

Лекція 4

Функціональне моделювання

Зміна фокусу: від "Що?" до "Хто?"

У попередній лекції (DFD) ми дивилися на систему як інженери-сантехніки: нас цікавило, куди течуть дані. Ми ігнорували те, навіщо вони туди течуть і хто натиснув кнопку.

Тепер ми змінюємо оптику. Функціональне моделювання (Use Case Modeling) — це погляд на систему очима користувача. Перше питання, яке ми ставимо: "Хто буде користуватися нашою системою?". Цих "когось" в UML називають Акторами (Actors).





Поняття Актора (Actor)

Актор — це зовнішня сутність (людина або система), яка взаємодіє з вашим програмним забезпеченням.

Актор — це Роль

Не посада чи ім'я. Один співробітник може грати роль "Менеджера" вдень і роль "Клієнта" ввечері. Актор — це "капельюх", який надягає людина.

Актор завжди ЗОВНІ

Знаходиться за межами кордонів системи. Ми не пишемо код для актора, ми пишемо інтерфейс для нього.

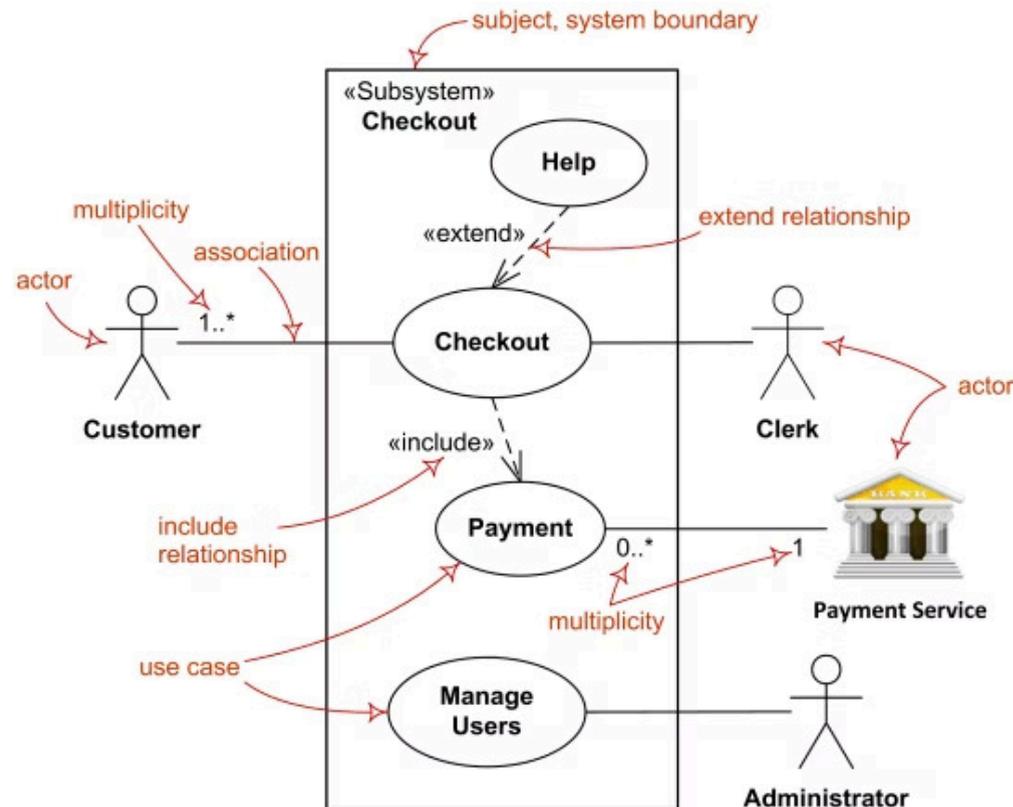
Класифікація Акторів

За природою

- **Люди:** Користувачі з клавіатурою/телефоном (Клієнт, Адмін)
- **Системи:** Інші програми через API (LiqPay, Nova Poshta API)
- **Час:** Таймер або Scheduler для автоматичних процесів

За важливістю

- **Первинний актор:** Починає діалог, його мету задовольняє Use Case (малюється зліва)
- **Вторинний актор:** Система звертається до нього за допомогою (малюється справа)



Успадкування ролей (Generalization)

Актори можуть мати ієрархію, схожу на успадкування класів у програмуванні. Це дозволяє не дублювати стрілки на діаграмах.

01

Створюємо актора "Пасажир"

Базова роль з мінімальними можливостями

02

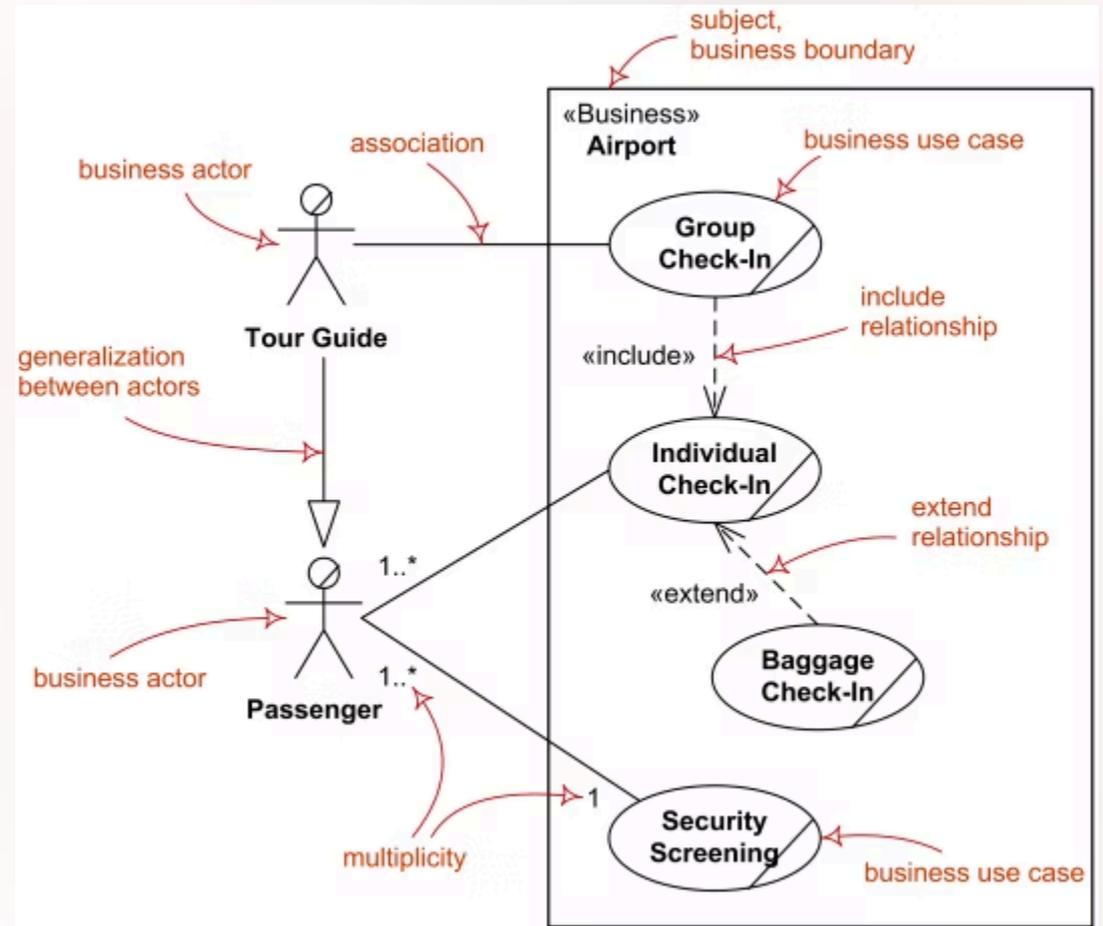
Створюємо актора "Гід"

Розширена роль з додатковими правами

03

Малюємо Generalization

Трикутна біла стрілка від "Гіда" до "Пасажира"



Анатомія діаграми (Нотація)

1

Варіант використання (Use Case)

Символ: Овал (еліпс). Окремий сценарій взаємодії з завершеною цінністю.

Правило: Дієслово + Іменник. Правильно: "Зняти готівку", "Створити звіт".

2

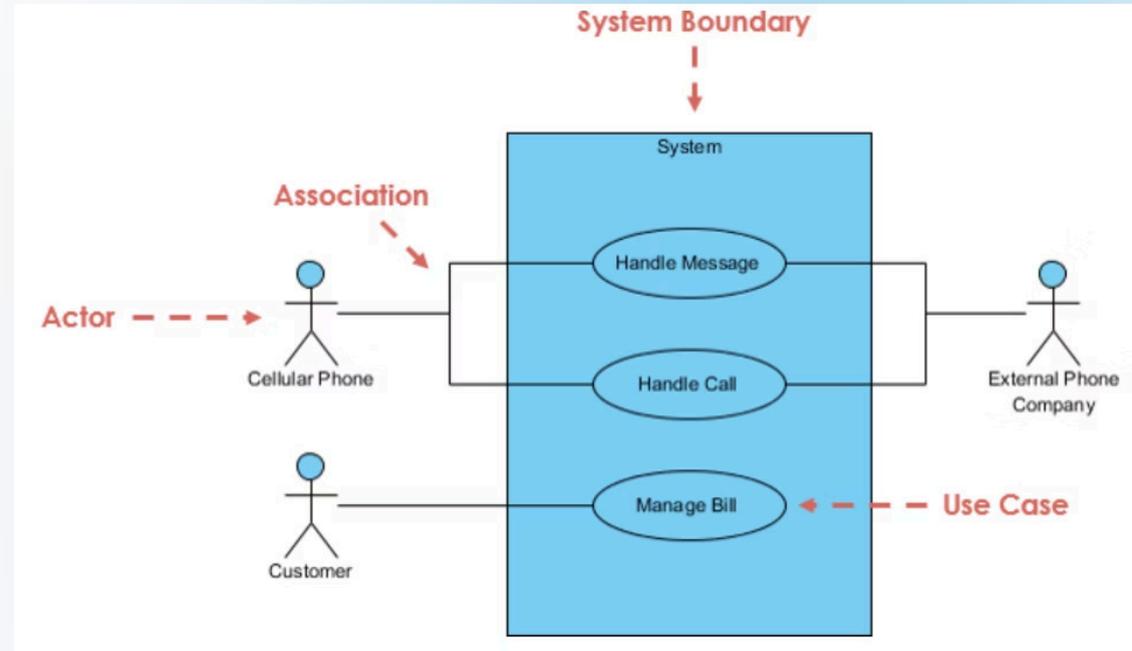
Межі системи (System Boundary)

Символ: Великий прямокутник навколо овалів. Це "паркан" — все всередині ми кодимо, ззовні — світ користувачів.

3

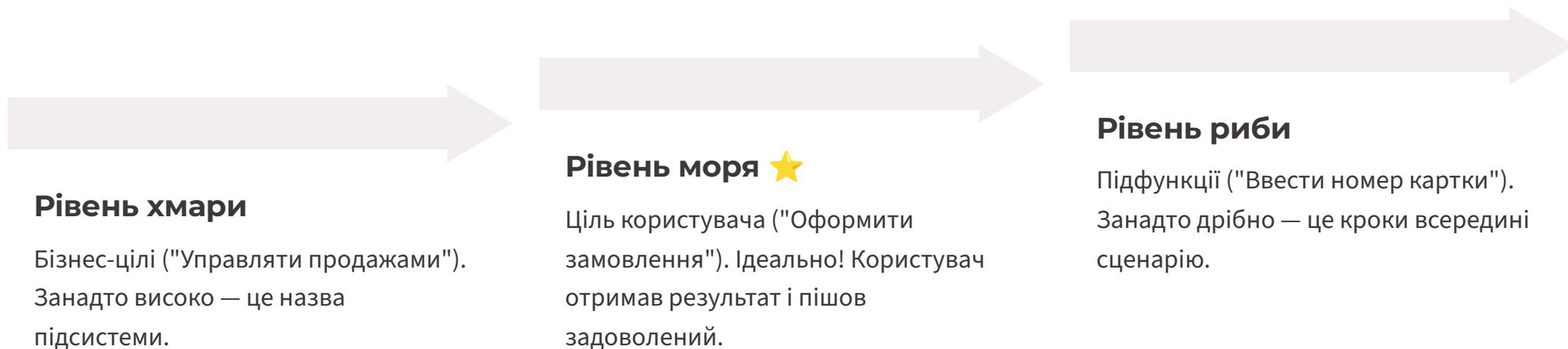
Асоціація (Association)

Символ: Суцільна лінія між Актором і Use Case. Показує, хто має право запускати сценарій.

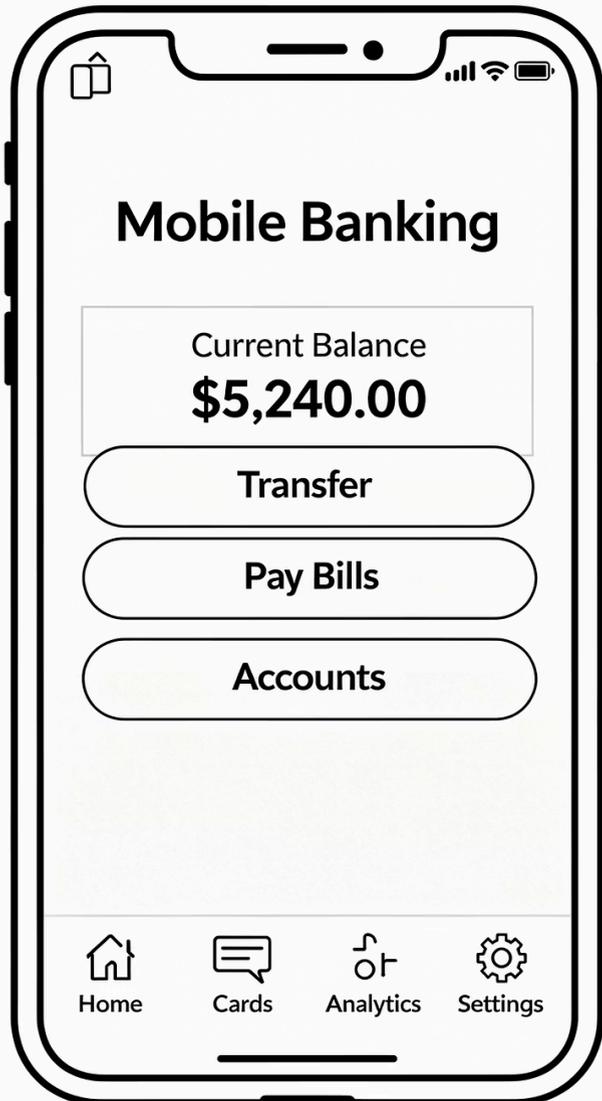


Рівень деталізації: "Рівень моря"

Найскладніше питання: "Наскільки дрібним має бути Use Case?". Алістер Коберн запропонував метафору рівнів:



📌 **Золоте правило:** Use Case — це дія, за яку користувач готовий заплатити або яка приносить йому відчутний результат.



Практичний приклад: Інтернет-банкінг

Use Cases всередині



Питання на засипку: Чи є "Авторизація" (Login) окремим юзкейсом? Так. Хоча це не самоціль, але це важливий безпековий бар'єр зі складними сценаріями (відновлення пароля, 2FA).

Принцип DRY в діаграмах

У програмуванні є принцип DRY (Don't Repeat Yourself). Якщо шматок коду повторюється в 10 місцях, ви виносите його в окрему функцію. У написанні Use Case сценаріїв діє те саме правило.

Уявіть сценарії: "Зняти готівку", "Переказати кошти", "Змінити PIN". У кожному першим кроком є: "Перевірити PIN-код". Чи треба копіювати ці кроки в кожен документ?

Ні. Ми виносимо їх в окремий Use Case і "викликаємо" його. Для цього в UML є два типи зв'язків: **<<include>>** та **<<extend>>**.

Ці механізми дозволяють створювати модульні, підтримувані специфікації без дублювання.

Відношення Включення (<<include>>)

Це аналог виклику підпрограми (функції) в кодї. Базовий Use Case завжди (обов'язково) викликає Включений Use Case.

Синтаксис

- Пунктирна стрілка
- Стереотип <<include>>
- Напрямок: Базовий → Включений
- Читання: "Я включаю в себе тебе"

Приклад

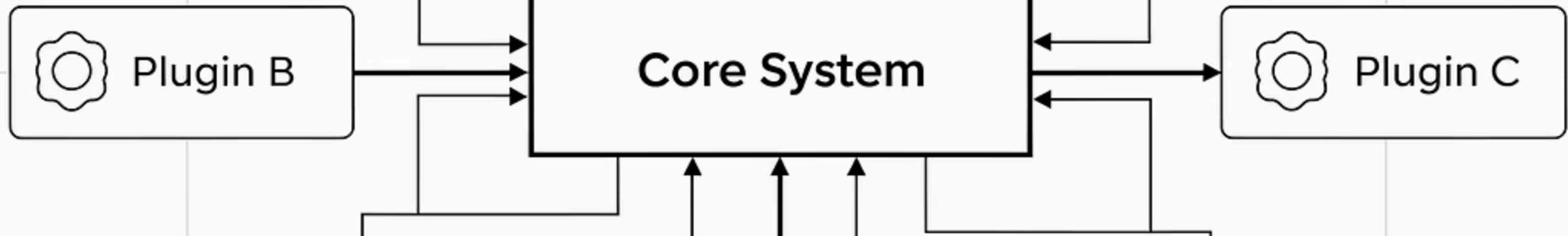
Базовий: "Оформити замовлення"

Включений: "Перевірити наявність товару"

Ви не можете завершити оформлення без перевірки складу — це обов'язковий крок.

Коли використовувати:

1. Для виділення загальних частин, що повторюються (Login, Validation)
2. Для декомпозиції дуже довгого сценарію



Відношення Розширення (<<extend>>)

Це аналог плагіна, DLC у грі або команди `if (condition) { doSomething() }`. Базовий Use Case може викликати Розширюючий, а може і не викликати.

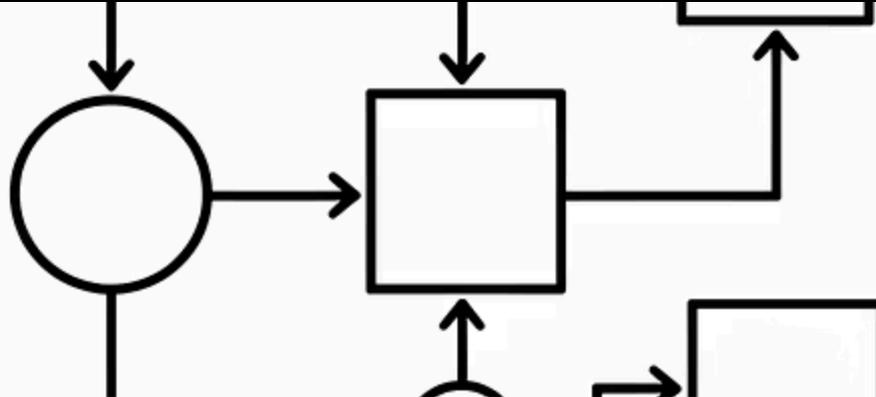
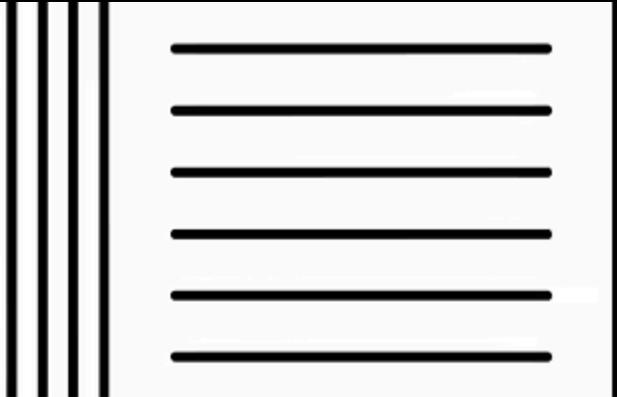
Синтаксис

- Пунктирна стрілка
- Стереотип <<extend>>
- Напрямок: Розширюючий → Базовий (навпаки від include!)

Точки розширення

У Базовому овалі вказуємо, де саме може спрацювати розширення (Extension Points).

Приклад: Базовий — "Оформити замовлення". Розширюючий — "Застосувати промокод". Ви можете оформити замовлення без промокоду. Промокод — це опція, яка вклинюється тільки якщо користувач натиснув кнопку.



Текст vs Діаграма

Який опис кращий: текстовий чи діаграма? Вибір за вами та вашою командою.

Переваги текстового опису:

- Зрозуміліший замовникам (вони теж воліють читати та узгоджувати варіанти використання)
- Текст простіше і швидше відредагувати, ніж діаграми + текст
- Менше часу на підтримку документації

Коли використовувати діаграми:

- Коли потрібна візуалізація складних зв'язків
- Для презентацій та обговорень
- Якщо команда звикла до візуального формату
- Коли є очевидні додаткові переваги

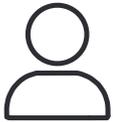
📄 У перші роки роботи аналітики використовують діаграми або діаграми + текстовий опис. З досвідом багато переходять на текстовий формат через його практичність.

Кому і коли потрібні сценарії



Розробникам

Дуже зручно, коли розгалужена вимога описана за допомогою основного та альтернативного потоку подій. Усе чітко і зрозуміло: хто, коли і що викликає, і що виходить у результаті.



Замовникам

Описано людською мовою, замовник своєчасно може підтвердити, що це саме те, чого він чекає, або виправити.



Тестувальнику

Майже готовий тест-кейс :-)



Усій проєктній команді

Якщо сценарій потрібно узгодити, а на кожній нараді пара-трійка альтернативних варіантів сценарію звучить інакше, допоможе суворо описаний потік подій.

У яких випадках вони потрібні

- Якщо потрібна якісна, повна специфікація вимог
- Для підтримки системи (виявити помилку, розібратися на якому кроці що пішло не так)
- Якщо потрібно описати частину функціональності у вигляді сценарію
- Для опису складних функцій з багатьма нюансами



Передумови та Постумови (Контракт)

Це "юридичні рамки" сценарію — перепустка на вхід і гарантія на виході.

Передумови (Pre-conditions)

Це перепустка в сценарій. Якщо передумова не виконана, сценарій навіть не починається.

- Банкомат у робочому стані
- У банкоматі є готівка
- Картка активна

Ми не перевіряємо це в кроках — ми вважаємо це фактом.

Постумови (Post-conditions)

Це гарантія результату. Що змінилося в світі після завершення?

- Баланс клієнта зменшено на суму видачі
- Запис про транзакцію створено в журналі
- Готівка видана

Це потрібно тестувальникам для перевірки.

Приклад 1: Розблокувати обліковий запис

Простий короткий приклад без альтернативного потоку подій.

Дійові особи	Адміністратор, Система
Мета	Змінити статус облікового запису користувача на «активний»
Передумова	Обліковий запис користувача не активний
Успішний сценарій	<ol style="list-style-type: none">Адміністратор вибирає користувача і активує «Розблокувати».Система перемикає обліковий запис користувача в статус «активний», і надсилає повідомлення користувачеві на email (якщо email не порожній)
Результат	Обліковий запис користувача було переведено в статус «активний»

Приклад 2: Авторизація користувача (1/2)

Складніший приклад з альтернативними потоками.

Дійові особи	Користувач, Система
Цілі	Користувач: авторизуватися в системі та почати працювати Система: ідентифікувати користувача та його права
Успішний сценарій	<ol style="list-style-type: none">1. Користувач запускає систему. Система відкриває сесію користувача, пропонує ввести логін і пароль.2. Користувач вводить логін і пароль.3. Система перевіряє логін і пароль.4. Система створює запис в історії авторизацій (IP-адреса користувача, логін, дата, робоча станція).5. Система видає користувачеві повідомлення з приводу успішної авторизації.
Результат	Користувач успішно авторизований і може працювати із системою

 Розширення (альтернативні потоки) на наступному слайді

Приклад 2: Розширення (2/2)

Альтернативні потоки для сценарію авторизації:

Розширення:

- ***a** Немає доступу до БД
Система видає повідомлення.
Результат: користувач не може увійти.
- **1a** У налаштуваннях безпеки для даної IP-адреси існує заборона на вхід у систему
Результат: форма логіна не надається, система видає повідомлення користувачеві.
- **2a** Користувач вибирає: «Нагадати пароль»
Викликається сценарій «Нагадати пароль».
- **3a** Користувача з введеними логіном і паролем не знайдено
Результат: відмова в авторизації.
Система видає повідомлення.
Перехід на крок 2.
- **3b** Кількість невдалих спроб авторизуватися досягла максимальної
Результат: користувач не може увійти.
Видається повідомлення.
Вхід з IP-адреси Користувача заблоковано на встановлений час.

 ***a** — це загальне розширення до сценарію, не до конкретного кроку. Розширення до кроку 1 позначається як 1a, 1b тощо.

Важливі моменти при написанні

1 Єдиний стиль

Якщо вся функціональність описана у вигляді юзкейсів, опишіть навіть прості сценарії юзкейсами. Нехай специфікація буде в єдиному стилі.

2 Мінімалізм

Використовуйте мінімальну кількість слів і пунктів для однозначного розуміння. Якщо юзкейс занадто довгий — розбийте на кілька.

3 Повторне використання

Якщо в двох і більше сценаріях повторюється однаковий набір кроків — винесіть їх в окремий сценарій і посилайтеся на них.

4 Централізація повідомлень

Список повідомлень, тексти листів розташуйте в єдиному місці та посилайтеся на них із різних юзкейсів.

5 Перечитайте

Перечитайте документ кілька разів перед тим, як віддавати його. Завжди знаходяться моменти для покращення.

6 Шаблони

Незаповнену таблицю для опису юзкейсу є сенс зберегти як шаблон для копіювання.

Висновки до Лекції 4

Ми завершили модуль функціонального моделювання. Тепер ви можете створити "скелет" вимог для будь-якої системи.



Поки що ми говорили лише про поведінку. Ми знаємо, ЩО хоче користувач ("Оформити замовлення"). Ми навіть написали текстовий сценарій.?

📄 **Наступна лекція:** Class Diagram (Діаграма класів) та моделювання даних — найпопулярніша діаграма у світі розробки.