Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №3
з дисципліни «Технології розроблення програмного забезпечення»
Тема: «Основи проектування розгортання»

Виконав:
студент групи ІА-34
Тунік О.

Перевірив:
асистент кафедри ІСТ
Мягкий М.Ю.

Київ 2025

Зміст

Мета: Навчитися проєктувати діаграми розгортання та компонентів для системи що проєктується, а також розробляти діаграми взаємодії, а саме діаграми послідовностей, на основі сценаріїв зроблених в попередній лабораторній роботі.

Теоретичні відомості

Діаграма розгортання (Deployment Diagram)

Діаграми розгортання представляють фізичне розташування системи, показуючи, на якому фізичному обладнанні запускається та чи інша складова програмного забезпечення.

Головними елементами діаграми є вузли, пов'язані інформаційними шляхами. Вузол (node) – це те, що може містити програмне забезпечення. Вузли бувають двох типів. Пристрій (device) – це фізичне обладнання: комп'ютер або пристрій, пов'язаний із системою. Середовище виконання (execution environment) – це програмне забезпечення, яке саме може включати інше програмне забезпечення, наприклад операційну систему або процес-контейнер (наприклад, вебсервер). Рівень представлення (layer) – спосіб організації та розгляду моделі на одному рівні абстракції, що представляє горизонтальний зріз архітектури моделі, тоді як розбиття представляє її вертикальний зріз.

Діаграма компонентів

Діаграма компонентів UML є представленням проєктованої системи, розбитої на окремі модулі. Залежно від способу поділу на модулі розрізняють три види діаграм компонентів:

- логічні;
- фізичні;
- виконувані.

Коли використовують логічне розбиття на компоненти, то у такому разі проєктовану систему віртуально уявляють як набір самостійних, автономних модулів (компонентів), що взаємодіють між собою.

Коли на діаграмі представляють фізичне розбиття, то в такому разі на діаграмі компонентів показують компоненти та залежності між ними. Залежності показують, що класи в з одного компонента використовують класи з іншого компонента. Фізична модель використовується для розуміння які компоненти повинні бути зібрані в інсталяційний пакет. Також така діаграма показує зміни в якому компоненті будуть впливати на інші компоненти.

Діаграма компонентів розробляється для таких цілей:

- візуалізації загальної структури вихідного коду програмної системи;
- специфікації виконуваного варіанта програмної системи;

- забезпечення багаторазового використання окремих фрагментів програмного коду;
- представлення концептуальної та фізичної схем баз даних.

Діаграми послідовностей

Діаграма послідовностей (Sequence Diagram) – це один із типів діаграм у моделюванні UML (Unified Modeling Language), який використовується для моделювання взаємодії між об'єктами системи у певній послідовності часу. Вона відображає, як об'єкти обмінюються повідомленнями, показуючи порядок і логіку виконання операцій.

Діаграма складається з таких основних елементів:

- **Актори (Actors):** Зазвичай позначаються піктограмами або назвами. Це користувачі чи інші системи, які взаємодіють із системою. Актори можуть бути зовнішніми стосовно моделювання системи.
- **Об'єкти або класи**: Розміщуються горизонтально на діаграмі. Вони позначаються прямокутниками з іменем об'єкта або класу під прямокутником. Кожен об'єкт має «життєвий цикл», який представлений вертикальною пунктирною лінією (лінія життя).
- **Повідомлення**: Це лінії зі стрілками, які з'єднують об'єкти. Вони показують передачу повідомлень чи виклик методів. Стрілка може бути синхронною (звичайна стрілка) або асинхронною (лінія з відкритим трикутником) та з пунктирною лінією, що показує повернення результату.
- **Активності**: Вказують періоди, протягом яких об'єкт виконує певну дію. На діаграмі це позначається прямокутником, накладеним на лінію життя.
- **Контрольні структури**: Використовуються для відображення умов, циклів або альтернативних сценаріїв. Наприклад, блоки "alt" (альтернатива) або "loop" (цикл).

Хід роботи

Поставлене завдання

Flexible automation tool (strategy, command, abstract factory, facade, interpreter, SOA)

Інструмент автоматизації повинен забезпечувати найпростіші автоматичні дії для зручності користувача: завантаження нових фільмів / книг / файлів при випуску (наприклад, щоп'ятниці з'являються нові серії улюблених серіалів); встановити статуси в комунікаторах (skype – away при нульовій активності на тривалий час) і т.д. Автоматизація забезпечується шляхом введення правил (на зразок IFTTT.com сервісу), запису макросів (натискання клавіш, дії миші), планувальника завдань (о 5 ранку – початок роздачі торрент-файлів).

Діаграма розгортання

Проаналізувавши тему, проєктуємо діаграму розгортання відповідно до обраної теми лабораторного циклу (рис. 1).

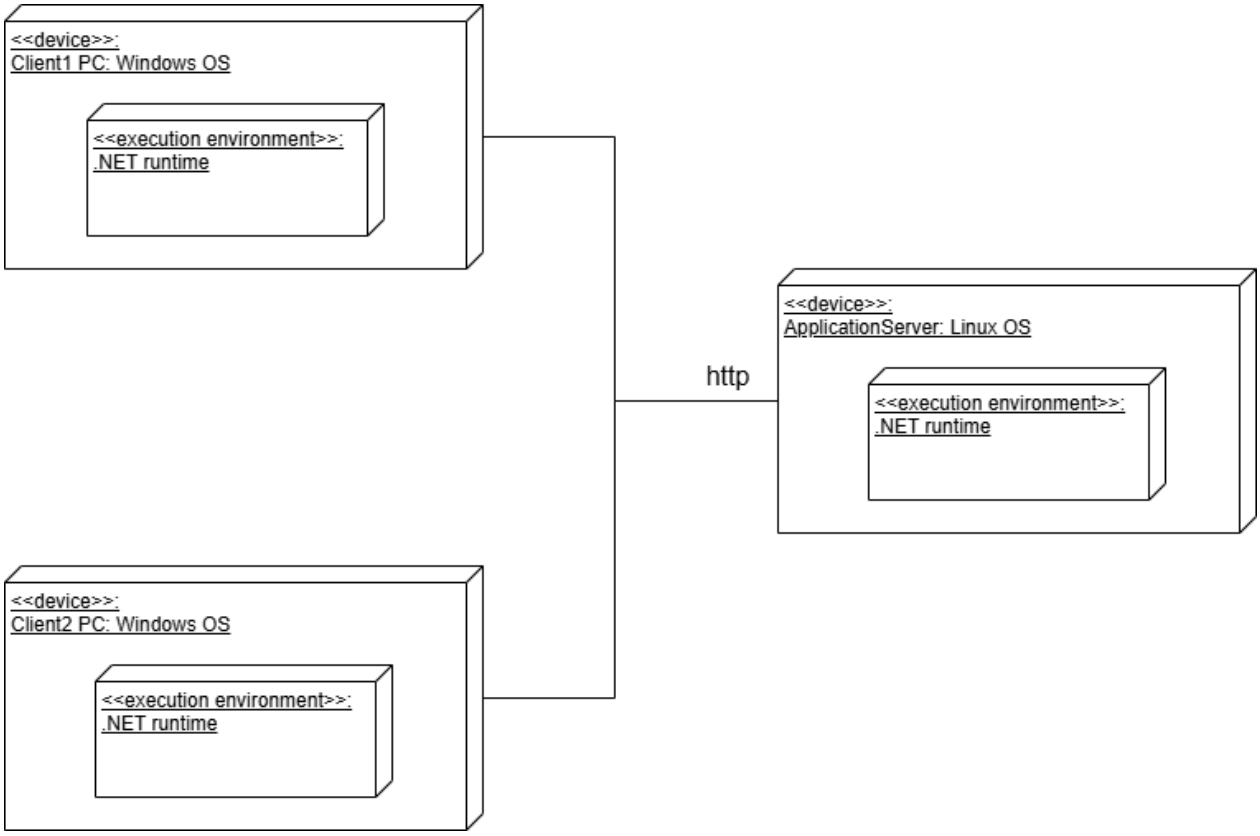

Рисунок 1. Діаграма розгортання

Діаграма показує розгортання системи FlexibleAutomationTool, де два клієнтські пристрої на Windows виконують GUI-додаток і взаємодіють із серверним сервісом, розгорнутим на Linux. Сервер обробляє запити клієнтів через

HTTP або TCP, а спільні middleware-бібліотеки використовуються на обох вузлах для забезпечення функціональності системи.

Діаграма компонентів системи

Проєктуємо діаграму компонентів предметної області (рис.2).

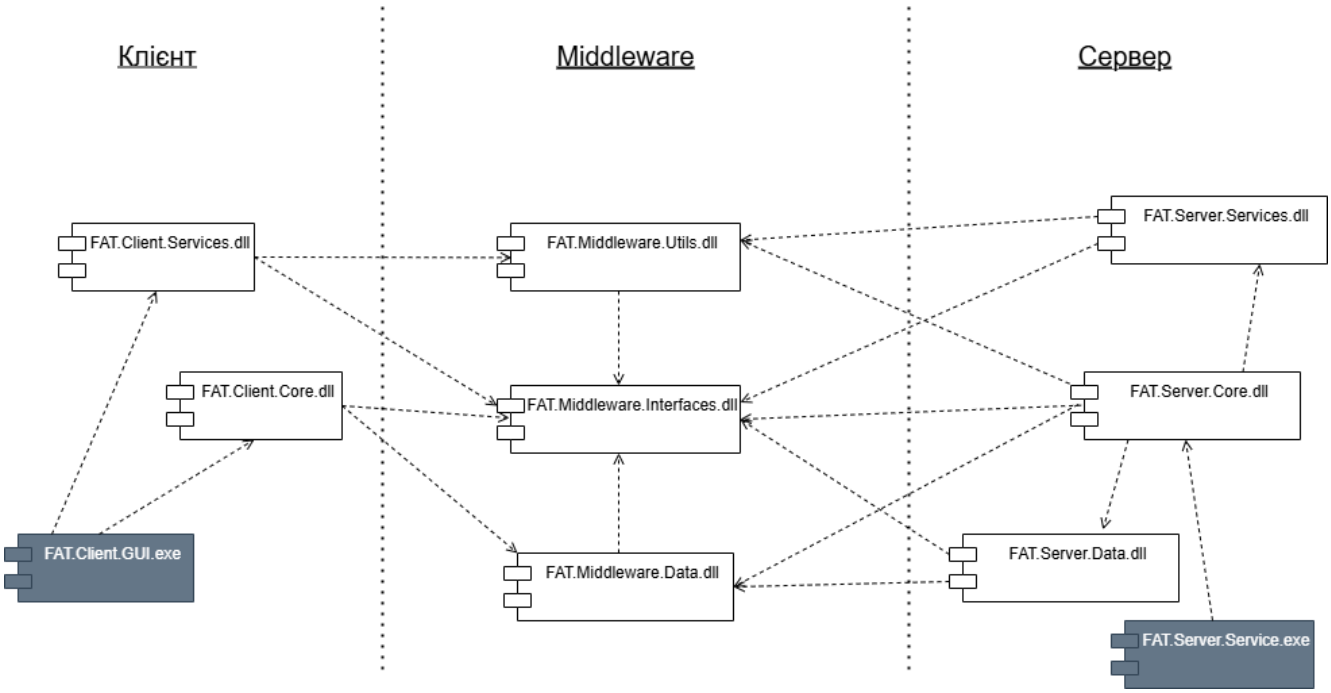

Рисунок 2. Діаграма компонентів системи

Діаграма компонентів проєкту показує, як клієнтська та серверна частини взаємодіють із загальними проміжними бібліотеками. Клієнтські компоненти (GUI, Core, Services) залежать від проміжних інтерфейсів, даних та утиліт, що забезпечує узгоджене обмінювання даними та логікою. Серверні компоненти аналогічно використовують спільні методи та дані, а також внутрішню серверну логіку, що дозволяє централізовано обробляти бізнес-процеси та дані.

Діаграми послідовності

Перетворимо описані в минулій лабораторній роботі сценарії використання у діаграми послідовності (рис. 3-5).

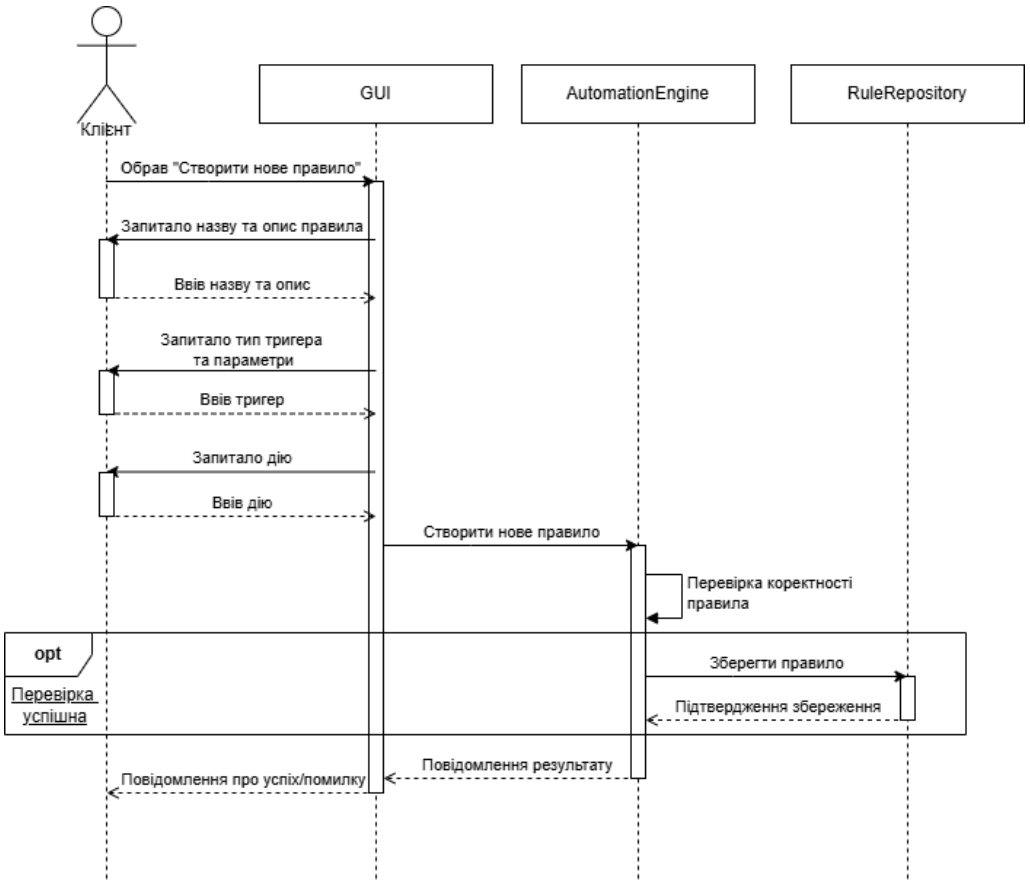

Рисунок 3. Діаграма послідовності сценарію «Створення правила автоматизації»

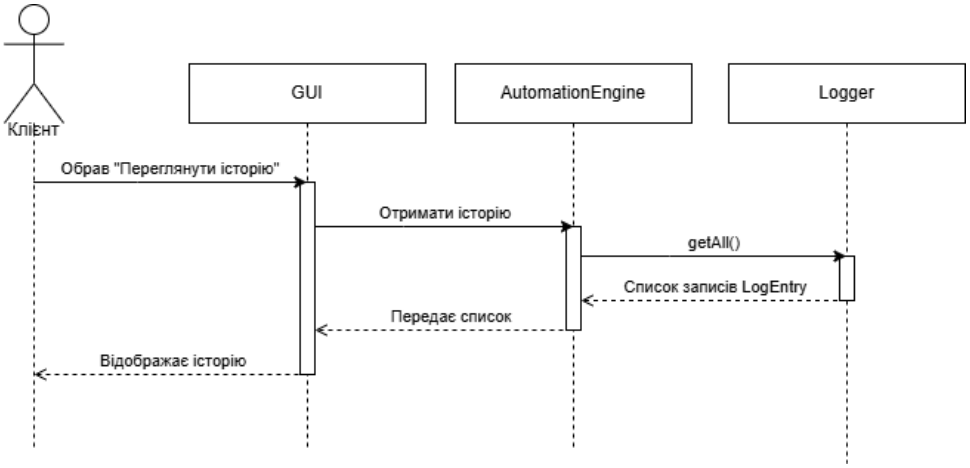

Рисунок 4. Діаграма послідовності сценарію «Перегляд історії виконаних правил»

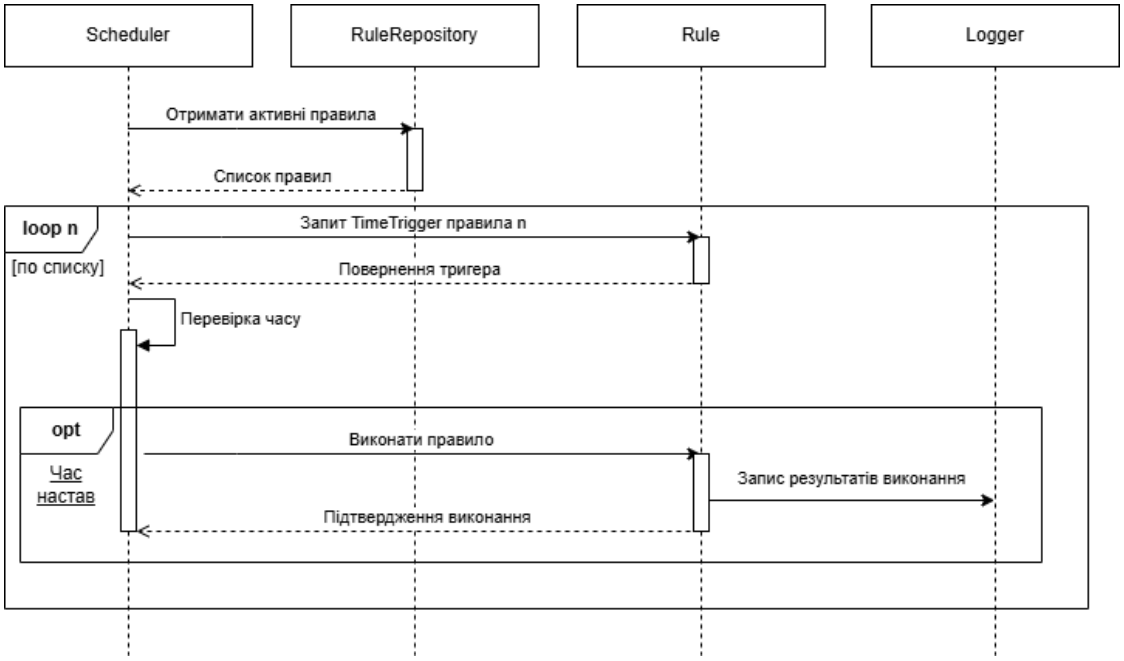

Рисунок 5. Діаграма послідовності сценарію «Автоматичне виконання правила за розкладом»

Відповіді на контрольні запитання

1. Що собою становить діаграма розгортання?

Діаграма розгортання показує фізичну архітектуру системи, включаючи розміщення програмних компонентів на апаратних вузлах і способи їх взаємодії.

2. Які бувають види вузлів на діаграмі розгортання?

На діаграмі розгортання виділяють фізичні вузли, які представляють апаратні пристрої, та програмні вузли (execution environment), де виконуються програмні компоненти.

3. Які бувають зв’язки на діаграмі розгортання?

Зв’язки на діаграмі розгортання відображають комунікаційні канали між вузлами, залежності між програмними компонентами та можливі асоціації для передачі даних.

4. Які елементи присутні на діаграмі компонентів?

На діаграмі компонентів присутні програмні компоненти, інтерфейси, пакети та залежності між ними, що показують архітектуру системи на рівні логічних одиниць.

5. Що становлять собою зв’язки на діаграмі компонентів?

Зв’язки на діаграмі компонентів відображають залежності між компонентами, реалізацію інтерфейсів, використання сервісів іншими компонентами та обмін даними.

6. Які бувають види діаграм взаємодії?

Види діаграм взаємодії включають діаграми послідовностей, діаграми комунікацій, діаграми часових подій та діаграми взаємодії із часовими рамками, які показують порядок і умови обміну повідомленнями.

7. Для чого призначена діаграма послідовностей?

Діаграма послідовностей призначена для відображення порядку викликів і обміну повідомленнями між об’єктами системи в конкретному сценарії, допомагаючи зрозуміти динамічну поведінку системи.

8. Які ключові елементи можуть бути на діаграмі послідовностей?

Ключові елементи діаграми послідовностей включають об’єкти (lifelines), повідомлення (messages), активації (activation bars), а також умови, альтернативні потоки та цикли виконання.

9. Як діаграми послідовностей пов’язані з діаграмами варіантів використання?

Діаграми послідовностей пов’язані з діаграмами варіантів використання тим, що деталізують поведінку сценаріїв, показуючи, які об’єкти взаємодіють і в якій послідовності для виконання конкретного варіанту використання.

10. Як діаграми послідовностей пов’язані з діаграмами класів?

Діаграми послідовностей пов’язані з діаграмами класів тим, що об’єкти на послідовності є екземплярами класів, а повідомлення відповідають викликам методів цих класів, демонструючи зв’язок статичної структури з динамічною поведінкою.

Висновки

Виконана лабораторна робота дозволила ознайомитися з основами проєктування розгортання та компонентів програмних систем, а також відпрацювати перетворення сценаріїв використання у діаграми послідовностей. Спроектували діаграми розгортання та компонентів для системи FlexibleAutomationTool, що демонструють взаємодію клієнтських і серверних компонентів через спільні middleware-бібліотеки та забезпечують узгоджене обмінювання даними і виконання бізнес-логіки.

Створення діаграм послідовностей за сценаріями використання дозволило наочно відобразити порядок викликів та взаємодію об’єктів системи під час основних операцій: створення правил автоматизації, їх автоматичного виконання за розкладом і перегляду історії виконаних дій. Отримані результати сприяють кращому розумінню динамічної поведінки системи та взаємозв’язку статичної та динамічної моделей програмного забезпечення.