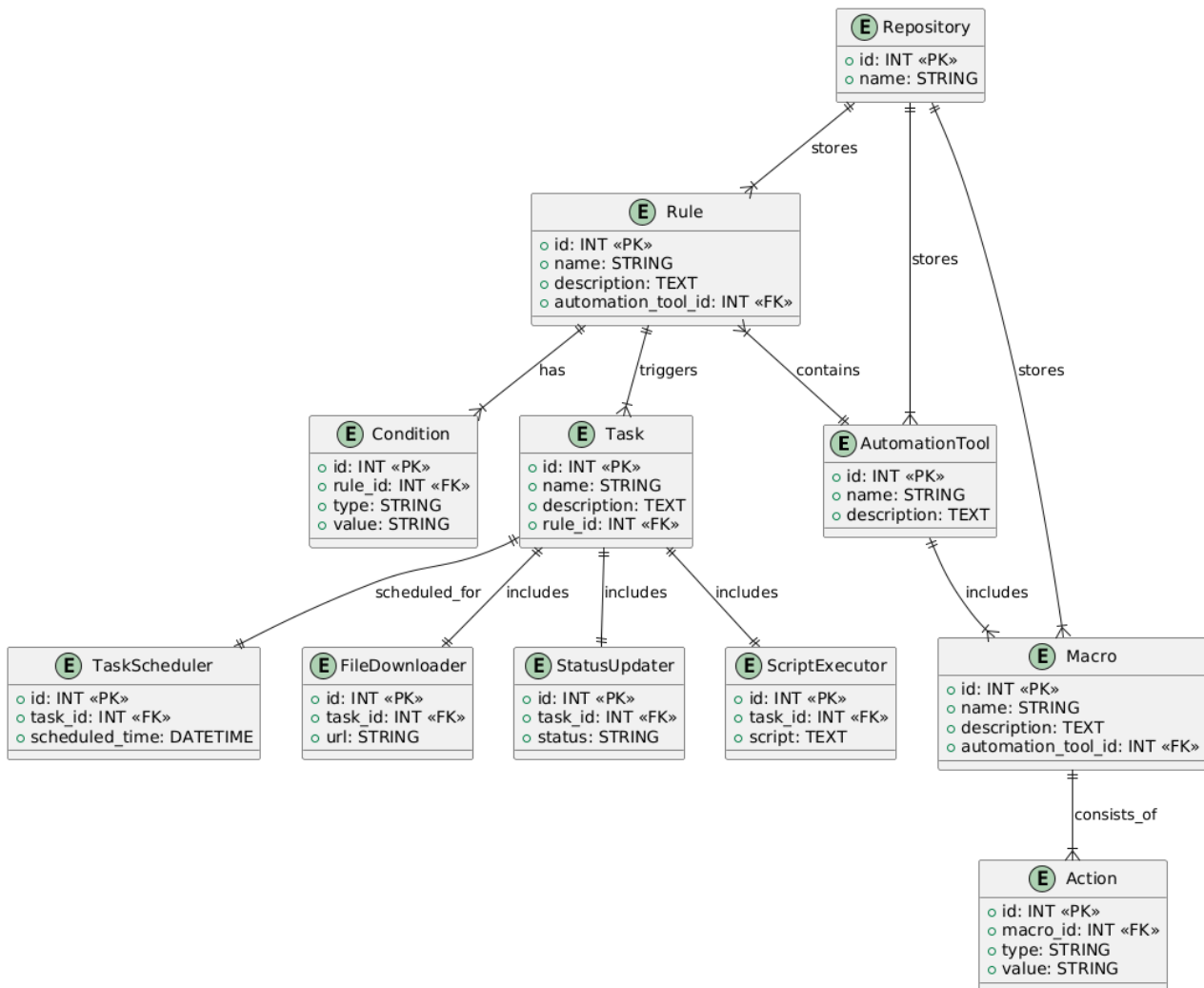
- Клас **AutomationTool** додає **Macro** через метод `addMacro()`.
- Клас **AutomationTool** взаємодіє з **TaskScheduler** для планування завдань.
- **Rule:**
 - **Rule** може мати кілька **Condition**, що перевіряються через метод `addCondition()`.
 - **Rule** містить список **Task**, що виконуються після виконання умови через методи `addTask()` (якщо такий є).
 - Коли умови виконуються, **Rule** тригерить виконання **Task**.
- **Condition:**
 - **Condition** перевіряє свої умови за допомогою методу `checkCondition()`, визначаючи, чи повинні виконуватися завдання **Task**.
- **Task:**
 - **Task** виконує операції, такі як **FileDownloader**, **StatusUpdater** та **ScriptExecutor**.
 - **Task** використовує **FileDownloader** для завантаження файлів.
 - **Task** використовує **StatusUpdater** для оновлення статусу.
 - **Task** використовує **ScriptExecutor** для виконання скриптів.
- **Macro:**
 - **Macro** містить **Action**, які записуються через метод `recordAction()`.
 - **Macro** може відтворювати дії через метод `playback()`.
- **Action:**
 - **Action** визначає конкретні дії, які виконуються в макросах (наприклад, натискання клавіші чи рух миші).
- **FileDownloader:**
 - **FileDownloader** виконує завдання завантаження файлів для **Task**.
- **StatusUpdater:**
 - **StatusUpdater** оновлює статуси в різних комунікаторах або програмах, коли це потрібно **Task**.
- **ScriptExecutor:**
 - **ScriptExecutor** виконує скрипти в рамках завдань **Task**.
- **TaskScheduler:**
 - **TaskScheduler** відповідає за планування завдань у майбутньому через метод `scheduleAt()`.
- **IRepository (Інтерфейс):**
 - **IRepository** визначає методи для збереження та завантаження об'єктів.

- **Repository** реалізує інтерфейс **IRepository** і надає методи збереження і завантаження правил і макросів.

□ **Repository:**

- **Repository** зберігає об'єкти **Rule** та **Macro**.
- **Repository** дозволяє **AutomationTool** зберігати та завантажувати правила і макроси.

4. Структура бази даних



1. AutomationTool

- ID (Primary key)** - унікальний ідентифікатор інструменту автоматизації
- Name** - назва інструменту автоматизації (STRING)
- Description** - детальний опис інструменту автоматизації (TEXT)

2. Rule

- ID (Primary key)** - унікальний ідентифікатор правила (INT)
- Name** - назва правила автоматизації (STRING)

- c) **Description** - детальний опис правила (TEXT)
 - d) **Automation_tool_id (Foreign key)** - зовнішній ключ до таблиці **AutomationTool**, що вказує на інструмент автоматизації, який містить це правило (INT)
-

3. Condition

- a) **ID (Primary key)** - унікальний ідентифікатор умови (INT)
 - b) **Rule_id (Foreign key)** - зовнішній ключ до таблиці **Rule**, що вказує на правило, до якого належить ця умова (INT)
 - c) **Type** - тип умови (STRING)
 - d) **Value** - значення, яке потрібно перевірити для умови (STRING)
-

4. Task

- a) **ID (Primary key)** - унікальний ідентифікатор завдання (INT)
 - b) **Name** - назва завдання (STRING)
 - c) **Description** - детальний опис завдання (TEXT)
 - d) **Rule_id (Foreign key)** - зовнішній ключ до таблиці **Rule**, що вказує на правило, яке активує це завдання (INT)
-

5. Macro

- a) **ID (Primary key)** - унікальний ідентифікатор макросу (INT)
 - b) **Name** - назва макросу (STRING)
 - c) **Description** - детальний опис макросу (TEXT)
 - d) **Automation_tool_id (Foreign key)** - зовнішній ключ до таблиці **AutomationTool**, що вказує на інструмент автоматизації, який містить цей макрос (INT)
-

6. Action

- a) **ID (Primary key)** - унікальний ідентифікатор дії (INT)
 - b) **Macro_id (Foreign key)** - зовнішній ключ до таблиці **Macro**, що вказує на макрос, до якого належить ця дія (INT)
 - c) **Type** - тип дії (наприклад, натискання клавіші, рух миші) (STRING)
 - d) **Value** - значення для дії (наприклад, клавіша для натискання, координати для руху миші) (STRING)
-

7. FileDownloader

- a) **ID (Primary key)** - унікальний ідентифікатор завдання завантаження файлів (INT)
- b) **Task_id (Foreign key)** - зовнішній ключ до таблиці **Task**, що вказує на завдання, яке включає завантаження файлів (INT)
- c) **URL** - адреса файлу для завантаження (STRING)

8. StatusUpdater

- a) **ID (Primary key)** - унікальний ідентифікатор завдання для оновлення статусу (INT)
- b) **Task_id (Foreign key)** - зовнішній ключ до таблиці **Task**, що вказує на завдання, яке включає оновлення статусу (INT)
- c) **Status** - новий статус для оновлення в комунікаторі (STRING)

9. ScriptExecutor

- a) **ID (Primary key)** - унікальний ідентифікатор завдання для виконання скрипта (INT)
- b) **Task_id (Foreign key)** - зовнішній ключ до таблиці **Task**, що вказує на завдання, яке включає виконання скрипта (INT)
- c) **Script** - текст скрипта для виконання (TEXT)

10. TaskScheduler

- a) **ID (Primary key)** - унікальний ідентифікатор завдання для планувальника (INT)
- b) **Task_id (Foreign key)** - зовнішній ключ до таблиці **Task**, що вказує на завдання, яке планується на виконання (INT)
- c) **Scheduled_time** - час, на який заплановане виконання завдання (DATETIME)

11. Repository

- a) **ID (Primary key)** - унікальний ідентифікатор сховища (INT)
- b) **Name** - назва сховища (STRING)

Висновки:

Робота над цією лабораторною задачею дозволила розвинути навички моделювання складних інформаційних систем за допомогою діаграм прецедентів, класів і баз даних, а також детально розглянути всі етапи розробки від планування до реалізації логіки взаємодії між компонентами системи. В результаті було створено базову архітектуру інструменту автоматизації, що може бути використана для подальшого розвитку та вдосконалення системи.

Код: <https://github.com/Lepseich/trpz/tree/main/lab2/files>