МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

М.А. Гринченко, А.І. Роговий

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з дисципліни
«КОМП'ЮТЕРНЕ МОДЕЛЮВАННЯ ПРОЦЕСІВ ТА СИСТЕМ»
(Частина 1)
для здобувачів вищої освіти першого (бакалаврського) рівня
денної та заочної форми навчання

за спеціальністю F3 – «Комп’ютерні науки»

Харків
НТУ «ХПІ»
2025

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

М.А. Гринченко, А.І. Роговий

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з дисципліни
«КОМП'ЮТЕРНЕ МОДЕЛЮВАННЯ ПРОЦЕСІВ ТА СИСТЕМ»
(Частина 1)

для здобувачів вищої освіти першого (бакалаврського) рівня
денної та заочної форми навчання

за спеціальністю F3 – «Комп’ютерні науки»

Рекомендовано
редакційно-видавничою
радою університету,
протокол № 2 від 26.06.2025р.

Харків
НТУ «ХПІ»
2025

Методичні вказівки до виконання лабораторних робіт з дисципліни «Комп'ютерне моделювання процесів та систем» (Частина 1) для здобувачів вищої освіти спеціальності F3 «Комп’ютерні науки» першого (бакалаврського) рівня денної та заочної форми навчання / Укладачі М.А. Гринченко, А.І. Роговий – Х.: НТУ «ХПІ», 2025. – 51 с.

Укладачі: М.А. Гринченко
А.І. Роговий

Рецензент А.М. Копп

Кафедра управління проєктами в інформаційних технологіях

ЗМІСТ

ВСТУП

У сучасних умовах цифрової трансформації, штучного інтелекту та гнучкого управління проєктами, здатність моделювати складні інформаційні системи є однією з ключових професійних компетентностей фахівця в галузі комп'ютерних наук. Особливого значення вона набуває для здобувачів спеціальністю F3 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки. Штучний інтелект та управління проєктами», що реалізується на кафедрі управління проєктами в інформаційних технологіях НТУ «ХПІ».

Метою цих методичних вказівок є формування практичних навичок комп'ютерного моделювання процесів і систем з використанням сучасних засобів візуалізації, зокрема мови UML (Unified Modeling Language), яка є міжнародним стандартом моделювання програмного забезпечення. Засвоєння принципів побудови UML-діаграм сприяє розвитку аналітичного мислення, вмінню структурувати вимоги, приймати техніко-організаційні рішення, а також проектувати інформаційні системи різного рівня складності.

Матеріали методичних вказівок охоплюють типові завдання, що виникають на етапах аналізу, проєктування та впровадження IT-рішень. Структура лабораторних робіт передбачає покрокове вивчення діаграм варіантів використання, класів, послідовності, кооперації, діяльності та станів, як основних інструментів побудови функціональних та архітектурних моделей систем.

Методичні вказівки призначено для здобувачів першого (бакалаврського) рівня вищої освіти та може бути використаний під час виконання лабораторних занять, курсового проєктування, а також у самостійній роботі при підготовці до практичної діяльності в галузі інформаційних технологій та управління проєктами.

Методичні вказівки складені державною мовою, відповідають вимогам стандартів вищої освіти України та освітньо-професійної програми «Комп'ютерні науки. Штучний інтелект та управління проєктами». Видання також сприяє реалізації системного підходу до навчання, розвитку критичного мислення і практичних навичок у здобувачів вищої освіти.

Лабораторна робота №1
РОЗРОБКА ДІАГРАМ ВИКОРИСТАННЯ В НОТАЦІЇ UML

Мета роботи: ознайомитися з базовими прийомами проєктування систем та процесів з використанням універсальної мови моделювання (UML). Вивчення основних етапів проєктування та основних елементів нотації, оволодіння інтерфейсом і створення нового проєкту за допомогою CASE-засобу, вивчення діаграм варіантів використання та їх застосування в процесі постановки функціональних вимог до інформаційної системи.

1.1 Теоретична частина

Візуальне моделювання з використанням нотації UML можна уявити як процес порівневого спуску від найбільш загальної та абстрактної концептуальної моделі вихідної бізнес-системи до логічної, а потім і до фізичної моделі відповідної програмної системи. Для досягнення цих цілей спочатку будується модель у формі діаграми варіантів використання (Use Case Diagram), яка описує функціональне призначення системи.

Діаграма варіантів використання (Use Case Diagram) – діаграма, на якій зображуються відносини між акторами і варіантами використання.

Діаграма варіантів використання – це вихідне концептуальне уявлення або концептуальна модель системи в процесі її проєктування і розробки. Створення діаграми варіантів використання має наступні цілі:

- визначити спільні межі та контекст модельованої предметної області на початкових етапах проєктування системи;
- сформулювати загальні вимоги до функціональної поведінки системи;
- розробити вихідну концептуальну модель системи для її подальшої деталізації у формі логічних і фізичних моделей;
- підготувати вихідну документацію для взаємодії розробників системи з її замовниками і користувачами.

На рисунку 1.1 показано варіант використання, що описує дій адміністратора для роботи з системою.

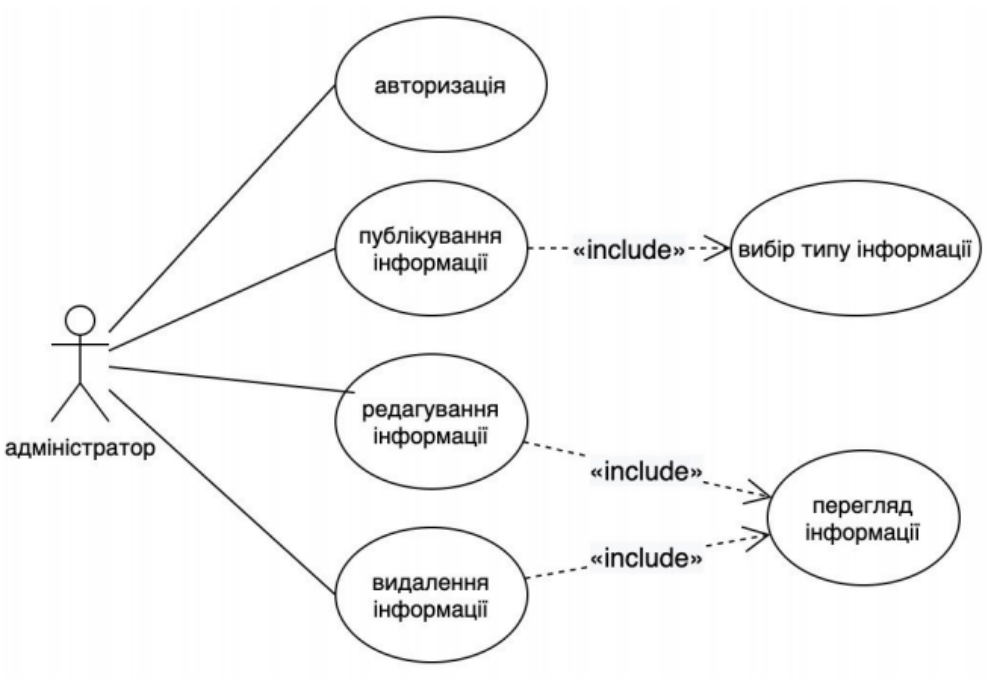

Рисунок 1.1 – Діаграма варіантів використання для адміністратора

Призначення даної діаграми складається в наступному: програмна система представляється в формі варіантів використання, з якими взаємодіють зовнішні сутності або актори.

При цьому актором або діючою особою називається будь-який об'єкт, суб'єкт або система, що взаємодіє з модельованою бізнес-системою зовні. Це може бути людина, технічний пристрій, програма або будь-яка інша система, яка служить джерелом впливу на модельовану систему так, як визначить розробник.

Кожен варіант використання визначає набір дій, який чинять над системою при діалозі з актором. При цьому нічого не говориться про те, яким чином буде реалізовано взаємодія акторів з системою і власне виконання варіантів використання.

Варіант використання є типовою взаємодією користувача та проєктованої системи і характеризується такими властивостями:

– варіант охоплює деяку очевидну для користувачів функцію;
– варіант може бути як невеликим, так і досить великим;
– варіант вирішує деяке дискретне завдання користувача.

У найпростішому випадку варіант використання створюється в процесі обговорення з користувачами тих речей, які вони хотіли б одержати від системи. Одне з головних призначень діаграми варіантів використання полягає в формалізації функціональних вимог до інформаційної системи.

Діаграма варіантів використання може служити основою для узгодження з замовником функціональних вимог до системи на ранній стадії проєктування. Будь-який з базових варіантів використання в подальшому може бути декомпозований на приватні варіанти використання.

Виконується послідовність дій, яка встановлена для даного варіанту використання. Послідовність дій виконується як відповідна реакція на повідомлення актора. При цьому актори можуть генерувати нові повідомлення для ініціювання варіантів використання. Подібна взаємодія триватиме до тих пір, поки не закінчиться виконання необхідної послідовності дій екземпляром варіанту використання, і зазначений в моделі екземпляр актора не отримає необхідний екземпляр сервісу. Закінчення взаємодії означає відсутність ініціалізації повідомлень від акторів для базових варіантів використання.

Варіанти використання можуть бути додатково специфіковані примітками з текстом, які в подальшому можуть стати прототипами операцій і методів спільно з атрибутами. Подальша розробка моделей пов'язана з реалізацією варіантів використання у вигляді графа діяльності за допомогою CASE-засобу. Взаємодія між варіантами використання і акторами може уточнюватися на діаграмі кооперації, коли описуються взаємозв'язки між системою, що містить ці варіанти використання, і оточенням або зовнішнім середовищем цієї системи.

Реалізація варіанту використання залежить від типу елемента моделі, в якому він визначений. Стосовно до бізнес-систем варіанти використання можуть реалізуватися співробітниками цієї системи. У всіх випадках елементи системи повинні взаємодіяти один з одним для спільного забезпечення необхідного поводження і виконання варіантів використання моделі.

1.2 Практична частина

За допомогою CASE-засобу створити модель інформаційної системи, для чого побудувати діаграми варіантів використання для опису предметного середовища, яке задано у відповідно до індивідуального варіанту завдання.

1. Для розробки діаграми варіантів використання рекомендується послідовність дій:

– визначити головних або первинних і другорядних акторів;

– визначити цілі головних акторів по відношенню до системи;

– сформулювати основні варіанти використання, які специфікують функціональні вимоги до системи:

– організувати варіанти використання за рівнем зменшення ризику їх реалізації;

– розглянути всі базові варіанти використання в порядку зменшення їх ступеня ризику;

– виділити учасників, інтереси, передумови і постумови виконання обраного варіанту використання;

– написати успішний сценарій реалізації обраного варіанту використання;

– визначити виключення або неуспіх у виконанні сценарію варіанти використання;

– написати сценарії для всіх винятків;

– виділити загальні варіанти використання і зобразити їх взаємозв'язку з базовими зі стереотипом << include >>;

– виділити варіанти використання для винятків і зобразити їх взаємозв'язку з базовими зі стереотипом << extend >>;

– перевірити діаграму на відсутність дублювання варіантів використання і акторів.

Зберегти файл з діаграмою варіантів використання для подальших робіт.

Для виконання лабораторної роботи може бути використаний будь-який CASE-засіб UML діаграм (Rational Software, Visual Paradigm Community Edition, StarUML та інші).

2. Оформити звіт про лабораторну роботу.

Звіт з лабораторної роботи повинен бути оформлений відповідно до вимог і складатися з наступних структурних елементів: титульний лист, текстова частина, додаток: розроблена модель варіантів використання.

Текстова частина звіту повинна включати пункти:

– умови задачі;

– порядок виконання;

– словник термінів – короткі відомості про склад і компонентах побудованої моделі.

3. Представити звіт про лабораторну роботу для захисту.

Захист звіту з лабораторної роботи полягає у надані викладачу отриманих результатів у вигляді файлу і демонстрації отриманих навичок при відповідях на контрольні питання викладача.

1.3 Контрольні запитання

1 Що означає абревіатура UML?
2 Кому і навіщо потрібен UML?
3 Що послужило причиною виникнення UML?
4 Для чого використовують UML на практиці?
5 Як визначено UML?
6 З чого складається UML?
7 Що таке модель UML?
8 З яких елементів складається модель?
9 Як комбінуються елементи моделі?
10 Яка загальна структура моделі?
11 У яких випадках доцільно застосовувати моделювання використання?
12 Що таке моделювання використання і навіщо воно потрібне?
13 Які засоби застосовуються при моделюванні використання?
14 Як ідентифікувати варіанти використання і акторів?
15 Як реалізуються варіанти використання?

Лабораторна робота №2.
РОЗРОБКА ДІАГРАМИ КЛАСІВ В НОТАЦІЇ UML

Мета роботи: вивчення основних елементів діаграми класів та створення діаграми класів, а також отримання навичок роботи з CASE-засобами моделювання інформаційних систем.

2.1 Теоретична частина

Діаграма класів (class diagram) використовуються для моделювання статичного виду системи з точки зору проєктування. Діаграма класів – діаграма, на якій показано безліч класів, інтерфейсів, кооперацій і відносин між ними.

Використовується для наступних цілей:

– для моделювання словника системи: передбачає прийняття рішення про те, які абстракції є частиною системи, а які - ні. За допомогою діаграм класів можна визначити ці абстракції та їх обов'язки;

– для моделювання простих кооперацій. Кооперація - це спільнота класів, інтерфейсів і інших елементів, що працюють спільно для забезпечення деякого кооперативного поведінки;

– для моделювання логічної схеми бази даних.

Існують три різні точки зору на побудову діаграм класів або будь-який інший моделі:

– концептуальна точка зору - діаграми класів які допомагають досліджувати предметну область. Концептуальна модель може мати слабке відношення або взагалі не мати ніякого відношення до її програмного забезпечення, що реалізується, тому її можна розглядати без прив'язки до якоїсь мови програмування;

– точка зору специфікації – розглядається програмна система, при цьому розглядається лише її інтерфейси, але не реалізація;

– точка зору реалізації – класи діаграми відповідають реальним класам програмної системи.

Розглядати діаграму класів рекомендується з концептуальної точки зору, яка використовується на початкових етапах моделювання та розробки.

Діаграма класів – це найбільш часто використовуваний тип діаграм, які створюються при моделюванні інформаційних систем. Показують набір класів, інтерфейсів і кооперацій, а також їх зв'язки.

На практиці діаграми класів застосовують для моделювання статичного уявлення системи (здебільшого це моделювання словника системи, кооперацій

або схем). Крім того, діаграми даного типу є основою для цілої групи взаємопов'язаних діаграм – діаграм компонентів і діаграм розміщення. Діаграми класів важливі не тільки для візуалізації і документування структурних моделей, а також для конструювання виконуваних систем за допомогою прямого і зворотного проектування.

Діаграма класів UML є різновидом статичної структурної діаграми, яка демонструє класи системи, їх атрибути, операції (або методи) і взаємозв'язки між об'єктами. У верхній частині діаграми задається ім'я класу. Посередині розташовуються поля (атрибути) класу. Нижня частина містить методи класу.

На діаграмах класів зазвичай представлені наступні елементи: класи, інтерфейси, залежності, узагальнення та асоціації.

Опис класу може включати безліч різних елементів, в мові передбачено групування елементів опису класу по розділах.

Стандартних розділів три:

– розділ ім'я – поряд з обов'язковим ім'ям може містити також стереотип, кратність і список властивостей;

– розділ атрибутів – містить список описів атрибутів класу;

– розділ операцій – містить список операцій класу.

Операція (метод) – це реалізація методу класу. Клас може мати будь-яке число операцій або не мати жодної. Часто виклик операції об'єкта змінює його атрибути.

У мові UML прийнята певна стандартизація записи атрибутів класу, яка підпорядковується деяким синтаксичним правилам. Кожному атрибуту класу відповідає окремий рядок тексту, яка складається з квантора видимості атрибута, імені атрибута, його кратності, типу значень атрибута і, можливо, його початкового значення:

<Квантор видимості> <ім'я атрибута> [кратність]:

<Тип атрибута> = <початкове значення> {рядок-властивість}

Квантор видимості може приймати одне з трьох можливих значень і, відповідно, відображається за допомогою спеціальних символів:

1. Символ "+" позначає атрибут з областю видимості типу загальнодоступний (public). Атрибут із цією областю видимості доступний або видимий з будь-якого іншого класу пакета, в якому визначена діаграма.

2. Символ "#" позначає атрибут з областю видимості типу захищений (protected). Атрибут із цією областю видимості недоступний або не видимий для всіх класів, за винятком підкласів даного класу.

3. Знак "-" позначає атрибут з областю видимості типу закритий (private). Атрибут із цією областю видимості недоступний або не видимий для всіх класів без виключення.

Кратність атрибута характеризує загальна кількість конкретних атрибутів даного типу, що входять до складу окремого класу. У загальному випадку кратність записується в формі рядка тексту в квадратних дужках після імені відповідного атрибута. Значення кратності з інтервалу йдуть в монотонно зростаючому порядку без пропуску окремих чисел, що лежать між нижньою і верхньою межами. При цьому дотримуються наступного правила: відповідні нижні і верхні межі інтервалів включаються в значення кратності. Якщо в якості кратності вказується однина, то кратність атрибута приймається рівною даному числу. Якщо ж вказується єдиний знак "*", то це означає, що кратність атрибута може бути довільним позитивним цілим числом або нулем.

Крім внутрішнього устрою або структури класів на відповідній діаграмі вказуються різні відносини між класами. При цьому сукупність типів таких відносин фіксована в мові UML і зумовлена семантикою цих типів відносин. Базовими відносинами або зв'язками в мові UML є:

– залежність означає таке ставлення між класами, при якому зміна специфікації класу-постачальника може вплинути на роботу залежного класу, але не навпаки;

– асоціація показує, що об'єкти однієї сутності (класу) пов'язані з об'єктами іншої сутності таким чином, що можна переміщатися від об'єктів одного класу до іншого. асоціація є загальним випадком композиції і агрегації;

– агрегація – це різновид асоціації при відношенні між цілим і його частинами;

– композиція – більш суворий варіант агрегації за значенням, при якому є жорстка залежність між часом існування примірників класу і типом примірників класів;

– реалізація - відношення між двома елементами моделі, в якому один елемент (клієнт) реалізує поведінку, задану іншим (постачальником);

– узагальнення показує, що один з двох пов'язаних класів (підтип) є приватною формою іншого (надтип), який називається узагальненням першого.

Приклад класу представлений на рисунку 2.1.

Рисунок 2.1 – Приклад класу та типи взаємозв'язку

Стрілка може позначатися необов'язковим, але стандартним ключовим словом в лапках і необов'язковим індивідуальним ім'ям. Для відносини залежності зумовлені ключові слова, які позначають деякі спеціальні види залежностей. Ці ключові слова (стереотипи) записуються в лапках поруч зі стрілкою, яка відповідає даній залежності. Приклади стереотипів для відносини залежності представлені нижче:

"Access" – служить для позначення доступності відкритих атрибутів і операцій класу-джерела для класів-клієнтів;

"Bind" – клас-клієнт може використовувати деякий шаблон для своєї подальшої параметризації;

"Derive" – атрибути класу-клієнта можуть бути обчислені по атрибутах класу-джерела;

"Import" – відкриті атрибути і операції класу-джерела стають частиною класу-клієнта, як якщо б вони були оголошені безпосередньо в ньому;

"Refine" – вказує, що клас-клієнт служить уточненням класу-джерела в силу причин історичного характеру, коли з'являється додаткова інформація в ході роботи над проектом.

При моделюванні поведінки проектованої або аналізованої системи виникає необхідність не тільки представити процес зміни її станів, але і деталізувати особливості алгоритмічної і логічної реалізації виконуваних системою операцій.

Структурована діаграма класів:

– сфокусована на одному аспекті статичного уявлення дизайну системи;
– містить тільки ті елементи, які істотні для розуміння цього аспекту;

– забезпечує деталізацію, що відповідає рівню абстракції діаграми, включаючи тільки доповнення, важливі для розуміння;

– не так спрощено, щоб створити у читача хибне уявлення про важливу семантиці;

– присвоюйте їй ім'я, відповідне її призначенням;

– уникайте перетину ліній або хоча б мінімізуйте його;

– організуйте елементи так, щоб семантичні близькі суті розташовувалися поруч;

– використовуйте примітки і кольори в якості міток, які акцентують увагу на важливих деталях діаграми;

– постарайтеся не показувати занадто багато видів зв'язків. Взагалі, на кожній діаграмі класів повинен домінувати тільки один вид зв'язків.

Приклад діаграми класів представлено на рисунку 2.2.

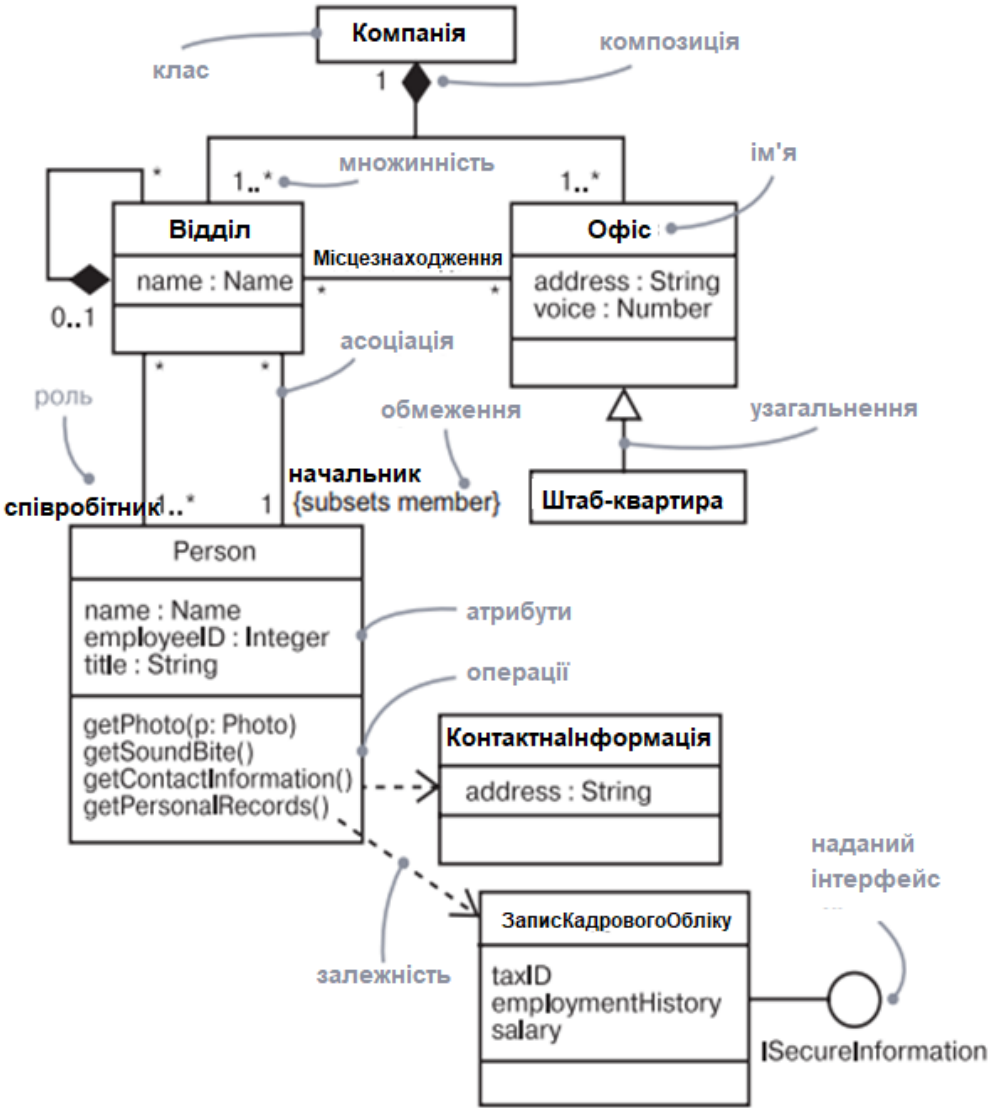

Рисунок 2.2 – Приклад діаграми класів

2.2 Практична частина

За допомогою CASE-засобу побудувати діаграму класів для опису інформаційної системи, яка задана відповідно до індивідуального варіанту завдання.

Для розробки діаграми класів використання рекомендується послідовність дій:

1. Створити діаграму класів для одного з сценаріїв діаграми варіантів використання, створеної в попередній лабораторній роботі.

2. Ідентифікувати ті класи в моделі, стан яких не повинен бути залежним від часу життя у програмному забезпеченні.

3. Для кожного класу необхідно задати атрибути і операції. Кожен клас повинен бути детально задокументований – необхідно задати текстовий опис самого класу, опису його атрибутів і операцій, визначити видимість кожного атрибута і операції.

4. Створити діаграму класів, яка містить класи, виявлені на першому етапі, визначити власний набір стереотипів і помічених значень, щоб представити специфічні для інформаційної системи деталі.

5. Розкрити структурні подробиці цих класів. в основному це означає необхідність детально уточняти їх атрибути, а також асоціації із зазначенням множинності, які пов'язують дані класи.

6. Розглянути поведінку класів, зазначених на першому етапі, розкривши операції, істотні для доступу до даних і забезпечення їх цілісності. взагалі, щоб чіткіше розмежувати різні аспекти системи, слід інкапсулювати бізнес правіла, що описують маніпуляції наборами цих об'єктів, що знаходиться над даними класами.

7. Виявити загальні зразки, які ускладнюють фізичне проєктування інформаційної системи, наприклад циклічні асоціації, визначити початковий набір класів, встановити між ними зв'язки, вказати основні дані, що зберігаються в об'єктах. Застосувати усі базові відносини між ідентифікованими класами в моделі.

8. Розробити пакети для групування класів, створених в пункті 1;

9. Згрупувати класи з пункту 1 в пакети.

10. Для кожного пакета створити свою діаграму класів.

11. Розробити головну діаграму класів.

12. Провести повірку діаграми класів на структурованість.

Зберегти файл з діаграмою класів для подальших робіт.

Для виконання лабораторної роботи може бути використаний будь-який CASE-засіб UML діаграм (Rational Software, Visual Paradigm Community Edition, StarUML та інші).

Оформити звіт про лабораторну роботу.

Звіт з лабораторної роботи повинен бути оформлений відповідно до вимог і складатися з наступних структурних елементів: титульний лист, текстова частина, додаток: розроблені діаграми класів.

Представити звіт про лабораторну роботу для захисту.

Захист звіту з лабораторної роботи полягає у надані викладачу отриманих результатів у вигляді файлу і демонстрації отриманих навичок при відповідях на контрольні питання викладача.

2.3 Контрольні запитання

1. Яке призначення діаграми класів?
2. Перелічите основні елементи діаграми класів?
3. Що таке клас та його структура?
4. Які використовуються рівні видимості атрибутів та операцій ?
5. Які існують стереотипи класів?
6. Що таке інтерфейс та його структура?
7. Що таке відношення «асоціація» в діаграмі класів?
8. Що таке відношення «узагальнення» в діаграмі класів?
9. Що таке відношення «залежність» в діаграмі класів?
10. Що таке відношення «реалізація» в діаграмі класів?
11. Що таке відношення «агрегація» в діаграмі класів?
12. Що таке відношення «композиція» в діаграмі класів?
13. Опішить правила моделювання діаграми класів?
14. Що таке структурована діаграма класів?
15. Які існують точки зору до побудови діаграми класів?

Лабораторна робота №3.
РОЗРОБКА ДІАГРАМ ПОСЛІДОВНОСТІ ТА КООПЕРАЦІЇ В НОТАЦІЇ UML

Мета роботи: вивчення основних елементів діаграм послідовності і кооперації та розробка цих діаграм, а також отримання навичок роботи з інструментальними засобами.

3.1 Теоретична частина

Для моделювання взаємодії об'єктів в мові UML використовуються відповідні діаграми взаємодії. При цьому враховуються два аспекти: по-перше, взаємодії об'єктів можна розглядати в часі, по-друге для передачі та прийому повідомлень між об'єктами. В першому випадку для представлення тимчасових особливостей передачі та прийому повідомлень між об'єктами використовується діаграма послідовності об'єктів. У другому для представлення структурних особливостей передачі і прийому повідомлень між об'єктами використовується діаграма кооперації.

Діаграми послідовності дозволяють моделювати лінію життя об'єкта, яка описує його існування в певний період часу. Діаграми послідовності відображають потік подій, що відбувається в рамках варіанту використання. На цих діаграмах відображаються тільки ті об'єкти, які безпосередньо беруть участь у взаємодії, тому ключовим моментом є саме динаміка взаємодії об'єктів в часі і не використовуються можливі статичні асоціації з іншими об'єктами.

При цьому діаграма послідовності має два виміри. Один – зліва направо у вигляді вертикальних ліній, кожна з яких зображує лінію життя окремого об'єкта, який бере участь у взаємодії. Другий вимір – вертикальна тимчасова лінія, спрямована зверху вниз. При цьому взаємодії об'єктів реалізуються за допомогою повідомлень, які надсилаються одними об'єктами іншим.

Лінія життя об'єкту зображується пунктирною вертикальною лінією, асоційованою з єдиним об'єктом на діаграмі послідовності. Лінія життя служить для позначення періоду часу, протягом якого об'єкт існує в системі і, отже, може потенційно брати участь у всіх її взаємодіях.

Існує кілька видів повідомлень: просте, синхронне, повідомлення з відмовою ставати в чергу та ін. *Просте повідомлення* використовується за замовчуванням, це означає, що всі повідомлення виконуються в одному потоці управління. *Синхронне* застосовується, коли клієнт посилає повідомлення і чекає відповіді користувача. *Повідомлення з відмовою ставати в чергу*: клієнт посилає

повідомлення серверу і, якщо сервер не може негайно прийняти повідомлення, воно скасовується. *Повідомлення з лімітованим часом очікування*: клієнт посилає повідомлення серверу, а потім чекає вказаний час; якщо протягом цього часу сервер не приймає повідомлення, воно скасовується. *Асинхронне повідомлення*: клієнт посилає повідомлення серверу і продовжує свою роботу, не чекаючи підтвердження про отримання.

Повідомлення мають стереотипи, які використовуються для відображення змісту взаємозв’язку з об’єктами. Ключові слова для опису часу життя об'єктів представлені у таблиці 3.1.

Таблиця 3.1 – Ключові слова для опису часу життя об'єктів

Ключові слова	Спосіб використання	Опис
create	стереотип операції в повідомленні	Операція створює об'єкт, тобто дане повідомлення є викликом конструктора
destroy	стереотип операції в повідомленні	Операція знищує об'єкт тобто дане повідомлення є викликом деструктора
destroyed	обмеження ролі класифікатора	Об'єкт знищується в процесі описуваного взаємодії
new	обмеження ролі класифікатора	Об'єкт створюється в процесі описуваного взаємодії
transient	обмеження ролі класифікатора	Об'єкт створюється і знищується в процесі описуваного взаємодії. Такий об'єкт називається тимчасовим. Дане обмеження еквівалентно одночасного вказівкою обмежень new і destroyed

Приклад діаграми послідовності наведено на рисунку 3.1.

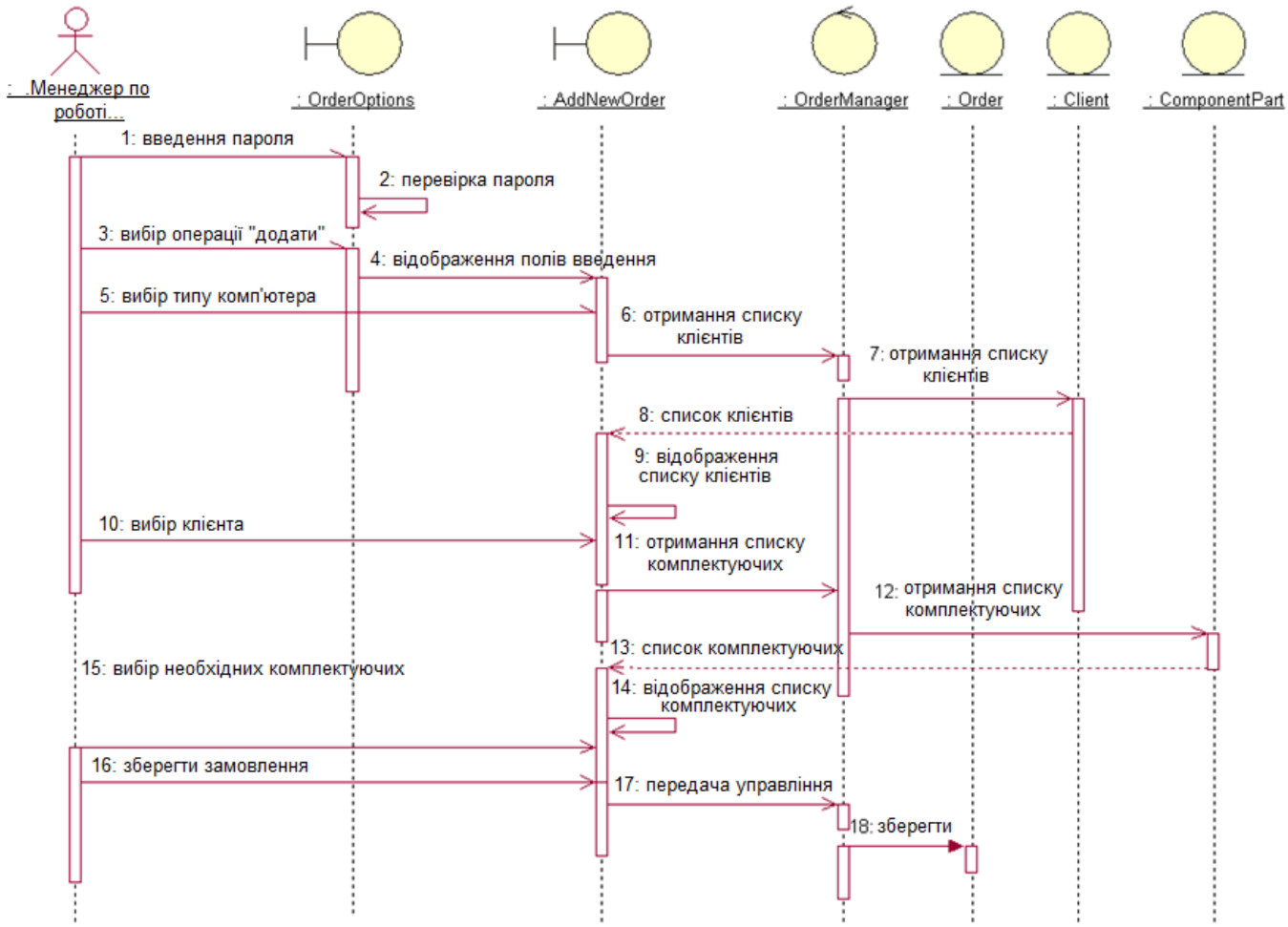

Рисунок 3.1 – Приклад діаграми послідовності

Подібно до діаграм послідовності, діаграми кооперації відображають потік подій в конкретному сценарії варіантів використання.

Головна особливість діаграми кооперації полягає в можливості графічно представити не тільки послідовність взаємодії, але і всі структурні відносини між об'єктами, які беруть участь в цій взаємодії.

Окремі аспекти специфікації об'єктів як елементів діаграм вже розглядалися раніше при описі діаграм послідовності. Ці ж об'єкти є основними елементами у тому числі в діаграмі кооперації. Для графічного зображення об'єктів використовується такий же символ прямокутника, що і для класів.

Об'єкт є окремим екземпляром класу, який створюється на етапі виконання програми. Він може мати своє власне ім'я і конкретні значення атрибутів. Для позначення ролі класифікатора необхідно вказати або ім'я класу (разом з двокрапкою), або ім'я ролі (разом з похилою рискою). В іншому випадку прямокутник буде відповідати звичайному класу.

В діаграмі кооперації визначають види об’єктів: активний та пасивний об’єкти, мультіоб’єкт, складений об'єкт.

В контексті мови UML всі об'єкти діляться на дві категорії: пасивні та активні наведені на рисунку 3.2. Пасивний об'єкт оперує тільки даними і не може ініціювати діяльність з управління іншими об'єктами. Активний об'єкт має свій процес управління і може ініціювати діяльність з управління іншими об'єктами.

Рисунок 3.2 – Приклад пасивного та активного об'єкту

Мультіоб'єкт являє собою безліч об'єктів на одному з кінців асоціації. На діаграмі кооперації мультіоб’єкт використовується щоб показати операції і сигнали, адресовані безлічі об'єктів, а не тільки одного.

Складений об'єкт або об'єкт-контейнер призначений для представлення об'єкта, що має власну структуру і внутрішні потоки управління. Складений об'єкт є екземпляром. На діаграмах кооперації складений об'єкт складається з двох секцій: верхньої і нижньої. У верхній секції записується ім'я складеного об'єкта, а в нижній - його елементи, які можуть бути складовими об'єктами. Приклад складеного об'єкту наведено на рисунку 3.3.

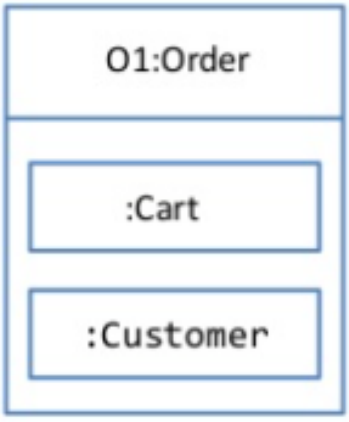

Рисунок 3.3 – Приклад складеного об'єкту

Зв'язок є екземпляром або прикладом асоціації. Зв'язок як елемент мови UML може мати місце між двома і більше об'єктами. Зв'язок на діаграмі кооперації відображається відрізком прямої лінії, що з'єднує два прямокутника об'єктів. На кожному з кінців цієї лінії можуть бути явно вказані імена ролей даної асоціації.

За допомогою діаграми кооперації легше зрозуміти потік подій і відносини між об'єктами, проте важче усвідомити послідовність подій, тому для сценарію створюють діаграми обох типів.

На відміну від діаграми послідовності, на діаграмі кооперації основна увага приділяється структурі взаємодії. Крім загальних елементів (примірників дійових осіб, об'єктів і повідомлень).

Структурована діаграма взаємодії наділене наступними ознаками:

– описує тільки ті об'єкти, які працюють разом для забезпечення поведінки, більшого ніж поведінка суми цих об'єктів, взятих порізно;

– має чіткий контекст і може представляти взаємодію об'єктів в контексті операції, класу або системи в цілому;

– ефективно забезпечує потрібну поведінку з оптимальним балансом часу і ресурсів;

– здатна до адаптації тобто, елементи взаємодії, які можуть змінюватися, повинні бути ізольовані з тим, щоб їх можна було легко модифікувати.

Приклад діаграми кооперації наведено на рисунку 3.4.

Для відображення взаємодії в UML потрібно:

– вибрати засіб відображення взаємодії: або за хронологічним порядком повідомлень, або по їх розташуванню в контексті якоїсь структурної організації об'єктів, при цьому застосовувати обидва способи одночасно не можна;

– звернути увагу на те, що події в деяких субпослідовностях впорядковані лише частково;

– показувати тільки ті властивості кожного об'єкта (значення атрибутів, ролі і стану), які важливі для розуміння взаємодії в його контексті;

– показувати лише ті властивості кожного повідомлення (параметри, семантику паралельності, повернені значення), які важливі для розуміння взаємодії в його контексті.

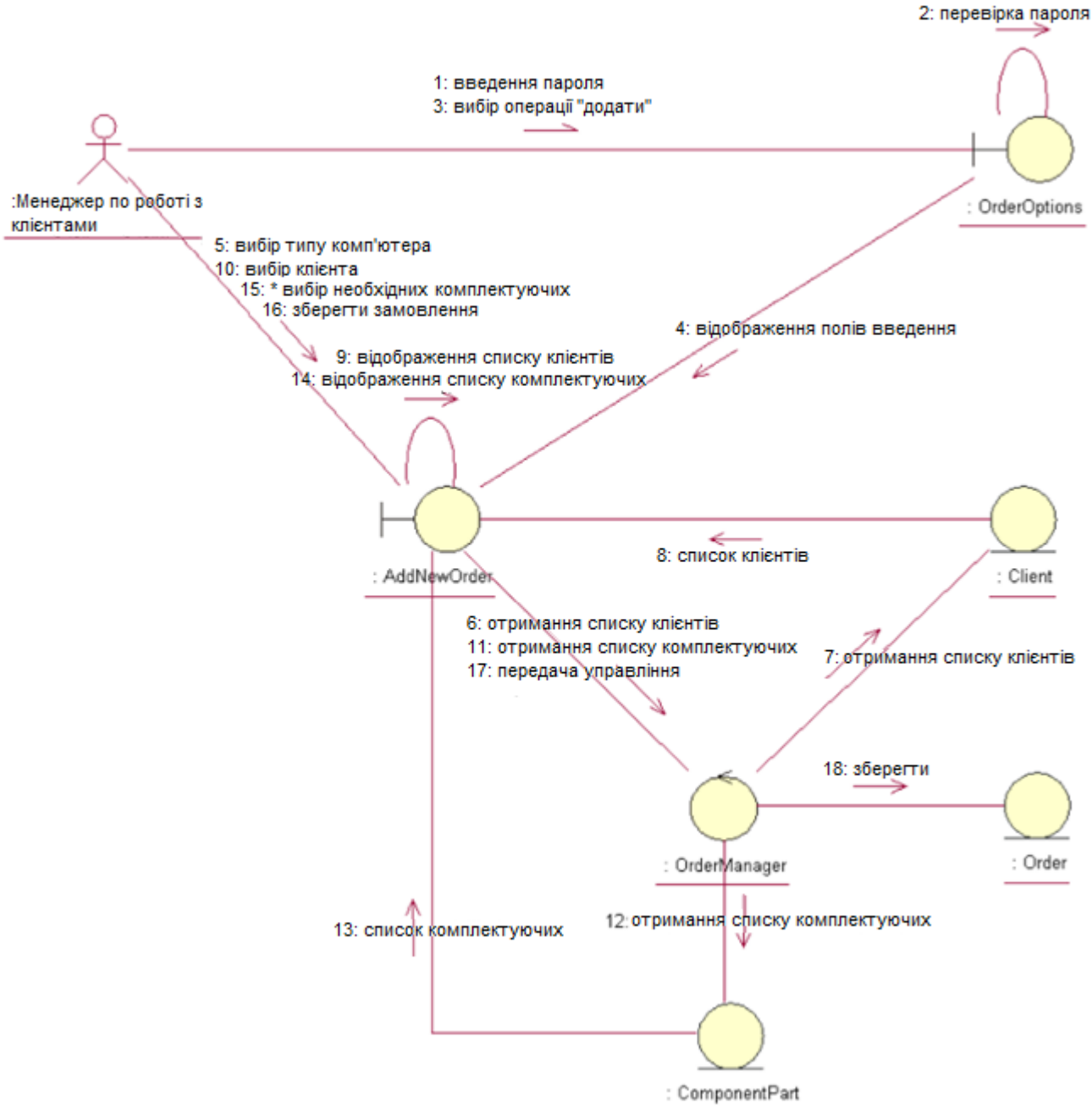

Рисунок 3.4 – Приклад діаграми кооперації

3.2. Практична частина

За допомогою CASE-засобу побудувати діаграми послідовності та кооперації для опису предметного середовища, яке задано відповідно до індивідуального варіанту завдання.

1. Для розробки діаграми послідовності рекомендується послідовність дій:

– встановити контекст взаємодії (тобто визначити, чи йде мова про контекст системи, підсистеми, операції або класу, або одного з сценаріїв варіанту використання, або кооперації);

– встановити основу взаємодії, визначені об'єкти, які грають роль у взаємодії. розташувати їх на діаграмі послідовності зліва направо від більш важливих об'єктів;

– зобразити лінію життя кожного об'єкта. Більшість з них існує протягом усієї взаємодії. Для тих же, які створюються і знищуються під час взаємодії, встановіть лінії життя відповідним чином, явно позначивши «народження» і «смерть» об'єкта повідомленнями зі стереотипами;

– починаючи з повідомлення, яке ініціює взаємодію, розташуйте всі наступні зверху вниз між лініями життя, показуючи властивості кожного повідомлення (зокрема, його параметри), наскільки це необхідно для пояснення семантики взаємодії;

– візуалізувати вкладеність повідомлень або моменту часу, в який виконується конкретне обчислення;

– забезпечте лінію життя кожного об'єкта його фокусом управління;

– використовуйте синхронні, асинхронні та інші типи повідомлень;

– уточняти тимчасові або просторові обмеження, забезпечте кожне повідомлення часовою міткою і приєднайте відповідні обмеження часу і простору;

– уточняти потік управління формально, підключіть до кожного повідомлення перед і постумови.

2. Для розробки діаграми кооперації рекомендується враховувати наступне:

– на діаграмі кооперації при запису повідомлень потрібно використовувати стереотипи, які були розглянуті раніше при побудові діаграми послідовності. Їх семантика і синтаксис залишаються без зміни, оскільки визначені в нотації мови UML;

– для обраного варіанту використання необхідно перенести з діаграми класів всі класи, які беруть участь в ньому, а з діаграми варіантів використання - дійові осіби;

– на діаграмі кооперації між класами слід відобразити асоціації, перенесені з діаграми класів, а також додати асоціації, що зв'язують дійових осіб з граничними класами;

– на стадії аналізу імена повідомленням можна давати довільно або користуватися типовими іменами (стереотипами).

– імена повідомлень повинні відповідати методам класів;

– процес побудови діаграми кооперації рівня прикладів повинен бути узгоджений з процесами побудови діаграми класів і діаграми послідовності;

– в першому випадку, необхідно стежити за використанням тільки тих об'єктів, для яких визначені породжують їх класи;

– у другому випадку потрібно узгоджувати послідовності переданих повідомлень;

– не допускається різний порядок проходження повідомлень для моделювання такою же самою взаємодією на діаграмі кооперації і діаграмі послідовності.

Таким чином, діаграма кооперації, з одного боку, забезпечує концептуально узгоджений перехід від статичної моделі діаграми класів до динамічних моделей поведінки, яку представляють діаграмами послідовності, станів і діяльності.

3. Зберегти файл з діаграмами послідовності та кооперації для подальших робіт.

Для виконання лабораторної роботи може бути використаний будь-який CASE-засіб UML діаграм (Rational Software, Visual Paradigm Community Edition, StarUML та інші).

4. Оформити звіт про лабораторну роботу

Звіт з лабораторної роботи повинен бути оформлений відповідно до вимог і складатися з наступних структурних елементів: титульний лист, текстова частина, додаток (діаграми послідовності та кооперації).

Текстова частина звіту повинна включати пункти: умови задачі, порядок виконання, словник термінів – короткі відомості про склад і компонентах побудованої моделі.

5. Представити звіт про лабораторну роботу для захисту

Захист звіту з лабораторної роботи полягає у надані викладачу отриманих результатів у вигляді файлу і демонстрації отриманих навичок при відповідях на контрольні питання викладача.

3.3 Контрольні запитання

1. До якої моделі проєктування відносяться діаграма послідовності та діаграма кооперації?
2. Яке призначення діаграми взаємодії?
3. Перелічить які основні типи діаграм використовуються в UML?
4. Яке призначення діаграми послідовності?
5. Перелічите основні елементи діаграми послідовності?

6. Що таке «лінія життя об'єкта», в чому її призначення?
7. Що таке «фокус управління», в чому його призначення?
8. Які існують типи повідомлень у діаграмі послідовності?
9. Яке призначення діаграми кооперації?
10. Перелічите основні елементи діаграми кооперації?
11. Що таке «активний об’єкт», в чому його призначення?
12. Що таке «складений об’єкт», в чому його призначення?
13. Що таке «мультіоб’єкт», в чому його призначення?
14. Які існують типи повідомлень у діаграмі кооперації?

Лабораторна робота №4.
РОЗРОБКА ДІАГРАМИ ДІЯЛЬНОСТІ В НОТАЦІЇ UML

Мета роботи: вивчення основних елементів діаграми діяльності та її створення, а також отримання навичок роботи з інструментальними засобами**.**

4.1 Теоретична частина

Моделювання динамічних аспектів систем за допомогою діаграм діяльності це моделювання послідовних (а іноді і паралельних) кроків обчислювального процесу.

Діаграми діяльності можуть використовуватися окремо для візуалізації, специфікації, конструювання та документування динаміки спільноти об'єктів або для моделювання потоку управління операцій. Якщо в діаграмах взаємодії акцент виконується на переходи потоку управління від одного об'єкта до іншого, то діаграми діяльності описують переходи потоку управління від одного кроку процесу до іншого. Діяльність – це структурований опис поточної поведінки. Здійснення діяльності розкривається у вигляді виконання окремих дій, кожна з яких може змінювати стан системи та надсилати повідомлення.

Основні елементи діаграми діяльності:

- початок процесу – позначає старт описуваного процесу, він може не збігатися з початком роботи програми або глобального процесу;
- дія – містить в собі опис операцій на поточному етапі виконання алгоритму;
- рішення – позначається ромбом, однак не містить в собі тексту. Текст умов розгалуження вказується на вихідних з рішення керуючих потоках;
- керуючий потік - вказує послідовність виконання дій;
- розділення – початок блоку незалежних операцій;
- з'єднання – завершення блоку незалежних операцій;
- завершення процесу – закінчення описуваного процесу, може не збігатися із закінченням роботи програми.

Дія може бути описана на природній мові, деякому псевдокоді або мовою програмування. Кожна діаграма діяльності повинна мати єдиний початковий і єдиний кінцевий стан.

Основними елементами діаграми діяльності є стани діяльності або стани дій. Стани діяльності й стани дій зображуються прямокутниками з округленими кутами. Стани дій відрізняються від станів діяльності тим, що вони не можуть

бути піддані декомпозиції. Стани дій з'єднуються лініями зі стрілками, які визначають напрямки переходів з одного стану дії в інший стан.

Прості послідовні потоки зустрічаються найчастіше, але це не єдиний засіб моделювання потоку управління. Так само, як в блок-схемі, можна включати в діаграму гілки, які специфікують альтернативні шляхи, які обираються на основі булевих виразів. Розгалуження може мати один вхідний потік і кілька вихідних.

Один з найбільш значущих недоліків звичайних блок-схем або структурних схем алгоритмів пов'язаний з проблемою зображення паралельних гілок окремих обчислень. Оскільки розпаралелювання обчислень істотно підвищує загальну швидкодію програмних систем, необхідні графічні позначення для представлення паралельних процесів. У мові UML для цієї мети використовується спеціальний символ для розділення і з'єднання паралельних обчислень або потоків управління.

В UML для опису розділення і з'єднання паралельних потоків управління застосовують лінійку синхронізації. Вона зображується у вигляді жирної горизонтальної або вертикальної лінії. Між точками розділення і з'єднання повинен підтримуватися баланс. Це означає, що число потоків, що виходять з точки розділення, має дорівнювати числу потоків, що приходять в відповідну точку з'єднання.

Діаграми діяльності відіграють важливу роль в розумінні процесів реалізації алгоритмів виконання операцій класів і потоків управління в системі, що моделюється. Застосування доріжок і об'єктів відкриває додаткові можливості для наочного представлення бізнес-процесів, дозволяючи специфікувати діяльність підрозділів компаній і фірм.

При моделюванні динамічних аспектів системи діаграми діяльності використовуються два засоби: моделювання потоку робіт і моделювання операції.

1. Моделювання потоку робіт. Увага зосереджена на тому, як виглядає діяльність з точки зору діючих осіб, які взаємодіють з системою. Потоки робіт знаходяться на периферії програмних систем і застосовуються для візуалізації, специфікації, конструювання та документування бізнес-процесів, які включає система. Приклад моделювання потоку робіт представлено на рисунку 4.1.

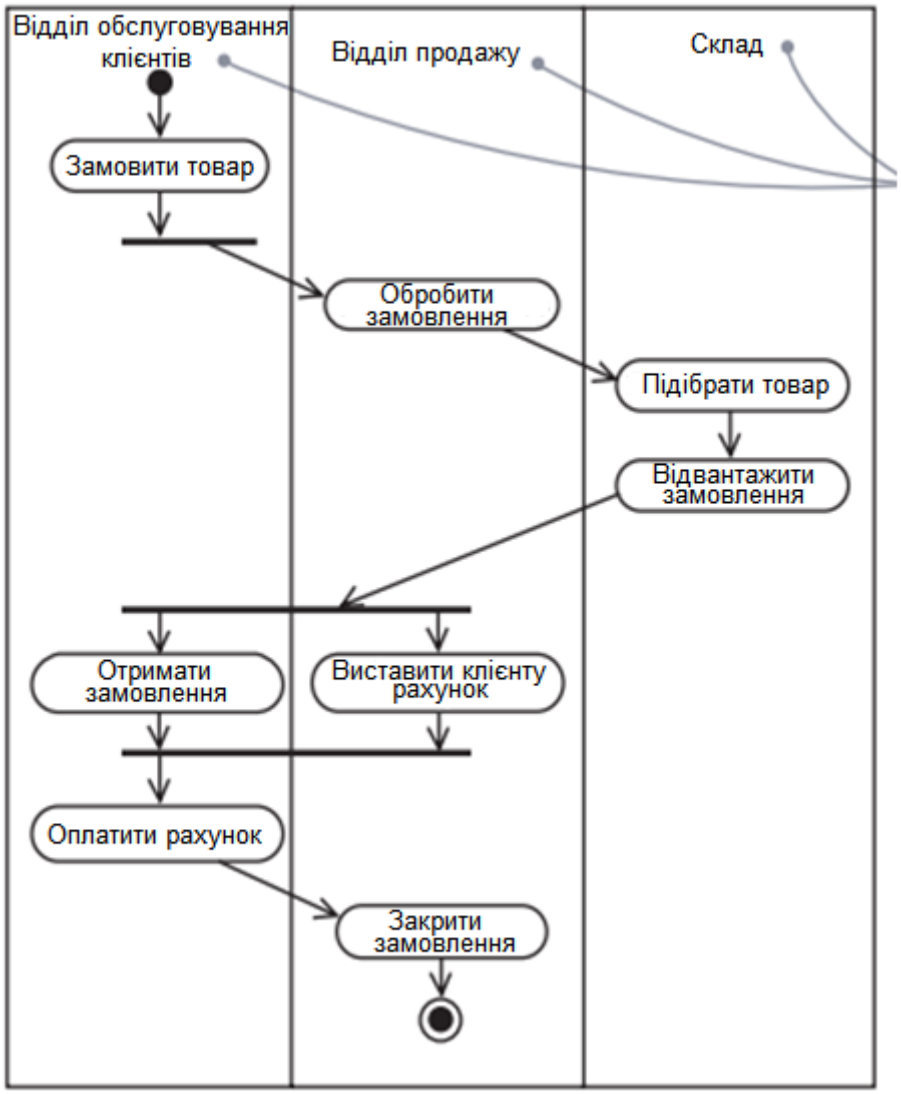

Рисунок 4.1 – Приклад моделювання потоку робіт

2. Моделювання операції. В цьому випадку діаграми діяльності виступають в якості схем, що моделюють етапи обчислювального процесу. При цьому особливо важливо моделювання розгалуження, розділення та з'єднання потоків управління. Контекст використовуваної діаграми діяльності включає параметри операцій та їх локальні об'єкти.

Діаграми діяльності можуть бути пов'язані з будь-яким модельованим елементом для візуалізації, специфікації, конструювання та документування його поведінки. Діаграма виступає просто як блок-схема дій, виконуваних цією операцією. Головна перевага діаграми діяльності в тому, що всі елементи, зазначені на ній, семантично зв'язуються в одну модель. Приклад моделювання операції представлено на рисунку 4.2.

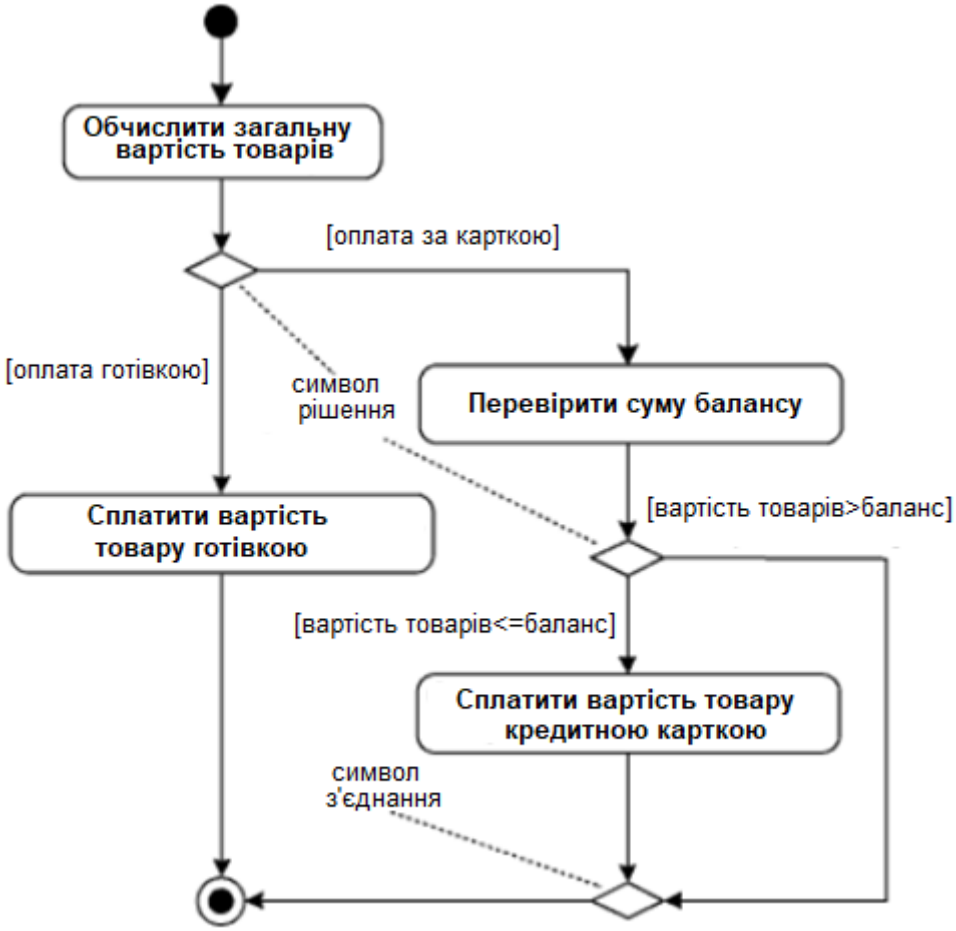

Рисунок 4.2 – Приклад моделювання операції

Діаграми діяльності важливі не тільки для моделювання динамічних аспектів системи, але і для конструювання інформаційних систем за допомогою прямого і зворотного проєктування.

4.2. Практична частина

За допомогою CASE-засобу побудувати діаграми діяльності для опису предметного середовища, яке задано відповідно до індивідуального варіанту завдання.

1. Для моделювання потоку робіт в діаграмі діяльності рекомендується:

– встановити фокус потоку робіт показати всі важливі потоки робіт системи в межах однієї діаграми;

– вибрати бізнес-об’єкти, які мають обов'язки найвищого рівня щодо всього потоку робіт або його частин. це можуть бути реальні сутності зі словника системи або більш абстрактні;

– створити «доріжку» для кожного важливого бізнес-об’єкта або підрозділу;

– ідентифікувати передумови початкового стану робочого потоку і постумови його кінцевого стану, це має значення для окреслення меж потоку робіт;

– починаючи зі стартового стану потоку робіт уточняти діяльності, що розгортаються в часі, і відобразити їх на діаграмі;

– складні дії, які застосовуються багаторазово, представити у вигляді викликів окремих діаграм діяльності;

– зобразити потоки, які з'єднуються з цими діями і вузлами діяльності (почати з послідовних потоків, потім розглянути розгалуження потім розділення і з'єднання);

– якщо існують важливі значення об'єктів, залучених до робочого потоку, відобразити їх на діаграмі та показати як змінюються їх значення і стан;

2 Для моделювання операцій в діаграмі діяльності рекомендується:

– зібрати разом всі абстракції, які беруть участь в ній;

– ідентифікувати передумови початкового стану операції і постумови її кінцевого стану, а також інваріанти класу, які повинні зберігатися протягом її виконання;

– з початку операції уточняти все діяльності та дії, що розгортаються в часі, і відобразити їх на діаграмі разом зі станами дій;

– при необхідності використовувати розгалуження для опису умовних шляхів та ітерацій;

– використовувати розділення і з'єднання для представлення паралельних потоків управління.

Структурована діаграма діяльності повинна мати наступні властивості:

– ім'я діаграми має відповідати її призначенням;

– починати з моделювання головного потоку;

– застосовувати розгалуження, паралельність і потік об'єктів на окремих діаграмах;

– фокусуватися на представленні лише одного аспекту динаміки системи;

– містити тільки ті елементи, які важливі для розуміння цього аспекту;

– представляти рівень деталізації, узгоджений з рівнем абстракції та показувати тільки ті доповнення, які важливі.

Зберегти файл з діаграмами діяльності для подальших робіт.

Для виконання лабораторної роботи може бути використаний будь-який CASE-засіб UML діаграм (Rational Software, Visual Paradigm Community Edition, StarUML та інші).

3. Оформити звіт про виконання лабораторної роботи.

Звіт повинен бути оформлений відповідно до вимог і складатися з наступних структурних елементів: титульний лист, текстова частина, додаток: розроблені діаграми діяльності.

Текстова частина звіту повинна містити:

- найменування і мету роботи;
- постановка завдання;
- діаграма діяльності бізнес-процесу та її опис;
- висновки по роботі.

4. Представити звіт про виконання лабораторної роботи для захисту.

Захист лабораторної роботи полягає у надані викладачу отриманих результатів у вигляді файлу і демонстрації отриманих навичок при відповідях на контрольні питання викладача.

4.3 Контрольні запитання

1. У яких випадках використовується діаграма діяльності?
2. Як позначаються стани дій на діаграмі діяльності?
3. Для чого використовуються стани дій на діаграмі діяльності?
4. Як позначаються переходи на діаграмі діяльності?
5. Для чого використовуються переходи на діаграмі діяльності?
6. Для чого використовуються доріжки на діаграмі діяльності?
7. Перелічите основні елементи діаграми діяльності?
8. Що таке «розгалуження», в чому його призначення?
9. Що таке «з'єднання», в чому його призначення?
10. Що таке «область розширення», в чому її призначення?
11. Які існують прийоми моделювання діаграми діяльності?
12. Перелічить вимоги до структурованої діаграми діяльності?

Лабораторна робота №5
РОЗРОБКА ДІАГРАМ СТАНІВ В НОТАЦІЇ UML

Мета роботи: вивчення основних елементів діаграми станів та її створення, а також отримання навичок роботи з інструментальними засобами.

5.1 Теоретична частина.

Для представлення динамічних особливостей взаємодії елементів моделі, в контексті реалізації варіантів використання, використовуються діаграми кооперації і послідовності. Однак для моделювання процесів функціонування більшості складних систем, особливо систем реального часу, цих уявлень недостатньо.

Діаграми станів найчастіше використовуються для опису поведінки окремих систем і підсистем. Вони також можуть бути застосовані для специфікації функціональності примірників окремих класів, тобто для моделювання всіх можливих змін станів конкретних об'єктів. Діаграма станів по суті є графом спеціального виду, який служить для представлення кінцевого автомату.

Діаграми станів можуть бути вкладені одна в одну, утворюючи вкладені діаграми для більш детального уявлення станів окремих елементів моделі. Для розуміння семантики конкретної діаграми станів необхідно представляти особливості поведінки модельованої сутності, а також мати загальні відомості з теорії кінцевих автоматів.

Кінцевий автомат – модель для специфікації поведінки об'єкта в формі послідовності його станів, які описують реакцію об'єкта на зовнішні події, виконання об'єктом дій, а також зміна його окремих властивостей.

Діаграма станів відображає кінцевий автомат у вигляді графу, вершинами якого є стани об’єкту, поведінка якого моделюється, а переходами – події, які переводять об’єкт, який розглядається, з одного стану в інший. При цьому вважається, що час перебування об’єкта в певному стані набагато більший за час, необхідний для переходу з одного стану в інший, тобто переходи між станами здійснюються миттєво. Опис станів дозволяє точно описати модель поведінки об'єкта при одержанні різних повідомлень і взаємодії з іншими об'єктами.

Стан – це ситуація в житті об’єкту, на протязі якої він задовольняє певній умові, здійснює певну діяльність або очікує якусь.

Графічно стан відображається у вигляді прямокутника з закругленими вершинами наведено на рисунку 5.1. Цей прямокутник, в свою чергу, може бути

розділений на дві секції горизонтальною лінією. Якщо вказана лише одна секція, то в ній записується тільки ім'я стану, як показано на рисунку 5.1а. В іншому випадку в першій з них записується ім'я стану, а в другій - список деяких внутрішніх дій або переходів в даному стані, як показано на рисунку 5.1б.

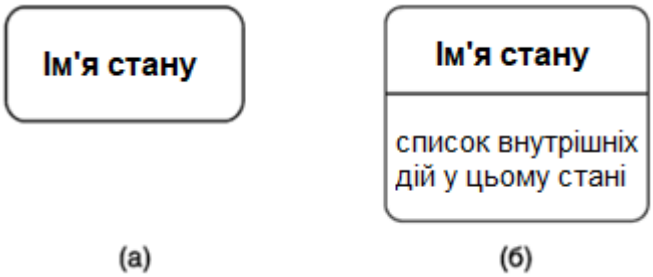

Рисунок 5.1 – Графічне зображення стану на діаграмі станів

Ім'я стану являє собою рядок тексту, який розкриває змістовний сенс або семантику даного стану. Ім'я повинно являти собою закінчену пропозицію і завжди записуватися з великої літери. Оскільки стан системи є частиною процесу її функціонування, рекомендується в якості імені використовувати дієслова в теперішньому часі або відповідні причастя.

Список внутрішніх дій містить перелік дій, які виконуються у процесі знаходження системи чи об’єкта в даному стані. Кожна дія відображається у форматі:

<період виконання>/<назва дії>,

де поле *<період виконання>* може набувати наступних значень: OnEntry – дія виконується під час того, як система входить у даний стан; OnExit – дія виконується при виході з даного стану; Do – дія виконується під час знаходження в даному стані; OnEvent – дія виконується при настанні певної (зовнішньої) події.

Подія – це специфікація суттєвого факту, який виникає у часі та просторі. В контексті автоматів подія – це стимул, здатний викликати спрацювання переходу.

Діяльність – це продовження неатомарного обчислення в середині автомату.

Дія – це атомарне обчислення, яке приводить до заміни стану або поверненню значення.

Перехід – це відношення між двома станами, яке показує, що об’єкт, який знаходиться у першому стані, повинний виконати деяку подію і перейти у другий стан, як тільки виникне певна подія і будуть виконані задані умови.

В мові UML події грають роль стимулів, які ініціюють переходи з одних станів в інші. Події можна розглядати як: сигнали, виклики, закінчення фіксованих проміжків часу або моменти закінчення виконання певних дій. Залежно від виду подій-стимулів в мові UML розрізняють два типи переходів: тригерні і нетригерні.

Перехід називається тригерним, якщо його подія-тригер пов’язана із зовнішніми умовами. Перехід називається нетригерним, якщо він відбувається по завершенні виконання діяльності в даному стані. Нетригерні переходи часто називають переходами по завершенні діяльності. Для них поруч зі стрілкою переходу не вказується ніякого імені події, а в початковому стані повинна бути описана внутрішня діяльність, після закінчення якої відбудеться той чи інший нетригерний перехід.

Зображений фрагмент діаграми станів на рисунку 5.2, моделює зміну станів банкомату при перевірці ПІН-коду.

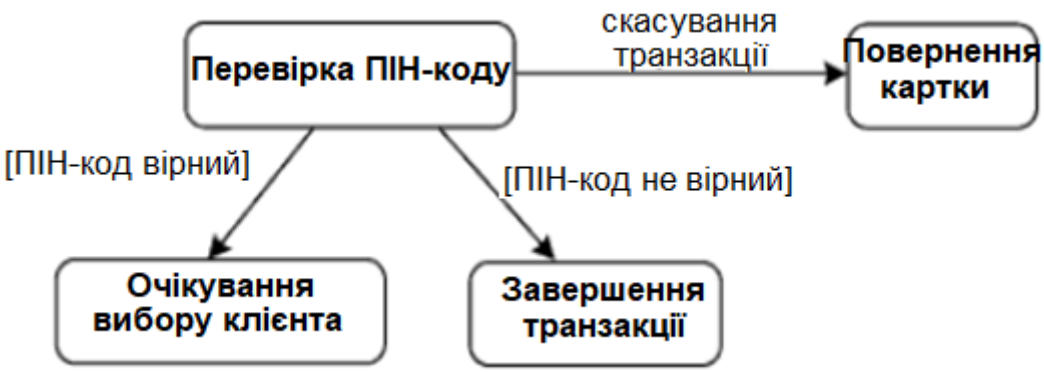

Рисунок 5.2 – Тригерні та нетригерні переходи на діаграмі станів

Нетригерні переходи на даній діаграмі позначені сторожовими умовами, які виключають конфлікт між ними. Сторожова умова – логічна умова, записана в прямих дужках і представляє собою булевий вираз. При цьому булевий вираз має приймати один з двох значень: "true" або "false". З контексту діаграми станів повинна явно слідувати семантика цього виразу, а для запису виразу може використовуватися звичайна мова, псевдокод або мова програмування.

Що стосується тригерного переходу, позначеного подією скасування транзакції, то він відбувається незалежно від перевірки ПІН-коду в тому випадку, коли клієнт вирішив відмовитися від введення ПІН-коду.

Формалізм кінцевих автоматів допускає вкладення одних кінцевих автоматів в інші для уточнення внутрішньої структури окремих більш загальних станів. В цьому випадку вкладені кінцеві автомати отримали назву кінцевих

підавтоматів. Підавтомати можуть використовуватися для внутрішньої специфікації процедур і функцій, реалізація яких обумовлює поведінку модельованої системи або об'єкта.

Моделювання складних об'єктів і систем, як правило, пов'язана з багаторівневим поданням їх станів. У цьому випадку виникає необхідність деталізувати окремі стани, зробивши їх складовими.

Складений стан – стан, який складається з інших вкладених в нього станів. Складений стан називають також станом-композитом. Вкладені стани виступають по відношенню до складеного стану як підстани.

Складений стан може містити або кілька послідовних підстанів, або кілька паралельних кінцевих підавтоматів. Кожен стан-композит може уточнюватися тільки одним із зазначених засобів. При цьому будь-яка з підстанів, в свою чергу, може бути станом-композитом і містити в собі інші вкладені підстани. Кількість рівнів вкладеності складових станів в мові UML не фіксоване.

Послідовні підстани – вкладені стани стану-композиту, в рамках якого в кожен момент часу об'єкт може знаходитися в одному і тільки одному підстанів. Поведінка об'єкта в цьому випадку представляє собою послідовну зміну підстанів, від початкового до кінцевого.

Графічне представлення складеного стану з вкладеними в нього послідовними підстанами показано на рисунку 5.3. У цьому випадку розміри графічного символу складеного стану збільшуються, так щоб вмістити в себе всі підстани.

Рисунок 5.3 – Графічне представлення складеного стану послідовними підстанами

Паралельні підстани (concurrent substates) - вкладені стану, що використовуються для специфікації двох і більше кінцевих підавтомат, які можуть виконуватися паралельно всередині складеного стану.

Кожен з кінцевих підавтомат займає деяку графічну область всередині складеного стану, яка відділяється від решти горизонтальною пунктирною лінією. Якщо на діаграмі станів є складений стан з вкладеними паралельними підстанами, то об'єкт може одночасно перебувати в кожному з цих підстанів.

Графічне представлення складеного стану з паралельними підстанами наведено на рисунку 5.4.

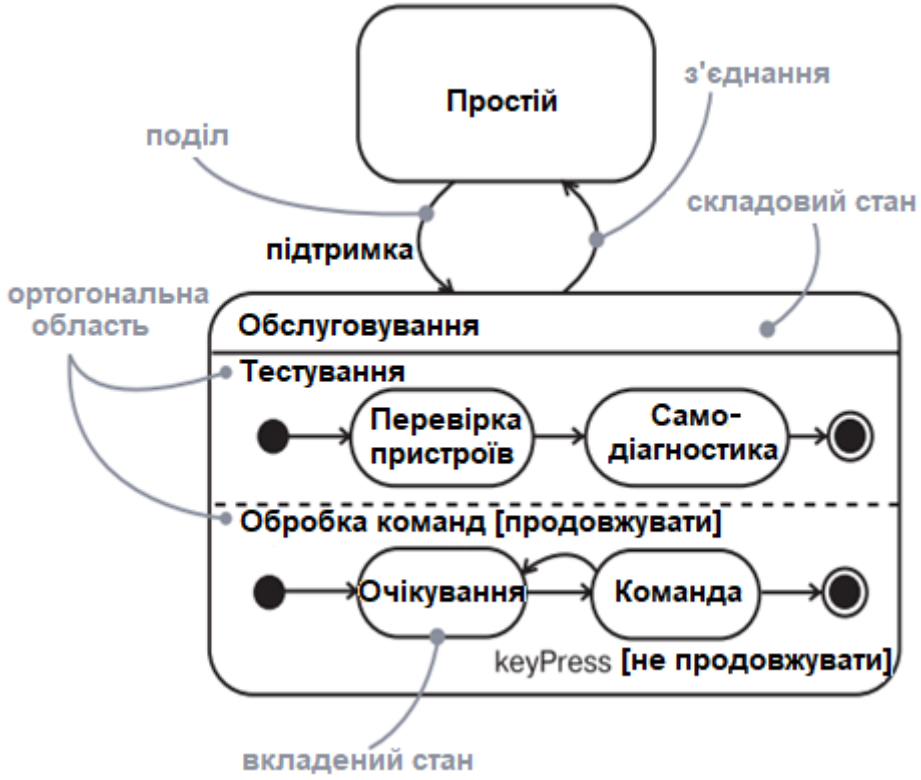

Рисунок 5.4 – Графічне представлення складеного стану з паралельними підстанами

Звичайний кінцевий автомат не дозволяє враховувати передісторію в процесі моделювання поведінки систем і об'єктів. Однак функціонування ряду систем засновано на можливості виходу з окремого стану-композиту з подальшим поверненням в цей же стан. Може виявитися необхідним врахувати ту частину діяльності, яка була виконана на момент виходу з цього стану-композиту, щоб не починати її виконання спочатку. Для цієї мети в мові UML існує історичний стан.

Графічне зображення недавнього (а) і давнього (б) історичного стану наведено на рисунку 5.5.

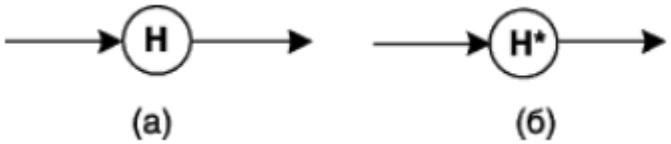

Рисунок 5.5 – Графічне зображення недавнього (а) і давнього (б) історичного стану

Історичний стан дозволяє складеному стану, що включає підстани «пам'ятати» підстан, який був активним на момент останнього переходу з складеного стану зовні. Приклад зображення історичного стану наведено на рисунку 5.6.

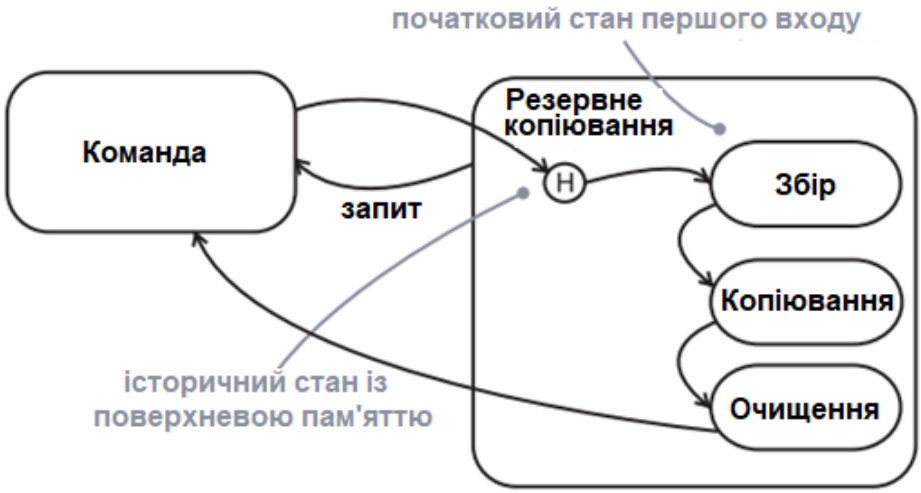

Рисунок 5.6 – Приклад зображення історичного стану

При розробці діаграми станів потрібно постійно стежити, щоб об'єкт в кожен момент може перебувати тільки в єдиному стані. При виділенні станів і переходів слід пам'ятати, що тривалість спрацьовування окремих переходів повинна бути істотно меншою, ніж знаходження об'єкта, що моделюється у відповідних станах.

Структурована діаграма станів має наступні характеристики:

– діаграма станів зосереджена на передачі одного аспекту динаміки системи містить тільки ті елементи, які істотні для розуміння цього аспекту;

– діаграма станів деталізована адекватно своїм рівнем абстракції;

– діаграма станів має ім'я, що відображає її призначення;

– діаграма станів показує розгалуження, паралелізм і потоки об'єктів на окремих діаграмах;

– у діаграмі станів елементи розміщуються так, щоб звести до мінімуму перетинання ліній;

– при створенні складних діаграм станів потрібно застосовувати розширені засоби;

– у діаграмі станів необхідно проводить обов'язкову перевірку, щоб ніякі два переходи з одного стану не могли спрацювати одночасно;

– у діаграмі станів використовувати історичні стани в тому випадку, коли необхідно організувати обробку виняткових ситуацій (переривань) без втрати даних або виконаної роботи.

5.2. Практична частина

За допомогою CASE-засобу побудувати діаграми станів для опису предметного середовища, яке задано відповідно до індивідуального варіанту завдання.

1. Створити одну діаграму станів для опису процесу функціонування обраної системи в цілому та дві діаграми для відповідних елементів системи.

Використовувати діаграму станів для авторизації користувачів забороняється.

Вимоги:

– кожна діаграма повинна містіть не менше 6 станів;

– для кожного переходу визначити хоча б одну з характеристик (тригер, сторожова умова, дія).

2. Для моделювання діаграм станів рекомендується:

– вибрати контекст автомата – клас та-або варіант використання;

– встановити початковий та кінцевий стани об'єкта;

– прийняти рішення щодо стабільних станів об'єкта, розглянувши умови, в яких об'єкт може існувати протягом певного періоду часу;

– почати з станів вищого рівня і тільки після цього перейти до розгляду можливих підстанів;

– прийняти рішення щодо упорядкування станів протягом часу життя об'єкта;

– прийняти рішення щодо подій, які можуть викликати переходи з одного стану в інший;

– змоделювати події як тригери переходів між станами;

– приєднати дії до переходів і/або до станів;

– розглянути можливості спрощення автомата за рахунок застосування підстанів, розгалуження, розділень, з'єднань та історичних станів;

– перевірити, чи немає яких-небудь «мертвих» станів, з яких об'єкт не зможе вийти ні при якому поєднанні подій;

– перевірити діаграми станів на відповідність характеристикам структурованості.

Зберегти файл з діаграмами станів для подальших робіт.

Для виконання лабораторної роботи може бути використаний будь-який CASE-засіб UML діаграм (Rational Software, Visual Paradigm Community Edition, StarUML та інші).

3 Оформити звіт про виконання лабораторної роботи

Звіт з лабораторної роботи повинен бути оформлений відповідно до вимог і складатися з наступних структурних елементів: титульний лист, текстова частина.

Текстова частина звіту повинна містити: найменування і мету роботи, постановка завдання, діаграми станів об’єктів та їх опис, висновки по роботі.

4. Представити звіт про лабораторну роботу для захисту

Захист звіту з лабораторної роботи полягає у надані викладачу отриманих результатів у вигляді файлу і демонстрації отриманих навичок при відповідях на контрольні питання викладача.

5.3 Контрольні запитання

1. Яке призначення діаграми станів?
2. Яке призначення кінцевий автомат?
3. Що таке подія та діяльність?
4. Які бувають переходи між станами?
5. Що таке сторожова умова?
6. Які специфікації можна задати для переходів між станами?
7. Що таке складний стан та як його задати?
8. Що таке історія стану?
9. Які ознаки послідовного підстану?
10. Які ознаки паралельного підстану?
11. Які характеристики має структурована діаграма станів?
12. Перелічить ознаки складних переходів та псевдосостояній?

Лабораторна робота №6.
РОЗРОБКА ДІАГРАМ КОМПОНЕНТІВ ТА РОЗГОРТАННЯ В НОТАЦІЇ UML

Мета роботи: формувати знання і навички щодо призначення і особливостей використання діаграм компонентів і розгортання; навчитися їх будувати і застосовувати в процесі проєктування архітектури інформаційної системи.

6.1 Теоретична частина.

Усі розглянуті раніше діаграми відображали концептуальні і логічні аспекти побудови моделі системи. Особливість логічного представлення полягає в тому, що воно оперує поняттями, які не мають матеріального втілення. Тобто, різні елементи логічного представлення, такі як класи, асоціації, стани, повідомлення, не існують матеріально або фізично. Вони лише відображають розуміння статичної структури тієї або іншої системи або динамічні аспекти її поведінки.

Для створення конкретної фізичної системи необхідно реалізувати усі елементи логічного представлення в конкретній матеріальній сутності. Для опису таких реальних сутностей призначений інший аспект модельного представлення, а саме – фізичне представлення моделі. В контексті мови UML це означає сукупність пов’язаних фізичних сутностей, включаючи програмне та апаратне забезпечення, а також персонал, який організовано для виконання спеціальних завдань.

В мові UML для фізичного представлення моделей систем використовуються діаграми реалізації, які включають дві окремі діаграми: діаграму компонентів і діаграму розгортання.

Діаграма компонентів описує особливості фізичного представлення системи. Діаграма компонентів дозволяє визначити архітектуру системи, що розробляється, встановивши залежності між програмними компонентами.

Діаграма компонентів забезпечує узгоджений перехід від логічного представлення до конкретної реалізації проекту у формі програмного коду. Одні компоненти можуть існувати тільки на етапі компіляції програмного коду, інші - на етапі його виконання. Діаграма компонентів відображає загальні залежності між компонентами, розглядаючи останні в якості стосунків між ними.

Компонент – фізично існуюча частина системи, яка забезпечує реалізацію класів, взаємовідносин і функціональної поведінки модельованої програмної системи. На рисунках 6.1а,б показано графічне зображення компонента.

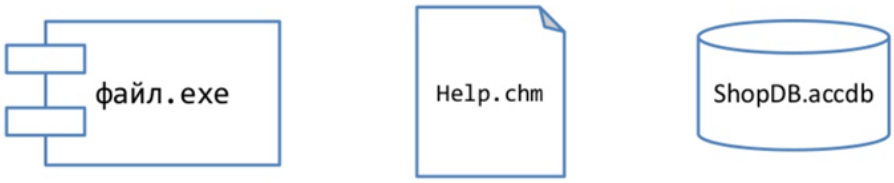

Рисунок 6.1 – Графічне зображення компонента

Компонент призначений для представлення фізичної організації асоційованих із ним елементів моделі. Додатково компонент може мати текстовий стереотип і помічені значення, а деякі компоненти - власне графічне представлення. Компонентом може бути програмний код окремого модуля, командні файли або файли, що містять скрипти, що інтерпретуються.

Для більш наочного зображення компонентів були запропоновані графічні стереотипи. Стереотипи для компонентів розгортання, які забезпечують безпосереднє виконання системою своїх функцій. Такими компонентами можуть бути спільні бібліотеки, які представлено на рисунку 6.2а, Web-сторінки на мові розмітки гіпертексту представлені на рисунку 6.2б і файли довідки, які показані на рисунку 6.2в. Стереотипи для компонентів у формі робочих продуктів. Як правило - це файли з вихідними текстами програм, які показано на рисунку 6.2г.

Ці елементи іноді називають артефактами, підкреслюючи при цьому їх закінчений інформаційний зміст, залежний від конкретної технології реалізації відповідних компонентів.

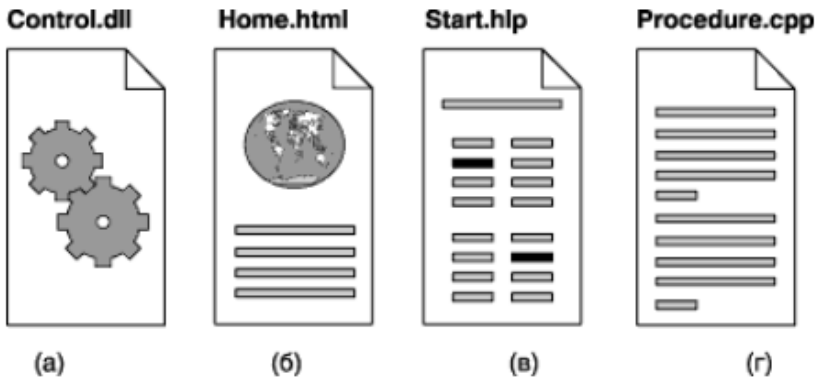

Рисунок 6.2 – Варіанти графічного зображення компонентів на діаграмі компонентів

Наступним графічним елементом діаграми компонентів є інтерфейс. У загальному випадку інтерфейс графічно зображується колом, яке з'єднується з компонентом відрізком лінії без стрілок, приклад наведено на рисунку 6.3а.

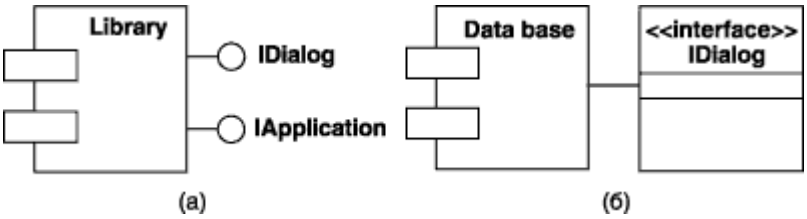

Рисунок 6.3 – Графічне зображення інтерфейсів на діаграмі компонентів

При цьому ім'я інтерфейсу, яке рекомендується починати з великої літери "I", записується поряд з колом. Семантично лінія означає реалізацію інтерфейсу, а наявність інтерфейсів у компонента означає, що даний компонент реалізує відповідний набір інтерфейсів. Крім того, інтерфейс на діаграмі компонентів може бути зображений у вигляді прямокутника класу зі стереотипом <<interface>> і секцією підтримуваних операцій, приклад наведено на рисунку 6.3б. Як правило, цей варіант позначення використовується для представлення внутрішньої структури інтерфейсу.

Розрізняють два способи зв'язку інтерфейсу і компонента. Якщо компонент реалізує деякий інтерфейс, то такий інтерфейс називають експортним або підтримуваним, оскільки цей компонент надає його в якості сервісу інших компонентів. Якщо ж компонент використовує певний інтерфейс, який реалізується іншим компонентом, то такий інтерфейс для першого компонента називається імпортованим. Особливість імпортованого інтерфейсу полягає в тому, що на діаграмі компонентів це відношення зображається за допомогою залежності.

Порт – це точка взаємодії між класифікатором та зовнішнім середовищем. Він групує семантично зв'язний набір послуг і необхідних інтерфейсів. Порт може використовуватися в UML без вказівки імені порту. Коли порт відображається за кордоном класифікатора, це означає, що порт є відкритим. Це також означає, що всі використовувані інтерфейси є загальнодоступними. Коли порт відображається всередині класифікатора, він або захищений, або закритий. Приклад зображення порту наведено на рисунку 6.4.

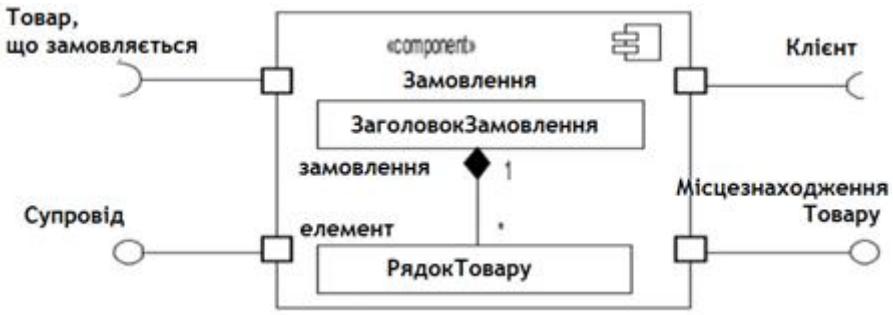

Рисунок 6.4 – Графічне зображення порту

Ставлення залежності на діаграмі компонентів зображується пунктирною лінією зі стрілкою, спрямованою від клієнта або залежного елемента до джерела або незалежного елементу моделі.

Залежності можуть відображати зв'язки окремих файлів програмної системи на етапі компіляції і генерації програмного коду. В інших випадках залежність може вказувати на наявність в незалежному компоненті описів класів, які використовуються в залежному компоненті для створення відповідних об'єктів. Стосовно діаграми компонентів залежно можуть пов'язувати компоненти і імпортовані цим компонентом інтерфейси, а також різні види компонентів між собою.

Так, наприклад, зображений на рисунку 6.5, фрагмент діаграми компонентів являє інформацію про те, що компонент з ім'ям Control залежить від імпортованого інтерфейсу IDialog, який, в свою чергу, реалізується компонентом з ім'ям DataBase. При цьому для другого компонента цей інтерфейс є експортним. Відображає зв'язок другого компоненту DataBase з цим інтерфейсом в формі залежності не можна, оскільки цей компонент реалізує вказаний інтерфейс.

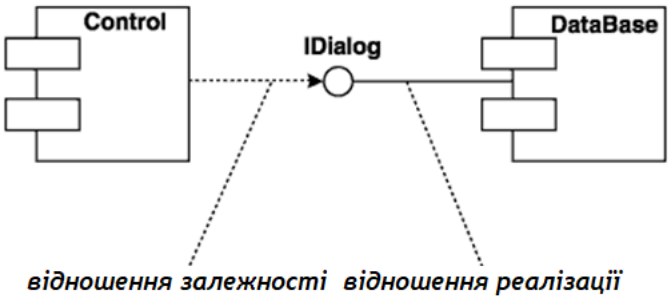

Рисунок 6.5 – Фрагмент діаграми компонентів з відносинами залежності і реалізації

Діаграма компонентів, як правило, розробляється спільно з діаграмою розгортання, на якій подається інформація про фізичну розміщенні компонентів програмної системи по її окремих вузлах.

Діаграма розгортання (розташування) –- діаграма, на якій представлені вузли виконання програмних компонентів реального часу, процесів і об'єктів. Графічно вузол на діаграмі розгортання зображується у формі тривимірного куба. Вузол має ім'я, яке вказується всередині цього графічного символу. Самі вузли можуть представлятися як на рівні типу, так і на рівні примірника. Обидва варіанти показано на рисунку 6.6.

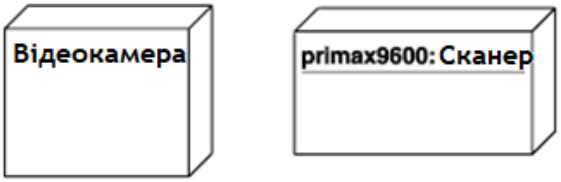

Рисунок 6.6 – Графічне зображення вузла

Діаграма розгортання призначена для візуалізації елементів і компонентів програми, існуючих тільки на етапі її виконання. При цьому представляються тільки ті компоненти програми, які є виконуваними файлами або динамічними бібліотеками. Компоненти, що не використовуються на етапі виконання, на діаграмі розгортання не зображуються. Так, компоненти з початковими текстами програм можуть бути присутніми тільки на діаграмі компонентів. На діаграмі розгортання вони не вказуються.

Зображення вузлів можуть розширюватися, щоб включити додаткову інформацію про специфікації вузла. Якщо додаткова інформація відноситься до імені вузла, то вона записується під цим ім'ям в формі поміченого значення, приклад зображеній на рисунку 6.7.

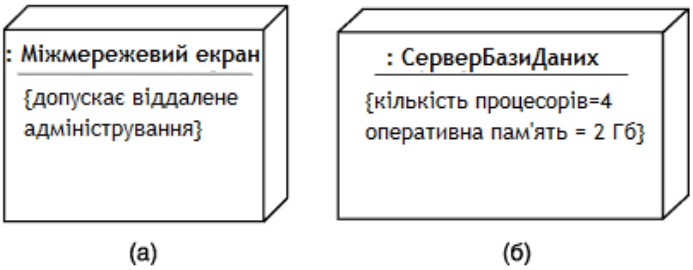

Рисунок 6.7 – Графічне зображення вузла-екземпляра з додатковою інформацією в формі поміченого значення

Діаграма розгортання містить графічні зображення процесорів, пристроїв, процесів та зав’язків між ними. На відміну від діаграм логічного представлення, діаграма розгортання є єдиною для системи в цілому, оскільки вона відображає всі особливості її реалізації. Діаграма розгортання розробляється спільно системними аналітиками, мережевими інженерами и системотехніками.

На діаграмі розгортання крім зображення вузлів вказуються відносини між ними. Як відносин виступають фізичні з'єднання між вузлами, а також залежності між вузлами і компонентами, які допускається зображати на діаграмах розгортання. З'єднання є різновидом асоціації і зображуються відрізками ліній без стрілок. На рисунку 6.8 представлено фрагмент діаграми розгортання та визначені рекомендації по технології фізичної реалізації з'єднань в формі примітки.

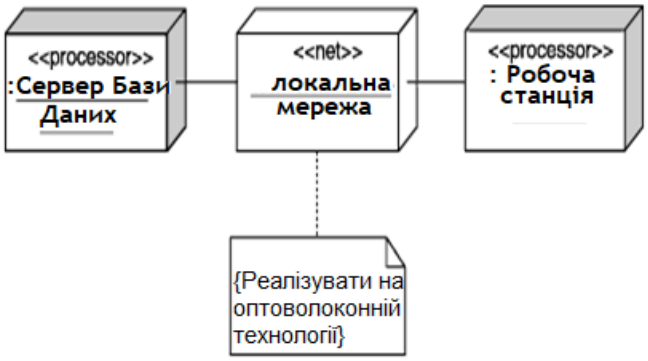

Рисунок 6.8 - Фрагмент діаграми розгортання з сполуками між вузлами

Крім з'єднань на діаграмі розгортання можуть бути присутніми відносини залежності між вузлом і розміщеними на ньому компонентами. Діаграма розгортання з відношенням залежності між вузлом і розгорнутими на ньому компонентами показана на рисунку 6.9.

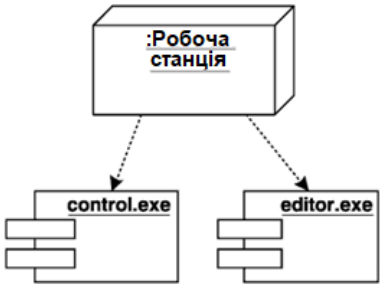

Рисунок 6.9 – Діаграма розгортання з відношенням залежності між вузлом і розгорнутими на ньому компонентами

Діаграма розгортання застосовується для представлення загальної конфігурації і топології розподіленої програмної системи і містить зображення розташування компонентів на окремих вузлах системи. Крім того, діаграма розгортання показує наявність фізичних з'єднань –- маршрутів передачі інформації між апаратними пристроями та задіяними в реалізації системи.

6.2 Практична частина

За допомогою CASE-засобу побудувати діаграми компонентів та розгортання для опису предметного середовища, яке задано відповідно до індивідуального варіанту завдання.

1. Для моделювання діаграми компонентів рекомендується:

– обрати обчислювальні платформи і операційні системи, на яких передбачається реалізовувати систему, а також обрати відповідні бази даних і мови програмування;

– після цього можна приступати до загальної структуризації діаграми компонентів;

– необхідно вирішити з яких фізичних частин (файлів) буде складатися програмна система;

– програмна система повинна відображати раціональне використання нею обчислювальних ресурсів, для цієї мети необхідно більшу частину описів класів, їх операцій і методів винести в динамічні бібліотеки, залишивши в виконуваних компонентах тільки найнеобхідніші для ініціалізації програми фрагменти програмного коду;

– після загальної структуризації фізичного представлення системи необхідно доповнити модель інтерфейсами, при розробці інтерфейсів слід звертати увагу на узгодження (стикування) різних частин програмної системи;

– встановити та нанести на діаграму компонентів взаємозв'язки між компонентами, а також відносин реалізації.

– відносини повинні відображати всі найважливіші аспекти фізичної реалізації системи, починаючи з особливостей компіляції початкового програмного коду і закінчуючи виконанням окремих частин програми на етапі її виконання, для цього використовувати різні види графічного зображення компонентів;

– при розробці діаграми компонентів слід дотримуватися загальних принципів створення моделей на мові UM, необхідно використовувати вже наявні в мові UML компоненти і стереотипи;

– використовувати додаткові стереотипи для окремих нетипових компонентів або помічені значення для уточнення їх окремих характеристик.

2. Для побудови діаграми розгортання рекомендується:

– здійснити ідентифікацію всіх апаратних, механічних та інших типів пристроїв, які необхідні для виконання системою всіх функцій;

– визначити структуру програмного комплексу системи (файли, бібліотеки, драйвери та ін.);

– визначити обчислювальні вузли системи, що володіють процесором і пам'яттю, при цьому використовувати стереотипи;

– визначити топологію та технологію реалізації комунікацій в системі;

– визначити типові вузли розгортання системи та з‘єднання між ними.

Зберегти файл з діаграмами компонентів та розгортання для подальших робіт.

Для виконання лабораторної роботи може бути використаний будь-який CASE-засіб UML діаграм (Rational Software, Visual Paradigm Community Edition, StarUML та інші).

4 Оформити звіт про лабораторну роботу.

Звіт з лабораторної роботи повинен бути оформлений відповідно до вимог і складатися з наступних структурних елементів: титульний лист, текстова частина, додаток: розроблені діаграми компонентів та розгортання.

Текстова частина звіту повинна включати пункти:

– умови задачі;

– порядок виконання;

– словник термінів – короткі відомості про склад і компонентах побудованої моделі.

5. Представити звіт про лабораторну роботу для захисту.

Захист звіту з лабораторної роботи полягає у надані викладачу отриманих результатів у вигляді файлу і демонстрації отриманих навичок при відповідях на контрольні питання викладача.

6.3 Контрольні запитання

1. Які складові містить повний проєкт програмної системи?
2. Які особливості логічного і фізичного представлення системи?
3. Що називається фізичною системою?
4. Що таке компонент? Для чого він призначений?
5. Яку проблему проєктування покликані розв’язати діаграми компонентів?

6. Які види елементів моделі представлені на діаграмі компонентів?
7. Які компоненти відповідають файлам коду? Назвіть ці відповідності.
8. У чому відмінність компонента від класу?
9. Як на діаграмі зображується компонент в UML?
10. Які стереотипи зумовлені в мові UML?
11. Що таке інтерфейс?
12. Як компоненти зв'язуються через інтерфейси?
13. Як на діаграмі відображається реалізація інтерфейсу компонентом?
14. Як на діаграмі відображається підключення компонента до інтерфейсу?
15. Навіщо потрібні порти?
16. Що показує діаграма розгортання (розташування)?
17. Які сутності відображуються на діаграмах розташування?
18. У яких випадках потрібно застосувати діаграми розташування?

СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ

1. Вислоух С.П. Комп’ютерне моделювання процесів та систем. Чисельні методи : підручник / С.П. Вислоух, О.В. Волошко, Г.С. Тимчик, М.В. Філіппова. – Київ : КПІ ім. Ігоря Сікорського, Вид-во «Політехніка», 2021. – 228 с. https://kafvp.kpi.ua/book/komp-iuterne-modeliuvannia-protsesiv-ta-system-chyselni-metody/
2. Marlon Dumas, Marcello La Rosa, et. all. Fundamentals of Business Process Management. Springer; 3th edition. 2018. - 399 p. https://link.springer.com/book/10.1007/978-3-662-56509-4
3. Квітка О. О. Комп’ютерне моделювання процесів і систем. Практичні заняття : навч. посіб. / О. О. Квітка; КПІ ім. Ігоря Сікорського, 2023. - 83 с https://ela.kpi.ua/bitstream/123456789/57251/1/NP_KompModProtsSyst_Prakt_2023_IKhF.pdf
4. Квітка О.О. Комп'ютерне моделювання процесів і систем: Організація розрахунків у середовищі MathCAD: навчальний посібник для самостійної роботи студентів спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» / КПІ ім. Ігоря Сікорського: уклад.:, А.М. Шахновський. – К.: КПІ ім. Ігоря Сікорського, 2022. – 89 с. https://ela.kpi.ua/bitstream/123456789/57252/1/NavchPosibn_KompModProtsSyst_SR_2022_IKhF.pdf
5. Кузяєв І.М. Основи комп'ютерного моделювання технічних систем : навчальний посібник / І.М. Кузяєв, В.І. Ситар. - Дніпро : ДВНЗ УДХТУ, 2020. - 392 с. https://discovery.kpi.ua/Record/000638965/Details
6. Обод І.І. Математичне моделювання систем : навчальний посібник для студентів спеціальностей "Комп'ютерна інженерія", "Комп'ютерні науки та інформаційні технології" / І.І. Обод, Г.Е. Заволодько, І.В. - Харків : Друкарня Мадрид, 2019. - 267 с. https://repository.kpi.kharkov.ua/server/api/core/bitstreams/b74155a0-800d-4968-a15b-54ebc07d498e/content
7. Jacek Makinia, Ewa Zaborowska. Mathematical Modelling and Computer Simulation of Activated Sludge Systems - Second Edition, The International water association, 2020. –670 p. https://books.google.com.ua/books/about/Mathematical_Modelling_and_Computer_Simu.html?id=pAT2DwAAQBAJ&redir_esc=y
8. Соколовський Я.І. Моделювання систем у GPSS WORLD : навчальний посібник / Я.І. Соколовський, Ю.В. Шабатура, Я.І. Виклюк, І.М. Крошний, М.В. Дендюк. - Львів : Видавництво "Новий Світ-2000", 2021. - 288 с. https://discovery.kpi.ua/Record/000636588
9. Жученко А. І. Математичне моделювання процесів і систем [Електронний ресурс] : Навч. посіб. / А. І. Жученко, Л. Р. Ладієва, М. С. Піргач, Я. Ю. Жураковський. – Київ : КПІ ім. Ігоря Сікорського, 2021. – 351 с. https://ooep.kpi.ua/downloads/disc/kmod/lab_km21.pdf

10. Дичка І. А. Математичне моделювання систем і процесів: комп’ютерний практикум : навч. посіб. для студ. спеціальності 121 «Інженерія програмного забезпечення», освітньої програми «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем» / І. А. Дичка, М.В. Онай, Т.М. Заболотня; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 2.49 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2023. – 128 с. https://ela.kpi.ua/bitstream/123456789/57239/1/Mathematical_Processes.pdf
11. Гаркуша І.М. Конспект лекцій з дисципліни “Проектування інформаційних систем” для студентів галузі знань 12 “Інформаційні технології” спеціальності 126 “Інформаційні системи та технології”. – Д.: НТУ «ДП», 2020. – 75 с.
12. Савеленко О.К., Лисенко І.А., Іванченко О.О. С12 CASE-технології у проектуванні інформаційних систем: Навчальний посібник. - Кропивницький: Видавець Лисенко В.Ф., 2018.- 240 с https://dspace.kntu.kr.ua/server/api/core/bitstreams/068475bf-8c5d-42dd-9fdf-b0f4aad5b062/content

Навчальне видання

Методичні вказівки

до виконання лабораторних робіт з дисципліни
«Комп'ютерне моделювання процесів та систем» (Частина 1)
для здобувачів вищої освіти спеціальності
F3 «Комп’ютерні науки»
першого (бакалаврського) рівня

денної та заочної форми навчання

Укладачі: ГРИНЧЕНКО Марина Анатоліївна
РОГОВИЙ Антон Іванович

Рецензент А.М. Копп
Відповідальна за випуск І.В. Шуба
Роботу до видання рекомендував І.П. Гамаюн

В авторській редакції

План 2025 р., поз. 609

Підп.до друку 2025 р. Формат 60х84 1/16. Папір офсетний.
Друк- різографія. Гарнітура Таймс. Ум. друк. арк.
Наклад 100 прим. Зам. № Ціна договірна

Видавничий центр НТУ “ХПІ”.
Свідоцтво про державну реєстрацію ДК № 3657 від 24.12.2009 р.
61002, Харків, вул. Кирпичова, 2

Друкарня НТУ “ХПІ”. 61002, Харків, вул. Кирпичова, 2