МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»

МЕТОДИЧНІ ВКАЗІВКИ
до практичних занять з дисципліни «Основи програмної інженерії»

Одеса: Одеська політехніка, 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»

МЕТОДИЧНІ ВКАЗІВКИ

до практичних занять з дисципліни
«Основи програмної інженерії»
для здобувачів першого (бакалаврського) рівня вищої освіти

за спеціальністю

121 – Інженерія програмного забезпечення

Затверджено
на засіданні кафедри ІПЗ
Протокол № 1 від 30.08.2024 р.

Одеса: Одеська політехніка, 2025

Методичні рекомендації до практичних занять з дисципліни «Основи програмної інженерії для здобувачів першого (бакалаврського) рівня вищої освіти за спеціальністю 121 – Інженерія програмного забезпечення (освітньо-професійна програма: Інженерія програмного забезпечення – 2024) / Укл.: *Н. П. Волкова, В. А. Крісілов.* – Одеса: Одеська політехніка, 2025. – 47 с.

Укладачі: **Волкова Н.П.**, к.т.н., доцент
Крісілов В. А., д.т.н., професор

Методичні вказівки містять теоретичні відомості, які достатні для роботи на практичних заняттях з дисципліни «Основи програмної інженерії».

Призначаються для студентів денної форми навчання.

ЗМІСТ

Вступ

У сучасному світі розробки програмного забезпечення важливо не лише вміти писати код, а й розуміти процеси, методології та принципи, що забезпечують якість, масштабованість і надійність програмних продуктів. Курс «Основи програмної інженерії» допомагає студентам сформувати уявлення про життєвий цикл програмного забезпечення, оволодіти методами аналізу вимог, проєктування, тестування та підтримки, а також навчитися працювати в команді з використанням сучасних інструментів та підходів.

Метою практичних занять з дисципліни «Основи програмної інженерії» є розвиток у студентів здатності системно підходити до розробки програмних систем, опановувати техніки проєктування й управління програмними проєктами, а також набувати практичних навичок застосування методологій програмної інженерії для створення якісного програмного забезпечення.

Практичне заняття 1
Опис предметної області програмної системи.

Мета заняття: Навчити аналізувати предметну область програмної системи, визначати функціональні та не функціональні вимоги до системи.

Завдання: Описати предметну область запропонованої програмної системи; визначити вимоги до системи; зробити аналіз здійсненності; зробити аналіз існуючих рішень.

Теоретичні основи

1.1. Аналіз предметної області

Предметну область можна визначити як сферу людської діяльності, виділену і описану згідно з установленими критеріями. В описувані поняття повинні входити відомості про її елементи, явища, відносини і процеси, що відображають різні аспекти цієї діяльності.

Аналіз предметної області, дозволяє виділити її сутності, визначити початкові вимоги до функціональності і визначити межі проекту.

Щоб можна було проаналізувати предметну область, необхідно розглянути основні означення.

Замовник (customer) – особа (або компанія), яка купує програмне забезпечення та буде ним володіти. З замовником слід обговорювати постановку бізнес-задачі. Питання, які йому варто задати, це:

1. Чому взагалі пішла мова про створення системи?
2. У чому Ви бачите її призначення?
3. Які бізнес-можливості вона повинна реалізувати?
4. Які проблеми повинна вирішити?

Зацікавлені особи (stakeholders) – будь-хто зацікавлений в успіху програмного проекту та результатах, до яких він призведе. Зацікавлені особи можуть стати у нагоді під час бізнес-моделювання. Питання до них, за суттю, зводяться до «Що, чому, коли, як і ким відбувається в предметній області, та як воно взаємопов’язано?»:

1. Які основні поняття предметної області, їх визначення і взаємозв'язки?
2. На підставі яких правил відбувається те, що відбувається в предметній ділянці?
3. Що реально (які процеси, події, факти) відбувається та у якій послідовності, взаємозв'язку?
4. Якими структурними і поведінковими властивостями володіє кожне з виділених понять?

Користувачі (users) – представники замовника, які безпосередньо працюватимуть з програмною системою. Вони визначають вимоги користувача. Питання до них:

1. На яку систему буде схожа створювана?
2. З якими системами і як давно Ви працюєте?
3. Яка у Вас освіта?
4. Які Ваші очікування від системи – що і як вона повинна робити, які завдання допомагати вирішувати, як повинна виглядати?
5. Які кроки необхідно зробити для рішення кожного завдання?
6. У якому випадку Ви будете вважати, що система «хороша»?

1.2. Визначення та специфікація вимог

Визначення вимог призначене для формалізації очікуваного користувачем від системи сервісу.

Відомо, що вимоги можуть бути згруповані на **функціональні** та **нефункціональні**. Для опису функціональних вимог можна скористатися з такого шаблону:

−назва функціональної вимоги;

−вхідні данні – змістовний опис даних, які потрібні для успішної реалізації функції;

−джерело вхідних даних;

−результат;

−початкова умова – умова, яка має отримати значення «істина» для того, щоб можна було розпочати працювати з функцією;

−кінцева умова – умова, яка отримує значення «істина» після успішного виконання функції;

−побічні ефекти – які зміни в інформаційній системі відбуваються під час виконання функції;

−причина – чому потрібна реалізація функції.

Для визначення основних функцій програмного продукту можна використовувати, наприклад, короткий опис варіанта використання. Опис кожної функції можна представити також у вигляді списку, що складається з трьох граф: дійова особа, мета та короткий опис варіанта використання.

Часта помилка формування **нефункціональних** вимог – нестача критеріїв, визначених для перевірки вимог. Кожна нефункціональна вимога повинна бути перевірена, для чого потрібний адекватний вибір критеріїв. Деякі критерії наведено в таблиці 1.1.

Таблиця 1.1 − Критерії для перевірки не функціональних вимог

Характеристика	Міра
Продуктивність	Кількість транзакцій за секунду Час відгуку
Розмір	Кількість записів у базі даних Потрібний розмір пам'яті
Зручність для користувача	Час, потрібний для навчання Розмір документації
Надійність	Вірогідність помилки транзакції Час між відмовами Доступність (час, коли програма доступна для користувача) Час перезавантаження програми після помилки Вірогідність втрати даних після помилки
Переносимість	Розмір платформозалежного коду Кількість платформ Вартість перенесення

1.3. Аналіз здійсненності

Розробка нових програмних систем повинна починатися з аналізу здійсненності. На підставі аналізу предметної області, загального опису системи і її призначення необхідно прийняти рішення про продовження або завершення проекту. Для цього необхідно відповісти на наступні питання.

1. Чи відповідає система, що розроблюється цілям замовника?
2. Чи можна реалізувати систему, використовуючи відомі технологи?
3. Чи можна об'єднати систему з іншими системами, що вже експлуатуються?

Частково вимоги, що виявлені з предметної області вже визначено, і тепер залишилося дослідити користувальницькі вимоги. Призначені для користувача вимоги, як можна зрозуміти, потрібно виявляти зі спілкування з потенційними користувачами системи.

Питання:

1. На яку систему буде схожа створювана?
2. Які завдання повинна допомагати вирішувати система і як повинна виглядати?
3. Які кроки необхідно зробити для вирішення кожного завдання?
4. Які критерії оцінки системи?

1.4. Аналіз існуючих рішень

Якщо існують аналоги програмного продукту, що розробляється, то необхідно провести аналіз існуючих рішень і виділити найбільш важливі вимоги і недоліки, які будуть виправлені, тобто оцінити програмне забезпечення з різних аспектів:

1. функціонал;
2. графічний інтерфейс;
3. технології, які використовували для реалізації програмного продукту;
4. очікувана аудиторія даного програмного забезпечення.

1.5. Інформація для завдання до практичної роботи

Ігор Томін, генеральний директор компанії Market Research, з'явився на роботі як завжди за 5 хвилин до початку робочого дня. Його компанія займалася дослідженням ринку. За 8 років існування компанії була напрацьована стабільна клієнтська база організацій, які купували в них звіти з аналізу ринку. Деякі великі клієнти також купували в Market Research спеціалізоване програмне забезпечення, призначене для створення звітів. Для таких клієнтів вони вирішили надавати додатковий сервіс – забезпечувати неопрацьованою і агрегованою інформацією для генерації власних звітів.

Незважаючи на напрацьовану клієнтську базу, Market Research постійно вела пошук нових клієнтів. Політика компанії приписувала пестити нових клієнтів, навіть якщо вони зацікавлені тільки в одержанні однократних, вузькоспеціалізованих звітів по ринку. Крім того, компанія вела пошук перспективних клієнтів і, хоча в повному сенсі це ще не клієнти, прагнула встановити з ними контакт.

Отже на поточний момент вони мали величезну кількість інформації про контакти, які класифікувалися як контакти з перспективними, реальними та колишніми клієнтами. Через необхідність обробляти ці обсяги даних і по-різному реагувати на різні події, пов'язані з різними категоріями контактів, для Market Research дуже важливо стало створити систему управління контактами з клієнтами.

Інформація до розмірковування

Система керування взаємодією з клієнтами (Customer Relationship Management System, CRM-система) – корпоративна інформаційна система, яка призначена для покрашення обслуговування клієнтів за рахунок збереження інформації про клієнтів та історії взаємин з ними, установлення та покращення бізнес-процедур на основі збереженої інформації та подальшої оцінки їхньої ефективності. Її основні принципи такі:

1. наявність єдиного сховища інформації, звідки в будь-який момент доступні всі відомості про всі випадки взаємодії із клієнтами;
2. синхронізованість керування множинними каналами взаємодії (тобто існують організаційні процедури, які регламентують використання цієї системи та інформації в кожному підрозділі компанії);
3. постійний аналіз зібраної інформації про клієнтів і прийняття відповідних організаційних рішень, наприклад, пріоритизації клієнтів на основі їхньої значущості для компанії.

Таким чином, цей підхід передбачає, що при будь-якій взаємодії з клієнтом по будь-якому каналі, співробітникові організації доступна повна інформація про всі

взаємини з клієнтами, та рішення приймається на її основі. Інформація про рішення, у свою чергу, теж зберігається і доступна при всіх наступних взаємодіях

Ігор вважав себе не абияким фахівцем в інформаційних технологіях, і він усвідомлював, що створення подібної системи – справа складна, яка зажадає великих часових і фінансових вкладень. Тому він уже прийняв рішення розпочати з усіченої, неповнофункціональної, версії CRM – потренуватися на «менеджері контактів». Тобто через два місяці система повинна забезпечити гнучке планування і перепланування видів діяльності, пов'язаних з контактами, так щоб співробітники могли успішно кооперувати свої зусилля в завоюванні нових клієнтів і стимулюванні існуючих взаємин із клієнтами. А потім уже будуть приймати рішення, чи розвивати систему далі.

Крім цього, Ігор прийняв ще два рішення:

1) нова система управління контактами має розроблятися власними силами і перебувати в розпорядженні всіх працівників компанії, але з забезпеченням різного рівня доступу;
2) шефство над системою беруть співробітники відділу обслуговування клієнтів (Customer Services Department).

1.6 Приклад виконання практичної роботи

1.6.1 Аналіз предметної області

Інтернет-магазин спортивних товарів — це вебсистема, що надає користувачам можливість переглядати асортимент спортивного обладнання, одягу та аксесуарів, здійснювати пошук, формувати замовлення та оплачувати їх онлайн.

Виявлення користувачів системи та їх обов’язків (список користувачів)

Стейкхолдери - (Покупець, Персонал)

Покупець − відвідувач сайту, який може реєструватися, переглядати каталог, додавати товари в кошик, оформлювати замовлення.

Персонал − керує каталогом товарів, цінами, акціями, обробляє замовлення та контролює доставку.

1.6.2. Визначення функціональних вимог

Інтернет-магазин спортивних товарів з повинен мати наступні функціональні вимоги:

1. Реєстрація та авторизація

Ця функція має надати користувачам системи (Користувач, Персонал) доступ до інтернет-магазину спортивних товарів

Передумова: Користувач бажає увійти на сайт магазину спортивних товарів.

Результат (післяумова): Користувач отримав доступ до функціоналу інтернет- магазину спортивних товарів відповідно до своєї ролі.

2. Перегляд каталогу товару

Ця функція має надати користувачам системи (Користувач, Персонал) можливість переглянути каталог товарів інтернет- магазину спортивних товарів

Передумова: Користувач, авторизований на сайті як покупець чи як персонал.

Результат (післяумова): Користувач (Персонал чи Покупець) отримав доступ до переліку товарів

3. Облік товарів

Ця функція має надати користувачу інтернет- магазину (Персонал) можливість проводити облік товарів

Передумова: Користувач авторизований на сайті як Персонал.

Результат (післяумова): Користувач (Персонал) виконав дію з додавання, редагування чи видалення товарів потрібної товарної категорії.

1.6.3. Визначення нефункціональних вимог

Інтернет-магазин з повинен мати наступні нефункціональні вимоги.

Характеристика «Інтерфейс користувача». Інтерфейс має бути зрозумілим та інтуїтивним. Не менш 90% користувачів системи мають підтвердити такі його властивості. Користувачі мають розуміти як виконати ті дії стосовно покупки товари, який вони бажають. Також після значної перерви у користуванні сайтом покупець не повинен мати потреби довго згадувати, як він виконував певні функції та на якому кроці зупинився.

Характеристика «Апаратні та програмні платформи». Система буде виконана у вигляді вебсайту, тому не має певних вимог до апаратної складової. Проте, для комфортної роботи веббраузерів, комп'ютерам слід мати Windows 7+/Windows Server 2008 R2+, Core i3 9100F, 4 ядра, ОЗУ 8 Гб.

Характеристика «Безпека та конфіденційність». Дані покупців для здійснення покупок та авторизації зберігаються з використанням хеш-алгоритмів. Сайт інтернет-магазину має бути захищеним від DDoS-атак.

Характеристика «Надійність». Сайт повинен працювати стабільно та без збоїв, бути доступним 95% усього часу. Усі дані повинні мати резервні копії. При пошкодженні даних щодо товару вони повинні відновлюватись не довше ніж на 5 годин, дані користувачів – не довше ніж за 2 години.

Характеристика «Продуктивність». Система повинна мати відгук на будь-які дії користувача не більше за 1,5 секунди при сильному навантаженні та не більше ніж за 1 секунду при звичайному навантаженні.

1.6.4. Аналіз здійсненності

Розробка нових програмних систем повинна починатися з аналізу здійсненності. На підставі аналізу предметної області, загального опису системи і її призначення необхідно прийняти рішення про продовження або завершення проекту. Для цього необхідно відповісти на наступні питання.

1. Чи відповідає система, що розроблюється цілям замовника?
2. Чи можна реалізувати систему, використовуючи відомі технологи?
3. Чи можна об'єднати систему з іншими системами, що вже експлуатуються?

1.6.5. Аналіз існуючих рішень

Таблиця 1.2 – Аналіз програмних аналогів

	Назва продукту			
Характеристики продукту	S4S	ROZETKA	ANDSPORT	Інтернет-магазин товарів
Зручний інтерфейс	+	+	+/-	+
Невелика кількість кроків для вибору товару	-	-	+/-	+
Відсутність реклами	+	-	-	+
Зворотній зв'язок від покупців щодо враження від користування сайтом	+	-	-	+

1.6.6. Висновки

У роботі виконано опис предметної області запропонованої програмної системи. Сформульовано функціональні та нефункціональні вимоги до інтернет-магазину спортивних товарів. Проведено аналіз здійсненності розробки запропонованої програмної системи та здійснено аналіз існуючих програмних рішень, що дало змогу виявити їхні переваги та недоліки й обґрунтувати доцільність створення власної системи.

Контрольні питання

1. Що таке предметна область програмної системи та які її основні елементи?
2. Які питання варто поставити замовнику під час аналізу предметної області?
3. Хто такі зацікавлені особи (stakeholders) та яку роль вони відіграють у проєкті?
4. Які питання доцільно ставити зацікавленим особам під час бізнес-моделювання?
5. Які питання слід задати потенційним користувачам програмної системи?
6. Що таке функціональні та нефункціональні вимоги?
7. За яким шаблоном можна описати функціональну вимогу?
8. Які аспекти слід врахувати під час аналізу здійсненності проєкту?
9. Які питання варто поставити при оцінюванні технічної та організаційної здійсненності системи?
10. Для чого потрібен аналіз існуючих рішень під час створення програмної системи?
11. Які основні характеристики слід враховувати при аналізі аналогів програмного забезпечення?

Практичне заняття 2

Розробка структури системи з використанням діаграм стандарту IDEF0 для опису бізнес-процесів.

Мета заняття: ознайомитися з принципами проектування на основі CASEтехнології; навчитися будувати структурну схему програмної системи; оволодіти навичками щодо побудови моделі з використанням стандарту IDEF0 в Ramus Educational.

Зміст заняття: На основі аналізу предметної області створити модель структури системи, шляхом декомпозиції її на підсистеми.

Теоретичні основи

2.1. Базові поняття

Модель – опис системи (текстовий або графічний) з визначеним рівнем деталізації.

Об'єкт моделі – об'єкт у базі даних інструментального середовища моделювання, який має певні атрибути та призначений для відображення реально існуючого об'єкта визначеного типу.

Бізнес-процес (процес) – цілеспрямована послідовність процедур, яка необхідна для отримання заданого кінцевого результату.

Процедура (підпроцес) – впорядкована послідовність операцій, яка спрямована на отримання проміжного результату.

Бізнес-операція (операція) – ряд впорядкованих дій, розглядати які окремо в рамках моделі, що створюється, недоцільно.

Декомпозиція бізнес-процесу – детальний опис бізнес-процесу, що здійснюється шляхом розподілу процесу на декілька частин і подальшого їх опису за допомогою більш докладних моделей.

Вхід бізнес-процесу – об'єкт бізнес-процесу (процедура, операція), який взаємодіє з

зовнішніми бізнес-процесами та отримує від них інформацію (ресурси).

Вихід бізнес-процесу – об'єкт бізнес-процесу (процедура, операція), який взаємодіє з зовнішніми бізнес-процесами та передає їм інформацію (ресурси), що є результатом виконання бізнес-процесу.

Ініціююча подія – об'єкт моделі бізнес-процесу, який відображає подію, яка є управляючим впливом, що необхідне для початку виконання процедури.

Завершуюча подія – об'єкт моделі бізнес-процесу, що відображає факт завершення процедури та отриманий при цьому результат.

Для опису роботи об'єктові управління (підприємства, проекту) необхідно побудувати модель. Модель є загальним описом предметної області. Проте модель має бути адекватна предметній області та містити в собі знання всіх учасників опису бізнес-процесів організації.

Найбільш зручною мовою моделювання процесів бізнесу є технологія структурного аналізу SADT (Structured Analysis and Design Technique), а саме стандарт IDEF0.

Сутність структурного підходу до розробки ІС, проектів бізнесу полягає в її декомпозиції, тобто розподілу на функції, що автоматизуються: система розподіляється на функціональні підсистеми, які, в свою чергу, поділяються на підфункції, що підрозділяються на завдання й так далі. Процес розподіл триває аж до конкретних процедур. Система, що автоматизується, зберігає цілісне уявлення, в якому всі компоненти взаємопов'язані. Під час розробки системи "знизу-вгору" від окремих завдань до всієї системи цілісність втрачається, виникають проблеми під час інформаційного стикування окремих компонентів.

Система (грец. – «складене з частин», «з'єднання» від «з'єдную») – множество елементів, що знаходяться у відносинах і зв'язках один з одним, яке утворює певну цілісність, єдність.

Як випливає з визначення, головною властивістю системи є її цілісність: комплекс об'єктів, що розглядаються в якості системи, повинен володіти суспільними властивостями і поведінкою. Очевидно, необхідно розглядати і зв'язки системи з навколишнім середовищем. У найзагальнішому випадку поняття «система» характеризується:

− наявністю безлічі елементів;
− наявністю зв'язків між ними;
− цілісним характером даного пристрою або процесу.

Система повинна являти собою сукупність елементів (об'єктів, суб'єктів), які перебувають між собою в певній залежності і складових деякий єдність (цілісність), спрямоване на досягнення певної мети. Система може бути елементом іншої системи вищого порядку (надсистема) і включати в себе системи нижчого порядку (підсистеми). Тобто систему можна розглядати як набір підсистем, організованих для досягнення певної мети і описаних за допомогою набору моделей (можливо, з різних точок зору), а підсистему – як групу елементів, частина яких становить специфікацію поведінки, представленого іншими її складовими.

До типових можна віднести наступні підсистеми:

− підсистему управління;
− підсистеми введення-виведення:
− підсистему налаштування параметрів;
− файлову підсистему;
− підсистему візуалізації;
− підсистему документування;
− підсистему взаємодії з базою даних;
− довідкову підсистему.

Отримана в результаті декомпозиції структура системи повинна супроводжуватися коротким описом включених в неї підсистем.

На рис. 2.1 представлений принцип системного аналізу.

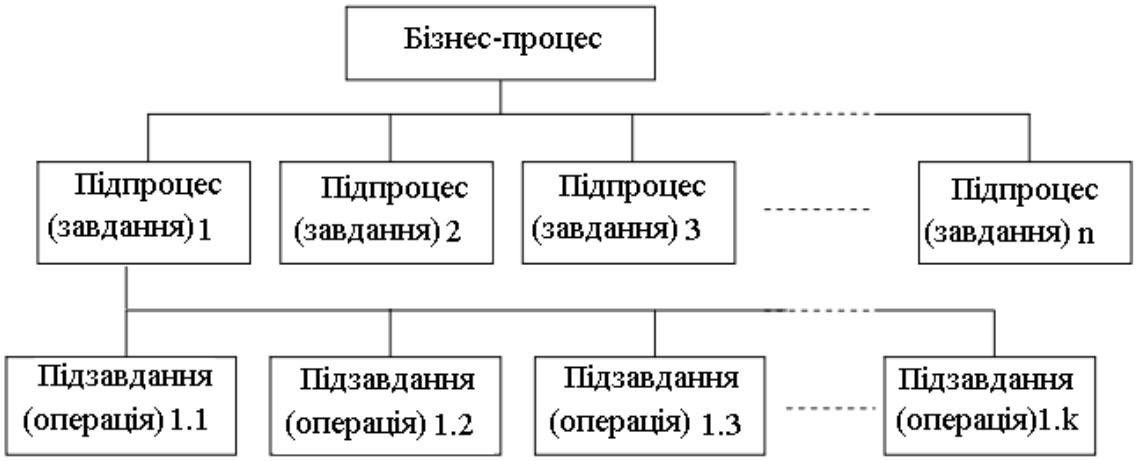


Рисунок 2.1 – Принцип системного аналізу

Усі найбільш поширені методології структурного підходу базуються на низці загальних принципів. Як два базові принципи використовуються такі:

− принцип "розподіляй і володарюй" – принцип розв'язання складних проблем шляхом їх розподілу на безліч менших незалежних завдань, легких для розуміння і вирішення;

− принцип ієрархічного впорядковування – принцип організації складових частин проблеми в ієрархічні деревовидні структури з додаванням нових деталей на кожному рівні.

Виділення двох базових принципів не означає, що решта принципів є другорядними, оскільки ігнорування будь-якого з них може призвести до непередбачуваних наслідків (у тому числі й до провалу всього проекту). Основними з цих принципів є такі:

− принцип абстрагування, який полягає у виділенні суттєвих аспектів системи і відвернення від несуттєвих;

− принцип формалізації, що полягає в необхідності суворого методичного підходу до вирішення проблеми;

− принцип несуперечності, який полягає в обґрунтованості й узгодженості елементів;

− принцип структуризації даних, який полягає в тому, що дані повинні бути структуровані й ієрархічно організовані.

У структурному аналізі використовуються в основному дві групи засобів, що ілюструють функції, які виконуються системою і відносини між даними. Кожній групі засобів відповідають певні види моделей (діаграм), найбільш поширеними серед яких є такі:

−SADT (Structured Analysis and Design Technique) – моделі, а також функціональні та динамічні діаграми (IDEF0, IDEF3);

−DFD (Data Flow Diagrams) – діаграми потоків даних;

−ERD (Entity-Relationship Diagrams) – діаграми "сутність-зв'язок".

Методологія SADT є сукупністю методів, правил і процедур, призначених для побудови функціональної моделі об'єкта будь-якої предметної області. Функціональна модель SADT відображає функціональну структуру об'єкта, тобто дії, які ним виконуються, і зв'язки між цими діями. Графічне подання блочного моделювання. Графіка блоків і дуг SADT-діаграми відображає функцію у вигляді блоку, а інтерфейси входу/виходу подаються дугами, що входять у блок і виходять з нього, відповідно. Взаємодія блоків один із одним описуються за допомогою інтерфейсних дуг, що виражають "обмеження", які в свою чергу визначають, коли і яким чином функції виконуються і управляються.

Основні елементи цієї методології ґрунтуються на таких концепціях:

строгість і точність. Виконання правил SADT вимагає достатньої строгості та точності, не накладаючи у той же час надмірних обмежень на дії аналітика. Правила SADT включають:

−обмеження кількості блоків на кожному рівні декомпозиції (як правило, 3 – 6 блоків);

−зв'язність діаграм (принципи нумерації блоків);

−унікальність міток і найменувань (відсутність імен, що повторюються);

−синтаксичні правила для графіки (блоків і дуг);

−розділення входів, механізмів та управлінь (правило визначення ролі даних);

− відокремлення організації від функції, тобто виключення впливу організаційної структури на функціональну модель.

Будь-яка система має межу, яка відокремлює її від зовнішнього світу (інших систем). Взаємодія системи з навколишнім світом описується як вхід (ресурс, який переробляється системою – показується з лівої сторони блоку), вихід (результат діяльності системи – показується з правої сторони блоку), управління (стратегії і процедури, під управлінням яких проводиться робота – показується з верхньої сторони блоку) і механізм (ресурси, необхідні для проведення роботи – показується з нижньої сторони блоку).

Перебуваючи під управлінням, система перетворює входи у виходи, використовуючи механізми перетворення. У IDEF0 система подана як сукупність взаємодіючих робіт або функцій. Така функціональна орієнтація є принциповою – функції системи аналізуються незалежно від об'єктів, якими вони оперують. Це дозволяє чіткіше моделювати логіку і взаємодію процесів організації.

Блоки у IDEF0 розміщуються за ступенем важливості, як її розуміє автор діаграми. Цей відносний порядок називається домінуванням. Домінування розуміється як вплив, який один блок здійснює на інші блоки діаграми. Наприклад, найбільш домінуючим блоком діаграми може бути або перший з можливих функцій, який є ініціюючою подією, або функції планування чи контролю, які впливають на всі інші. Найбільш домінуючий блок зазвичай розміщується у верхньому лівому куті діаграми, а найменш домінуючий – у правому нижньому куті.

Одним з інструментів, що дозволяє структурно-функціональне моделювання, є Ramus. Існує дві основні версії програми Ramus: комерційна версія Ramus і Ramus Educational (який буде використовуватися в рамках цього лабораторного практикуму).

Ramus Educational – це безкоштовний аналог Ramus. Ramus Educational використовується для побудови діаграм у форматі IDEF0 і DFD.

2.2. Принципи побудови моделі IDEF0 у Ramus Educational

Процес моделювання будь-якої системи в IDEF0 у Ramus почина-ється з визначення контексту, тобто найбільш абстрактного рівня опису системи в цілому. У контекст входить визначення суб'єкта моделювання, мети і точки зору на модель.

Під суб'єктом розуміється сама система. Основним завданням є визначення складових системи і зовнішніх дій. На визначення суб'єкта системи суттєво впливає точка зору (позиція), з якої розглядається система, і ціль моделювання, тобто спочатку необхідно визначити область моделювання.

Опис області як системи в цілому, так й її компонентів, є основою побудови моделі. В процесі моделювання область може корек-туватися, але сформульована вона має бути спочатку, оскільки область визначає напрям моделювання і критерії завершення процесу моделю-вання. Під час формулювання області необхідно враховувати широту і глибину моделі (системи). Широта передбачає визначення меж моделі – кількість компонент всередині та поза системою і їх зв'язки. Глибина ви-значає рівень деталізації моделі.

Визначення меж моделі передбачає, що нові об'єкти не будуть вне-сені в модельовану систему, оскільки внесення нового об'єкта може змі-нити існуючі взаємозв'язки в системі (проблема "плаваючої області").

Ціль моделювання. Основою проектування є системний підхід, що визначає сувору залежність цілей і завдань проектування моделі. Модель не може бути побудована без чітко сформульованої мети. Мета повинна визначати предметну область проектування, завдання проектування, результати проектування. Формулювання цілі дозволяє групі ана-літиків фокусувати зусилля в потрібному напрямі. Прикладами формулювання мети можуть бути такі твердження: "Ідентифікувати і визначити поточні проблеми об'єкта управління (об'єкта

аналізу), зробити можливим аналіз потенційних поліпшень", "Визначити найбільш витратні підрозділи підприємства", "Описати організаційно-функціональну структуру підприємства з метою подальшої розробки інформаційної системи" і т. д.

Під час побудови моделі враховують думки різних фахівців, однак модель має будуватися згідно з єдиною точкою зору. Точку зору можна уявити як певний аспект моделювання. Слід зазначити, що точка зору повинна відповідати цілі моделювання. Як правило, вибирається точка зору фахівця, відповідального за процес моделювання. У ході вибору точки зору на модель важливим є документування альтернативних моделей, представлень предметної області. Для цієї цілі зазвичай використовують діаграми (IDEF0, IDEF3).

Модель може містити два типи діаграм:

1. контекстна діаграма (у кожній моделі може бути лише одна кон-текстна діаграма);
2. діаграма декомпозиції.

Контекстна діаграма є вершиною деревовидної структури діаграм та найзагальнішим описом системи та її взаємодії з зовнішнім середовищем. Після опису системи в цілому проводиться її розподіл на фрагменти. Цей процес називається функціональною декомпозицією, а діагра-ми, які описують кожен фрагмент і взаємодію фрагментів, називаються діаграмами декомпозиції. Після декомпозиції контекстної діаграми про-одиться декомпозиція кожного фрагмента системи на дрібніші і так далі, до досягнення потрібного рівня деталізації опису. Після кожного сеансу декомпозиції проводиться експертиза – експерти предметної області визначають відповідність реальних процесів бізнесу створеним діаграмам. Знайдені невідповідності виправляються, і здійснюється перехід до наступного рівня декомпозиції. Дана процедура дозволяє досягти відповідність моделі реальним процесам бізнесу на будь-якому рівні моделі. Синтаксис опису системи в цілому і кожного її фрагмента є однаковим у всій моделі.

2.3. Хід виконання роботи

1. Встановити JRE від версії 6 та вище.
Джерело: https://www.java.com/en/download/manual.jsp
2. Встановити Ramus, або Ramus Educational, або Ramus Portable.

Для початку роботи з Ramus Educational необхідно зайти в пункт меню **Пуск** та зі списку програм вибрати **Ramus Educational.**

Далі, відповідно до попередніх настройок, буде потрібно або вибрати пункт меню **Файл/Новий проект**, або вікно створення/відкриття моделі буде автоматично відкрито під час завантаження Ramus Educational.

Необхідно створити нову модель з назвою "Управління договорами" в стандарті IDEF0.

Після натискання кнопки **ОК** буде запропоновано внести властивості для створеної моделі, а саме: **Автор** – внести прізвище автора моделі (розробника) – слід внести у це поле свої дані; **Назва проекту** – ввести назву проекту; **Назва моделі** – ввести назву моделі. Обрати тип діаграми: IDEF0. Після цього натиснути на кнопку **Далі**.

На наступному етапі ввести назву підприємства, для якого створюються проект. Після цього натиснути на кнопку **Далі**.

На наступному етапі ввести короткий опис проекту. Після цього натиснути на кнопку **Закінчити**. Під час цього буде автоматично відкрито робочий простір для створення контекстної діаграми, визначається склад осіб, задіяних в даному процесі, визначається вхідна і вихідна інформація, будується структурна схема типу "чорний ящик" (рис.2.2).

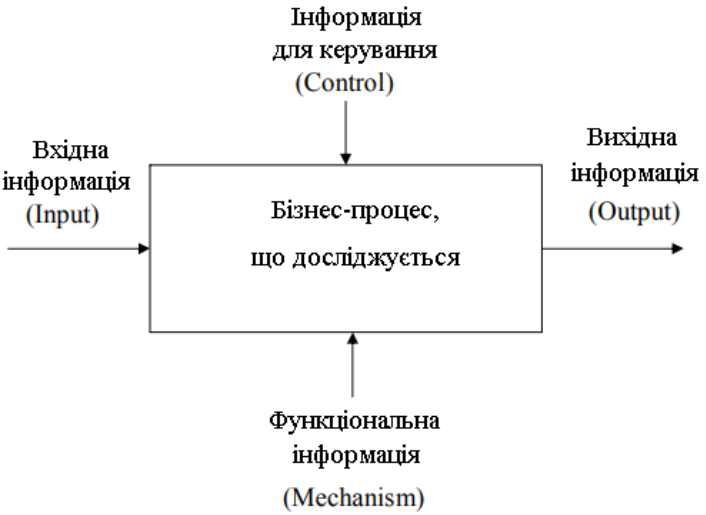


Рисунок 2.2 – Реалізація принципу чорного ящику

Будується структурна схему типу "чорний ящик" (рис. 2.3) для системи управління взаємодією з клієнтами (Customer Relationship Management System, CRM-система) - корпоративна інформаційна система, призначена для автоматизації CRM-стратегії компанії, зокрема, для підвищення рівня продажів, оптимізації маркетингу та покращення обслуговування клієнтів шляхом збереження інформації про клієнтів (контрагентів) та історії взаємовідносин з ними, встановлення та покращення бізнес-процедур та подальшого аналізу результатів.

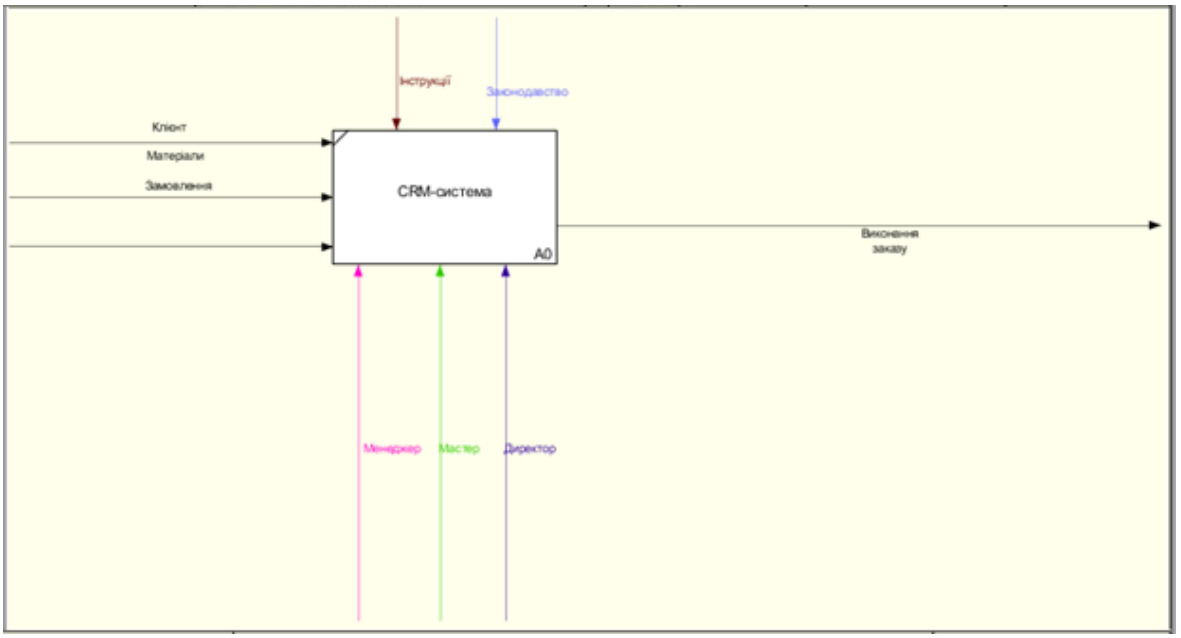


Рисунок 2.3 – Структурна схема типу "чорний ящик"

Для переходу на наступний рівень, необхідно провести декомпозицію контекстної діаграми. Для цього потрібно виділити необхідну роботу (у даному випадку вона одна) та вибрати інструмент **Перейти до дочірніх діаграм**. У вікні, що відкривається, обрати стандарт, який буде використовуватися на наступному рівні декомпозиції, кількість робіт на ньому та шаблон (найчастіше використовується шаблон "простий", але користувач має можливість обрати той, який для нього є найзручнішим).

Обираємо шаблон "Простий", стандарт IDEF0 та кількість робіт – 4, підтверджуємо натисканням кнопки **ОК**.

Контекстна (коренева) робота дерева має номер А0. Роботи декомпозиції А0 мають номери А1, А2, А3 і т. д. Роботи декомпозиції нижнього рівня мають номер батьківської роботи і черговий порядковий номер, напри-клад, роботи декомпозиції А3 матимуть номери А31, А32, А33, А34 і т. д. Роботи утворюють ієрархію, де кожна робота може мати одну батьківську і декілька дочірніх робіт, утворюючи дерево. Таке дерево називають деревом вузлів, а описану нумерацію – нумерацією по вузлах. Ramus автоматично підтримує нумерацію за вузлами, тобто у ході проведення декомпозиції створюється нова діаграма і їй автоматично привласнюється відповідний номер.

Декомпозиція CRM – системи показана на рисунку 2.4.

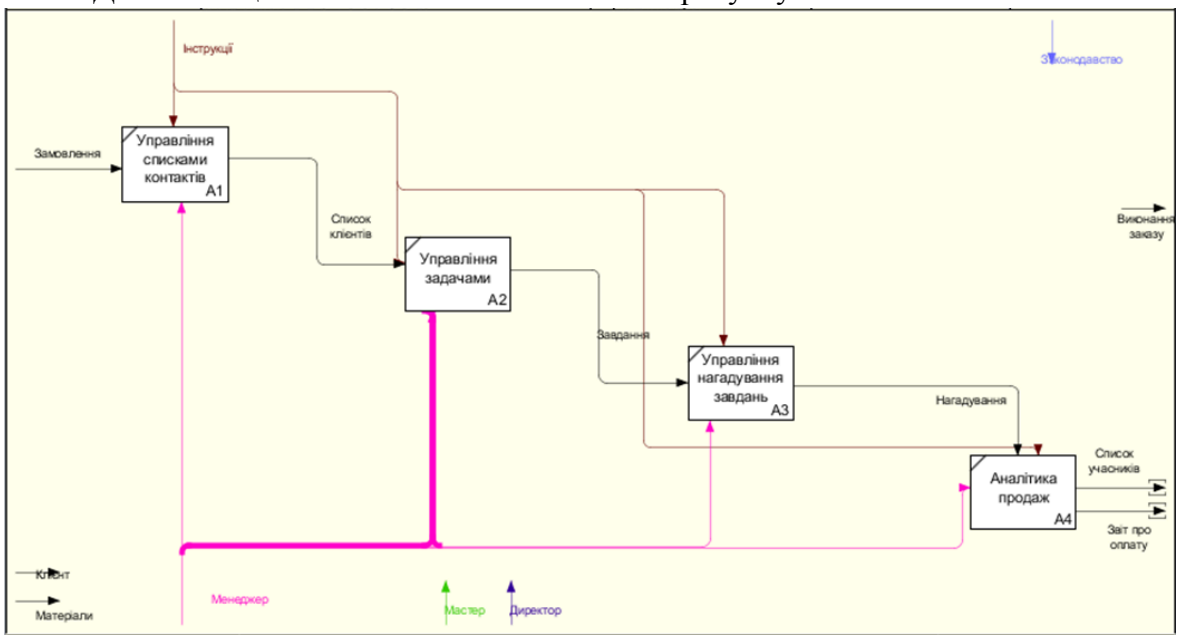


Рисунок 2.4 – Декомпозиція CRM – системи

Далі потрібно виконати декомпозицію кожного блоку. На рисунку 2.5 представлено приклад декомпозиції блоку «Управління списками контактів».

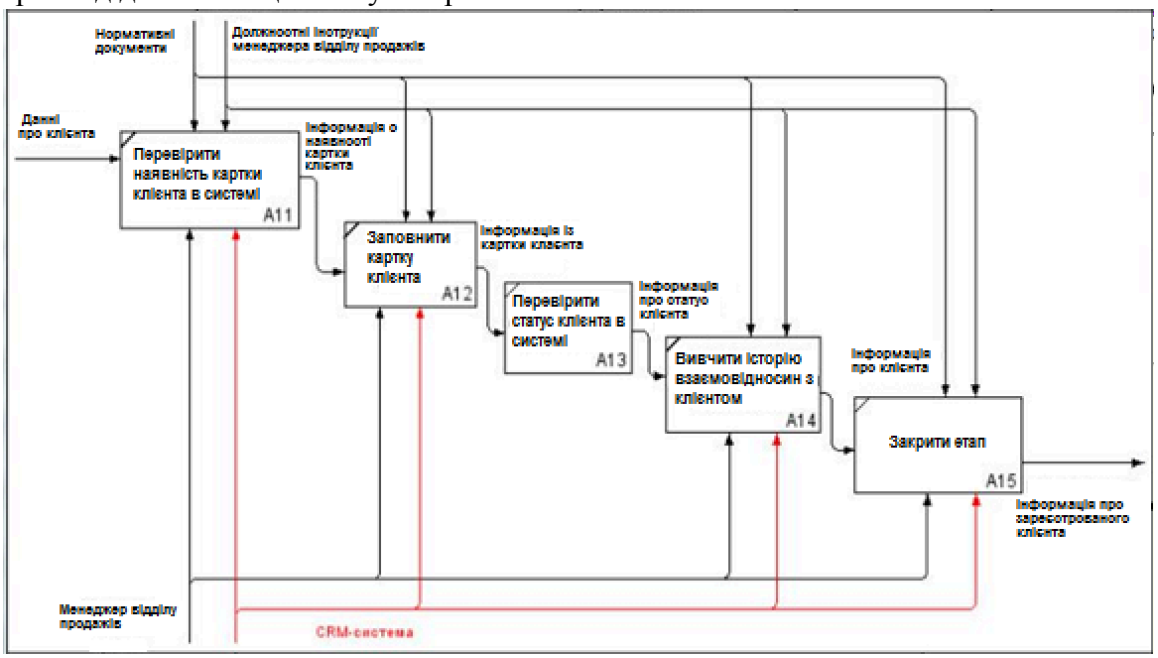


Рисунок 2.5 – Декомпозиція блоку «Управління списками контактів»

Дкомпозиція бізнес-функції «Вивчити історію взаємовідношень з клієнтами» може бути

представлено наступним чином (рис. 2.6).

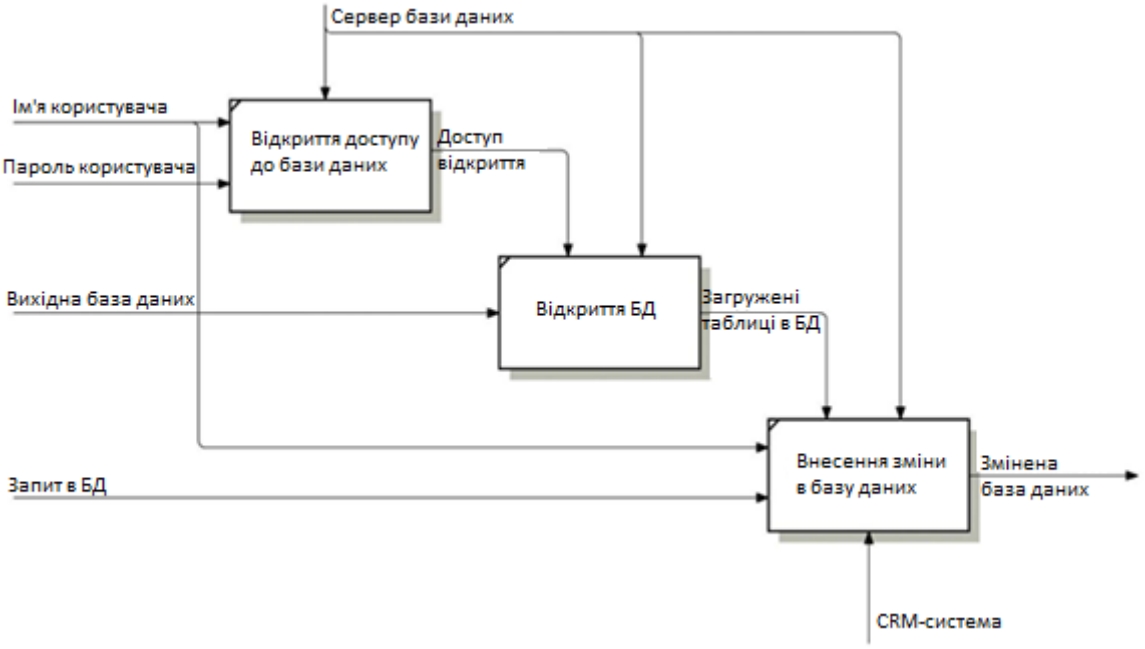


Рисунок 2.6 – Декомпозиція бізнес-функції «Вивчити історію взаємовідношень з клієнтами»

Завдання до практичної роботи:

1. На основі аналізу предметної області створити модель структури системи, шляхом декомпозиції її на підсистеми (побудувати діаграми рис. 2.3-2.6).
2. Виконати декомпозицію бізнес-функції відповідно до варіанту (табл. 2.1)
3. Оформити звіт про виконану роботу, який має містити хід виконання роботи та висновки.

Таблиця 2.1 − Варіанти завдань

Номер варіанту	Бізнес-функція для декомпозиції	Номер варіанту	Бізнес-функція для декомпозиції
1	Управління задачами	16	Перевірити наявність карточки клієнта в системі
2	Управління нагадування завдань	17	Заповнити карточку клієнта
3	Аналітика продаж	18	Перевірити статус клієнта в системі
4	Перевірити наявність карточки клієнта в системі	19	Управління задачами
5	Заповнити карточку клієнта	20	Управління нагадування завдань
6	Перевірити статус клієнта в системі	21	Аналітика продаж
7	Управління задачами	22	Перевірити наявність карточки клієнта в системі
8	Управління нагадування завдань	23	Заповнити карточку клієнта
9	Аналітика продаж	24	Перевірити статус клієнта в системі

10	Перевірити наявність карточки клієнта в системі	25	Управління задачами
11	Заповнити карточку клієнта	26	Управління нагадування завдань
12	Перевірити статус клієнта в системі	27	Аналітика продаж
13	Управління задачами	28	Перевірити наявність карточки клієнта в системі
14	Управління нагадування завдань	29	Заповнити карточку клієнта
15	Аналітика продаж	30	Перевірити статус клієнта в системі

Контрольні питання

1. Що таке модель та які рівні деталізації вона може мати?
2. Чим відрізняються поняття бізнес-процес, процедура та операція?
3. Що таке декомпозиція бізнес-процесу і для чого вона використовується?
4. Які основні властивості характеризують систему?
5. У чому полягає принцип «розподіляй і володарюй»?
6. Що означає принцип ієрархічного впорядкування?
7. Яке значення мають принципи абстрагування, формалізації, несуперечності та структуризації даних?
8. Які основні види діаграм використовують у структурному аналізі (SADT, DFD, ERD)?
9. Що таке контекстна діаграма?
10. Яке призначення контекстної діаграми?
11. Що таке діаграма декомпозиції?
12. Яке призначення діаграми декомпозиції?

Практичне заняття 3

Побудова UML діаграм використання.

Мета заняття: Навчити будувати UML діаграми використання, розробляти сценарії варіантів використання та перевіряти їх якість.

Зміст заняття: Побудувати UML-діаграму використання обраної програмної системи та розробити сценарії її варіантів використання, виконати перевірку якості запису сценаріїв варіантів використання.

Теоретичні основи

Моделювання вимог до системи за допомогою діаграми варіантів використання (прецедентів) виконується в такий спосіб:

1. Встановити контекст системи, ідентифікувавши акторів, які її оточують.
2. Для кожного актора розглянути поведінку, яку він очікує або потребує від системи.
3. Надати ім’я цим загальним варіантам поведінки як прецедентам.
4. Виокремити загальну поведінку до нових прецедентів, які будуть використовуватися іншими; виокремити варіації поведінки до нових прецедентів, які поширюють основні потоки подій.

5. Змоделювати ці прецеденти, акторів та відношення між ними на діаграмі прецедентів.

6. Доповнити прецеденти примітками, що описують не функціональні вимоги.

Щоб було зрозуміло зміст кожного з варіантів використання, слід записати сценарій його виконання. Для цього можна скористатися шаблоном, який надається далі, або шаблоном (таблиця 3.1).

Прецедент 1: Ім’я варіанта використання
Актори:
Мета:
Короткий опис:
Передумова:
Основний (успішний, типовий) сценарій:
1. основний етап 1– перехід до альтернативи А1, до помилки Е1
2. основний етап 2
3. основний етап 3
Набір альтернативних сценаріїв:
Альтернативний сценарій А1.
Альтернативний сценарій А2.
Альтернативний сценарій А3.
Набір помилок:
Помилка 1
Помилка 2
Помилка 3
Результат (післяумова)

Таблиця 3.1 – Шаблон для написання сценарію окремого варіанта використання

Головний розділ	Розділ «Типовий хід подій»	Розділ «Виключення»	Розділ «Примітки»
Ім’я варіанта використання	Типовий хід подій, що призводять до успішного виконання варіанта використання	Виключення 1	Примітка 1
		Виключення 2	Примітка 2
Актори		…	…
Мета			
Короткий опис			
Рівень цілі		Виключення *N*	Примітка *M*
Посилання на інші варіанти використання			

Для перевірки якості сценаріїв варіантів використання можна скористатися з тестів «пройшов/не пройшов» (таблиця 3.2). На питання тесту відповіді для якісного сценарію мають бути «так».

Таблиця 3.2 – Тести «пройшов/не пройшов» для варіантів використання

Поле	Питання
Назва варіанта використання	1. Чи є речення, що називає мету основної діючої особи, реченням з дієсловом дією?
	1. Може система задовольнити цю мету?
Короткий опис і рівень цілі	2. Чи заповнені поля?
Короткий опис	3. Чи розглядає варіант використання згадану в розділі Короткий опис систему як «чорний ящик»?

	4. Якщо система в розділі Короткий опис є тією, що повинна розроблятися, чи мають розробники створювати тільки те, що є в ній, і нічого, що до неї не входить?
Рівень цілі	5. Чи відповідає зміст варіанта використання заявленому рівню цілі?
	6. Чи дійсно мета перебуває на заявленому рівні цілі?
Основна діюча особа	7. Чи притаманна актору поведінка?
	8. Чи має актор мету відносно розроблювальної системи, що повинна бути послугою розроблюваної системи?
Основний сценарій	9. В ньому 3-9 кроків?
	10. Чи виконується він від тригера (умови входу) до забезпечення гарантії успіху (умови виходу)?
	11. Чи допускає він правильні зміни в послідовності?
Кожен крок у будь-якому сценарії	12. Він описує досягнення мети?
	13. Чи помітно просуває процес його успішне виконання?
	14. Чи ясно, яка діюча особа досягає мети, хто «ударяє по м'ячі»?
	15. Рівень мети кроку нижче за рівень мети варіанта використання в цілому? Він трохи нижчий (що переважніше) за рівень мети варіанта використання?
	16. Ви впевнені, що крок не описує проект інтерфейсу користувача?
	17. Чи ясно, яка інформація передається на кроці?
Загальний зміст варіанта використання	18. Замовникам і користувачам: «Це те, що вам потрібно?»
	19. Замовникам і користувачам: «Ви зможете сказати по завершенню реалізації, що одержали те, що хотіли?»
	20. Розробникам: «Ви можете це реалізувати?»

Візуальне моделювання в UML можна уявити як певний процес поступового спуску від найбільш загальної і абстрактної концептуальної моделі вихідної системи до логічної, а потім і до фізичної моделі відповідної програмної системи. Для досягнення цих цілей спочатку будується модель у формі так званої діаграми варіантів використання (use case diagram), яка описує функціональне призначення системи або, іншими словами, те, що система буде робити в процесі свого функціонування. Діаграма варіантів використання є вихідним концептуальним поданням або концептуальною моделлю системи в процесі її проектування і розробки.

Розробка діаграми варіантів використання переслідує мету:

- Сформулювати загальні вимоги до функціонального поведінки проектованої 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.

3.1 Склад діаграми Use Case

Діаграма варіантів використання складається з акторів, для яких система виробляє дію і власне дії Use Case, яка описує те, що актор хоче отримати від системи. Актор позначається у вигляді чоловічка, а Use Case - овалом. Додатково в діаграми можуть бути додані коментарі.

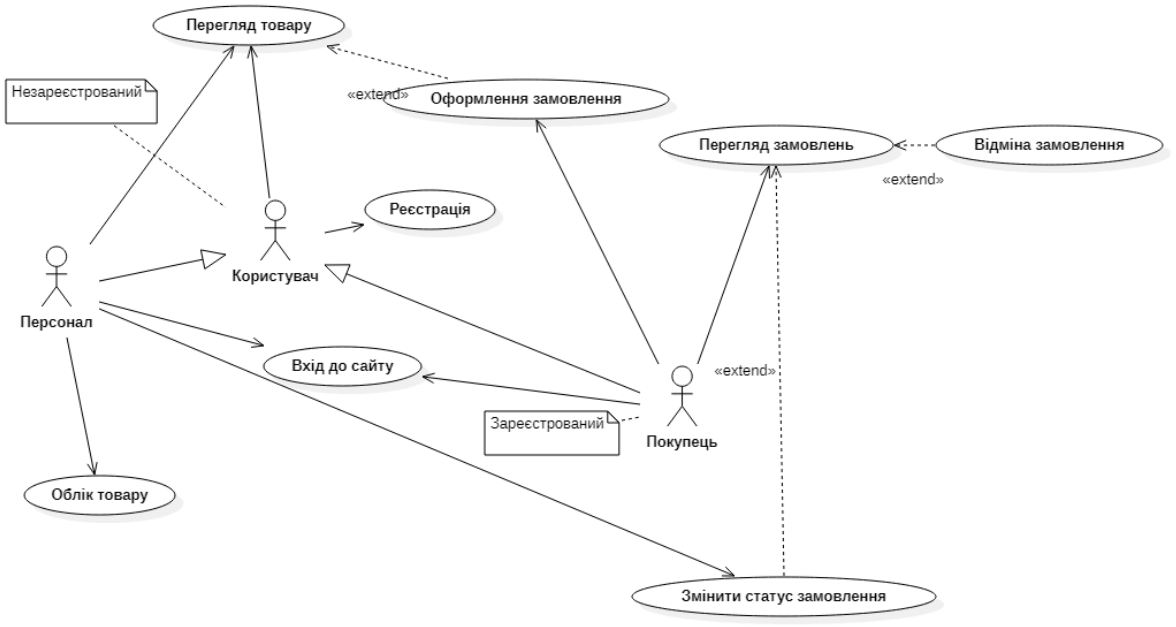


Рисунок 3.1 – Приклад діаграми Use Case

Між акторами і варіантами використання можуть бути різні види взаємодії. Основні види взаємодії наступні:

Проста асоціація - відбивається лінією між актором і варіантом використання (без стрілки). Відображає зв'язок актора і варіанти використання. На малюнку між актором адміністратор і варіантом використання переглядати замовлення.

Спрямована асоціація - то ж що і проста асоціація, але показує, що варіант використання ініціалізується актором. Позначається стрілкою.

Спадкування - показує, що нащадок успадкує атрибути і поведінку свого прямого предка. Може застосовуватися як для акторів, так для варіантів використання.

Розширення (extend) - показує, що варіант використання розширює базову послідовність дій і вставляє власну послідовність. При цьому дана лінія позначається стереотипом <<extend>>.

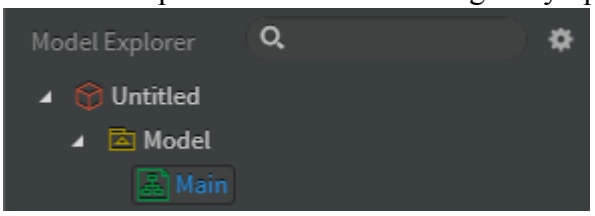
Включення - показує, що варіант використання включається в базову послідовність і виконується завжди (на малюнку не показаний). показувала на те, що кожен екземпляр першого варіанту використання завжди включає в себе функціональне поведінка або виконання другого варіанту використання. У цьому сенсі поведінка другого варіанту використання є частиною поведінки першого варіанту використання на даній діаграмі. Графічно дане відношення позначається як відношення залежності у формі пунктирною лінії зі стрілкою, спрямованої від базового варіанту використання до такого, що включається варіанту використання. При цьому дана лінія позначається стереотипом << include >>.

3.2 Розробка діаграм варіантів використання в середовищі StarUML

Робота над моделлю в середовищі StarUML починається із загального аналізу проблеми та побудови діаграми варіантів використання, яка віддзеркалює функціональне призначення проектованої програмної системи. Для нового проекту можна скористатися майстром типових проектів. Майстер типових проектів доступний з меню File→New або при завантаженні програми StarUML. У випадку розробки проекту, для якого не відома або не обрана технологія

його реалізації, слід скористатися шаблоном Default Approach, як результат чого з'явиться робочий інтерфейс програми StarUML із чистим вікном активної діаграми класів і іменем проекту untitled як усталено. Для зміни імені проекту, запропонованого програмою як усталено, слід зберегти модель у зовнішньому файлі на диску (файл буде мати тип uml).

Для розробки діаграми варіантів використання моделі в середовищі StarUML необхідно активізувати відповідну діаграму у вікні діаграми. Це можна зробити такими способами:

1. розкрити вид варіантів використання Use Case Diagram у браузері діаграм та двічі клацнути на піктограмі Main 

;
2. за допомогою операції головного меню Model → Add →Use Case Diagram (рис. 3.2).

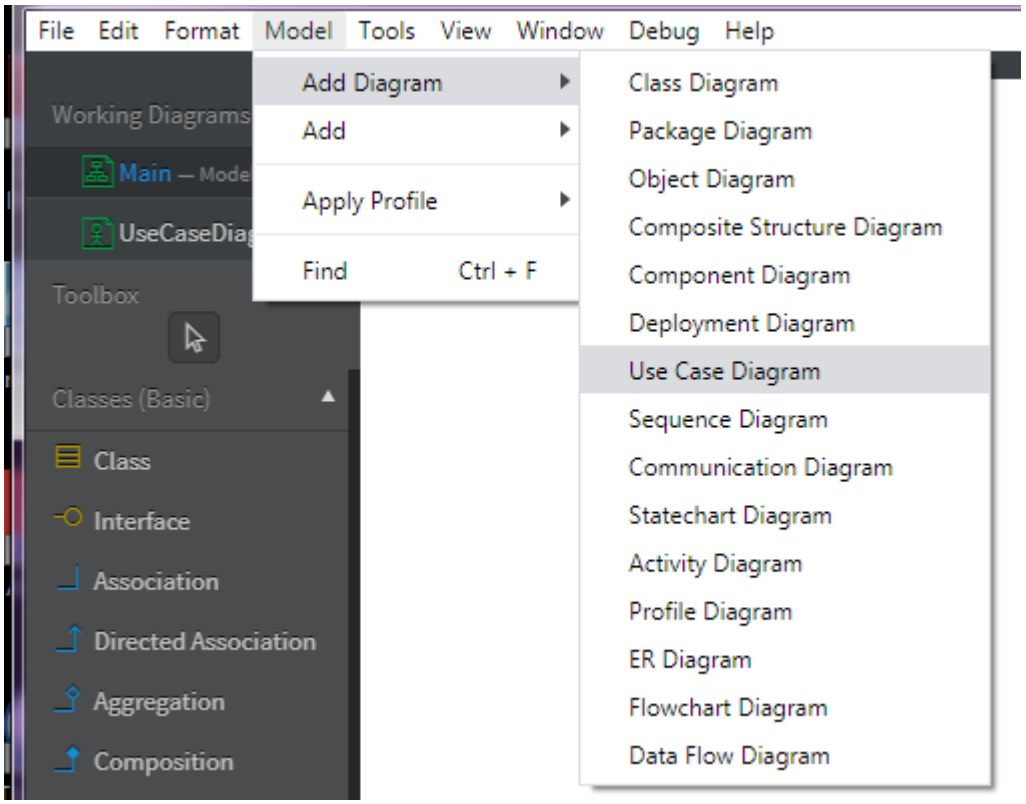
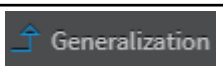


Рисунок 3.2 – Способ створення діаграми Use Case

При цьому з'являється нове вікно із чистим робочим аркушем діаграми варіантів використання та спеціальна панель інструментів, яка містить кнопки із зображенням графічних елементів, необхідних для розробки діаграми варіантів використання, з зазначенням їх призначення (табл.3.3).

Таблиця 3.3 – Призначення кнопок спеціальної панелі інструментів для діаграми варіантів використання

Зображення кнопки	Призначення кнопки
	Перетворює зображення курсору у форму стрілки для наступного виділення елементів на діаграмі
	Додає на діаграму пакет
	Додає на діаграму варіант використання
	Додає на діаграму актора
	Додає на діаграму неспрямовану асоціацію
	Додає на діаграму спрямовану асоціацію
	Додає на діаграму відношення узагальнення
	Додає на діаграму відношення залежності
	Додає на діаграму відношення включення
	Додає на діаграму відношення розширення

3.2.1 Додавання актора на діаграму варіантів використання та редагування його властивостей

Для додавання актора на діаграму варіанта використання потрібно за допомогою лівої кнопки миші нажати кнопку із зображенням піктограми актора  на спеціальній панелі інструментів, відпустити ліву кнопку миші та клацнути лівою кнопкою миші на вільному місці робочого аркуша діаграми. На діаграмі з'явиться зображення актора з маркерами зміни його геометричних розмірів і запропонованим програмою іменем (рис.3.3).

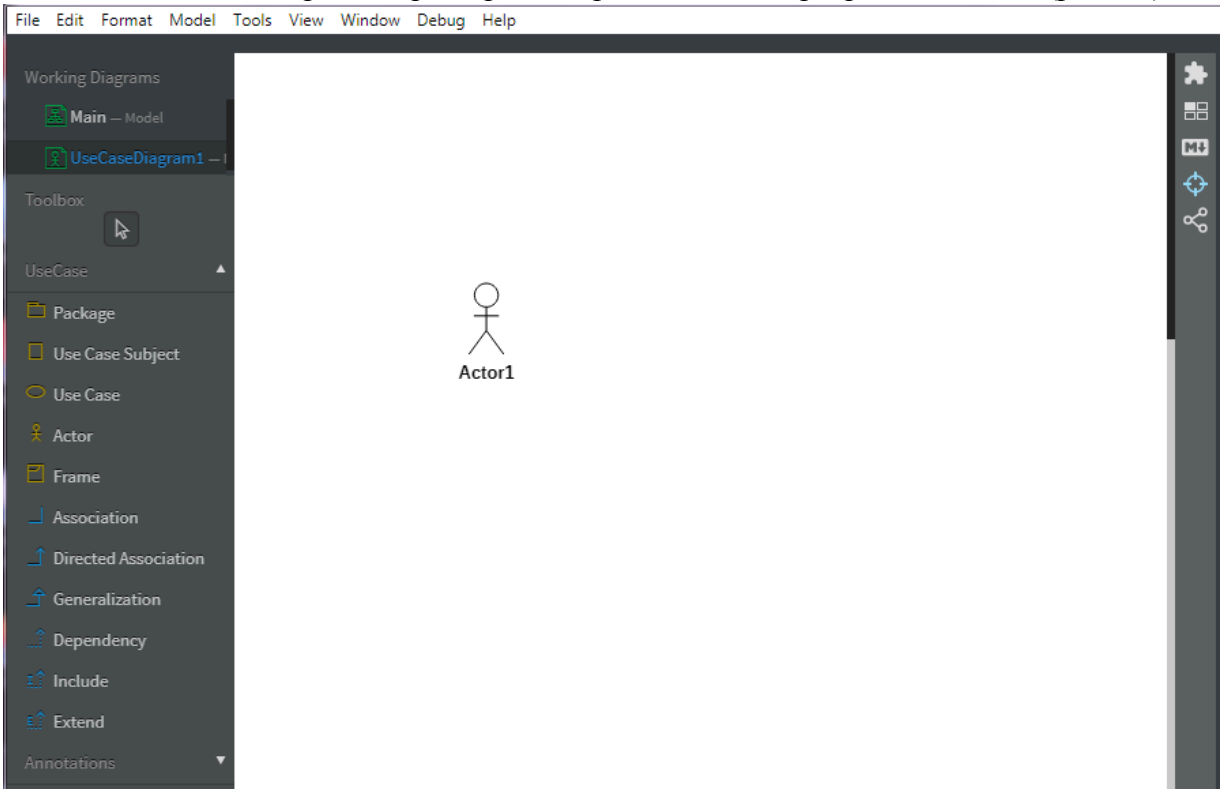


Рисунок 3.3 – Діаграма варіантів використання після додання на неї актора

Щоб змінити розташування зображення графічного елемента моделі, треба клацанням лівої кнопки миші виділити його на робочій ділянці діаграми, і, не відпускаючи лівої кнопки, перемістити на потрібне місце діаграми. При цьому виділений елемент візуально відрізняється від інших наявністю маркерів зміни його геометричних розмірів у формі невеликих білих квадратів.

Щоб змінити графічні розміри зображення елемента моделі, насамперед, треба клацанням лівої кнопки миші виділити його в робочій ділянці діаграми. Як результат цього з'явиться прямокутник, який зображує границі обраного геометричного елемента. Після чого, не відпускаючи лівої кнопки миші, слід змінити розміри цього прямокутника як потрібно.

Ім'я приміщеного на діаграму елемента розробник може змінити або відразу після додавання елемента на діаграму, або в ході подальшої роботи над проектом. Для цього слід вийти в режим редагування властивостей елементу двічі клацнувши лівою кнопкою миші по відповідному елементу, або скористатися з можливостей вікна редагування властивостей елементу.

3.2.2 Додавання та редагування варіанта використання

Для додавання варіанта використання на діаграму потрібно за допомогою лівої кнопки миші нажати кнопку із зображенням варіанта використання на спеціальній панелі інструментів, відпустити ліву кнопку миші та клацнути лівою кнопкою миші на вільному місці діаграми. На діаграмі з'явиться зображення варіанта використання з маркерами зміни його геометричних розмірів і запропонованим програмою іменем (рис.3.4).

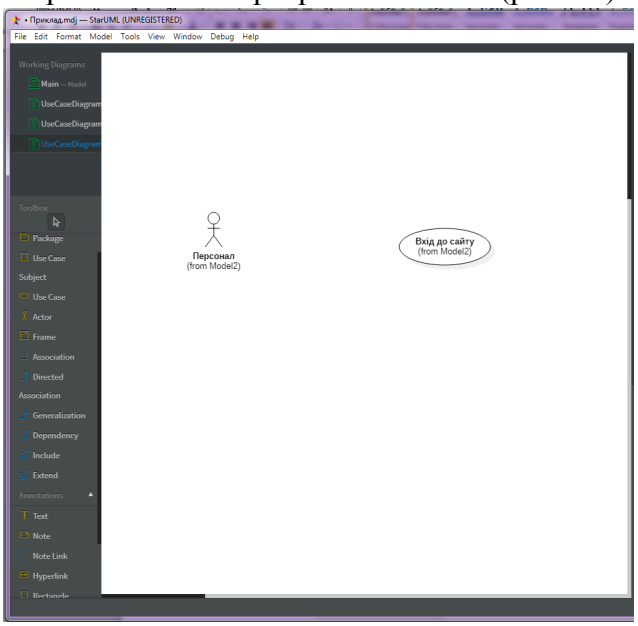


Рисунок 3.4 – Діаграма варіантів використання після додавання на неї варіанта використання

Для уточнення властивостей варіанта використання слід вийти в режим редагування властивостей елементу, двічі клацнувши лівою кнопкою миші по відповідному елементу, або скористатися з можливостей вікна редагування властивостей цього елементу.

3.2.3 Додавання асоціації

Для додавання асоціації між актором і варіантом використання на діаграму потрібно за допомогою лівої кнопки миші нажати на спеціальній панелі інструментів кнопку із зображенням піктограми спрямованої (неспрямованої) асоціації 

, відпустити ліву кнопку миші, клацнути лівою кнопкою миші на зображенні актора на діаграмі та відпустити її

на зображенні варіанта використання. Як результаті цих дій на діаграмі з'явиться зображення асоціації, яка з'єднує актора з варіантом використання (рис. 3.5).

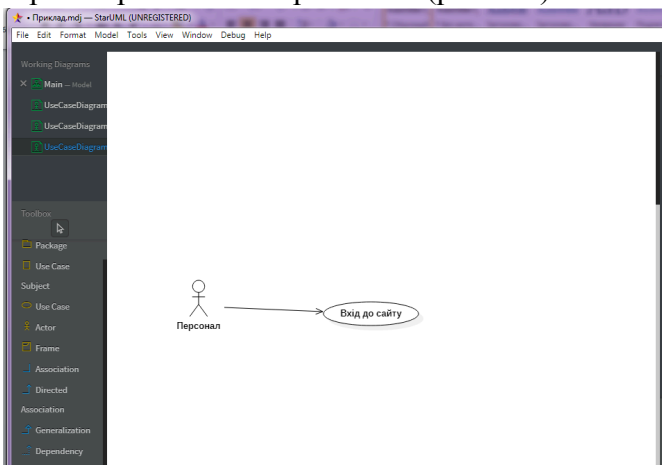
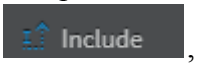


Рисунок 3.5 – Діаграма варіантів використання після додавання на неї спрямованої асоціації

При необхідності можна зробити спрямовану асоціацію неспрямованою, для чого слід скористатися вікном властивостей асоціації.

Якщо відомо, що асоціація неспрямована, то можна відразу додати на діаграму неспрямовану асоціацію за допомогою відповідної кнопки панелі інструментів.

3.2.4 Додавання відносини включення, розширення

Для додавання відносини **включення, розширення** між двома варіантами використання на діаграму необхідно за допомогою лівої кнопки миші нажати кнопку із зображенням піктограми залежності потрібного типу на спеціальній панелі інструментів 

, 

відповідно, відпустити ліву кнопку миші, клацнути лівою кнопкою миші на зображенні одного варіанта використання, який бере участь в залежності, та відпустити її на зображенні другого варіанта використання. Як результаті цих дій на діаграмі з'явиться зображення відповідного відношення, яке поєднує два обраних варіанти використання.

Текст стереотипу відношення в кутових дужках розташовується поруч із зображенням пунктирної лінії залежності, яка зв'язує відповідні варіанти використання (рис.3.6). З метою кращої візуалізації діаграми текстову область стереотипу можна перемістити в потрібне місце діаграми. Виконати це можна за допомогою загальноприйнятого способу переміщення графічних елементів моделі.

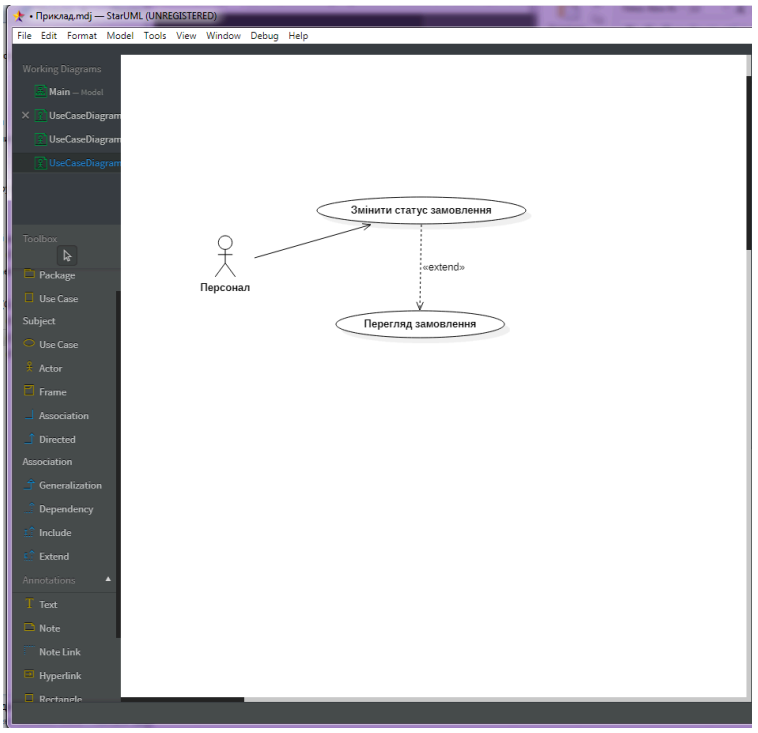


Рисунок 3.6 −Діаграма варіантів використання після додавання на неї відношення залежності

Нагадаємо, що діаграма варіантів використання є високорівневим концептуальним наданням моделі, тому вона не повинна містити занадто багато варіантів використання та акторів. Діаграма може бути змінена в будь-який момент за допомогою додавання нових елементів, таких як варіанти використання та актори, або їхнього видалення.

Для видалення будь-якого графічного елемента з діаграми його слід виділити на діаграмі та нажати клавішу Delete на клавіатурі. При цьому виділений елемент буде вилучений з активної діаграми, але не з моделі. Для видалення елемента не тільки з діаграми, а й з моделі проекту необхідно виділити на діаграмі елемент, що видаляється, та скористатися операцією головного меню Edit → Delete from Model.

При роботі з відношеннями на діаграмі варіантів використання слід пам'ятати про призначення відповідних відношень в нотації UML. Якщо для двох елементів обраний вид відношення не є припустимим, то в більшості випадків програма StarUML сповістить про це розробника, і відповідна лінія зв'язку не буде додана на діаграму.

Після закінчення сеансу роботи над проектом виконану роботу необхідно зберегти у файлі проекту з розширенням .uml. Це можна зробити через меню File → Save або File → Save As. При цьому вся інформація про проект, зокрема діаграми та специфікації елементів, буде збережена в одному файлі.

3.3 Приклади сценаріїв варіантів використання (прецедентів)

Надамо приклад сценаріїв варіантів використання для диаграми Use Case (рис. 3.2).

Прецедент 1: Вхід до сайту

Актори: Покупець, Персонал.

Мета: Отримати доступ до можливостей інтернет-магазину баскетбольного інвентарю.

Передумова: Користувач бажає увійти на сайт.

Основний (успішний) сценарій:

1. Користувач визначає спосіб входу на сайт. Якщо він вже зареєстрований, йому потрібно авторизуватись – перехід д п. 8.

2. Для реєстрації на сайті інтернет-магазину користувач натискає кнопку «Реєстрація».

3. Система надає форму для вводу e-mail, який використовується у якості логіну, та паролю.

4. Користувач вводить затребувані дані.

5. Система визначає роль користувача. Для цього з БД завантажується файл з e-mail працівників, що забезпечують роботу сайту. Якщо e-mail користувача співпадає з записом зі списку e-mail працівників, то система вважає, що реєструється Персонал, інакше – Покупець.

6. Система перевіряє введений пароль та визначає чи він є припустимим, тобто не коротше ніж 8 символів та містить лише букви та цифри.

7. Система зберігає реєстраційні дані Персонал чи Покупець у БД.

8. Система надає форму авторизації, що потребує ввід e-mail, який використовується у якості логіну, та паролю.

9. Користувач вводить авторизаційні дані.

10. Система виконує пошук введених даних у БД та авторизує користувача як Персонал чи Покупець.

11. Якщо користувач є Персонал, йому надаються права для роботи з баскетбольними товарами.

12. Якщо користувач є Покупець, система надає йому доступ до особистого кабінету.

13. Завершення прецеденту.

Набір альтернативних сценаріїв:

Альтернативний сценарій 1.

4a. Користувач вводить неповні дані.

4.1a. Система перевіряє повноту даних, визначає, що вони цілком чи частково відсутні, та виводить повідомлення.

Альтернативний сценарій 2.

5a. Система не може отримати доступ до БД для отримання списку e-mail працівників.

5.1a. Система повідомляє про помилку при завантаженні даних з БД.

Альтернативний сценарій 3.

6a. Введений пароль не є припустимим.

6.1a. Система повідомляє про помилку у вводі даних.

Альтернативний сценарій 4.

7a. Система не може отримати доступ до БД для збереження даних.

7.1a. Система повторно намагається звернутись до БД.

Результат: Користувач отримав доступ до функціоналу інтернет-магазину з продажу баскетбольного інвентарю.

Прецедент 2: Перегляд товару

Актори: Покупець, Персонал

Мета:

Отримати доступ до переліку товарів інтернет-магазину баскетбольного інвентарю.

Передумова: Користувач авторизований на сайті як Покупець чи Персонал.

Основний (успішний) сценарій:

1. Користувач переходить на головну сторінку інтернет-магазину.

2. Користувач натискає кнопку "Товари" на головній сторінці або вибирає відповідний пункт меню.

3. Система виконує запит до бази даних і відображає перелік доступних товарів (Назву товару, Опис товару, Ціну, Наявність на складі, Зображення товару)

4. Користувач може переглядати список товарів, прокручуючи його вниз або використовуючи фільтри для звуження вибору (за категорією, ціною тощо).

5. Користувач може натиснути на конкретний товар для перегляду детальної інформації, включаючи:

- Розширений опис
- Характеристики
- Відгуки інших покупців

6. Користувач має можливість повернутися до загального списку товарів після перегляду детальної інформації.

Набір альтернативних сценаріїв:

Альтернативний сценарій 1:

- 3а. Система не може отримати доступ до бази даних для завантаження товарів.
- 3.1а. Система повідомляє про помилку при завантаженні даних.

Альтернативний сценарій 2:

- 5а. Користувач натискає на товар, але система не може відобразити детальну інформацію.
- 5.1а. Система повідомляє про помилку при завантаженні даних товару.

Результат (післяумова): Користувач (Персонал чи Покупець) отримав доступ до переліку товарів

Прецедент 3: Облік товарів

Актори: Персонал

Мета: Персонал має можливість додавати, редагувати та видаляти товари в інтернет-магазині баскетбольного інвентарю.

Передумова: Користувач авторизований на сайті як Персонал.

Основний (успішний) сценарій:

1. Персонал відкриває інтерфейс управління товарами.
2. Персонал натискає на кнопку "Управління товарами" або вибирає цей пункт у меню.
3. Система відображає список доступних товарів у вигляді таблиці, де кожен товар представлений такими даними: Назва товару, Кількість на складі, Ціна, Категорія товару
4. Персонал має можливість вибрати одну з дій:
   - Додати новий товар
   - Редагувати існуючий товар
   - Видалити товар
5. Якщо Персонал натискає кнопку "Додати товар":
   - Система надає форму для введення даних нового товару (назва, опис, ціна, кількість, категорія).
   - Персонал заповнює всі необхідні поля та натискає кнопку "Зберегти".
   - Система зберігає новий товар у базі даних.
6. Якщо Персонал натискає кнопку "Редагувати" для обраного товару:
   - Система надає форму з попередньо заповненими даними товару.
   - Персонал вносить зміни в поля (наприклад, змінює кількість або ціну товару) та натискає "Зберегти".
   - Система оновлює дані товару в базі даних.
7. Якщо Персонал натискає кнопку "Видалити" для обраного товару:
   - Система запитує підтвердження на видалення товару.

- Персонал підтверджує видалення.
- Система видаляє товар з бази даних.

Набір альтернативних сценаріїв:
Альтернативний сценарій 1:

- 5а. Персонал не заповнив всі необхідні поля під час додавання товару.
- 5.1а. Система повідомляє про необхідність заповнити всі обов'язкові поля.

Альтернативний сценарій 2:

- 6а. Під час редагування товару в базі даних сталася помилка.
- 6.1а. Система повідомляє про помилку при збереженні змін і пропонує повторити спробу.

Альтернативний сценарій 3:

- 7а. Під час видалення товару система не може отримати доступ до бази даних.
- 7.1а. Система повідомляє про помилку і пропонує повторити спробу.

Результат (післяумова):
Персонал успішно додав, відредагував або видалив товари в системі, і дані оновилися в базі інтернет-магазину.

Завдання до практичної роботи:

1. Побудувати діаграму варіантів використання для CRM-системи у StarUML.
2. Записати сценарії варіантів використання CRM-системи.
3. Додати короткий опис актора у StarUML.
4. Додати короткий опис до варіантів використання у StarUML..
5. Виконати перевірку якості запису сценаріїв варіантів використання.
6. Оформити звіт, який має містити:
   - побудовану діаграму варіантів використання для CRM-системи;
   - опис сценаріїв кожного варіанта використання CRM-системи;
   - результат перевірки якості сценаріїв варіантів використання, скориставшись тестами (табл.3.2);
   - висновки.

Контрольні питання

1. Як запустити програму StarUML?
2. З яких компонент складається вікно StarUML?
3. Варіанти використання. Визначення. Позначення. Як нанести варіант використання на діаграму використання?
4. Актори. Позначення. Що є важливим для актора? Як нанести актора на діаграму використання?
5. Діаграми варіантів використання. Зв’язки на діаграмах варіантів використання.
6. Як нанести зв’язки на діаграму варіантів використання?
7. Як додати опис до дійової осіби?
8. Як додати опис до варіантів використання?

Практичне заняття 4

Створення діаграм робастності (Robustness diagram)

Мета заняття: Навчитися виявляти об’єкти системи для аналізу та виявлення недоліків

у розподілі функціональності між компонентами системи на основі сценаріїв варіантів використання.

Зміст заняття: Побудувати діаграми робастності обраної програмної системи на основі сценаріїв варіантів використання та проаналізувати отримані діаграми на предмет виявлення можливих недоліків у структурі системи.

Теоретичні основи

Після створення діаграми використання і створення сценаріїв прецедентів, перед програмістами постає питання вивчення тексту прецеденту і виявлення набору об'єктів, які братимуть участь в реалізації даного прецеденту з подальшою класифікацією цих об'єктів на основі їх характеристик. Для вирішення цього питання використовують діаграми робастності (Robustness diagram).

Діаграма робастності відображає об'єкти, що беруть участь у сценарії, та їхню взаємодію, а саме:

- Актор (actor) - зовнішній (до системи) суб'єкт/об'єкт, що бере участь у сценарії; відображає відповідного актора із діаграми прецедентів.

– Граничний об'єкт (boundary object) – об'єкт на межі системи із зовнішнім середовищем; зазвичай є об'єктом, що використовується актором при взаємодії із системою.

- Об'єкт-сутність (entity object) - інформаційний об'єкт системи; є об'єктом із моделі предметної області.

- Об'єкт-управління (control object) - керуючий об'єкт системи; є «з'єднувачем» граничного об'єкта та об'єкта-сутності; зазвичай відображає деяку функцію, необхідну для виконання прецеденту.

Приклад граничних об'єктів: елементи інтерфейсу користувача (вікна, екрани, діалоги, меню). Приклади об'єктів-сутностей: таблиці баз даних, файли з інформацією тривалого зберігання, результати пошуку.

Об'єкти-управління реалізують прикладну логіку системи, відокремлюючи її одночасно від граничних об'єктів та від об'єктів-сутностей. Насправді об'єкти-управління рідко реалізуються як об'єкти, вони зазвичай програмуються як методів класів. Тому об'єкт управління називають також контролером (controller).

Діаграми варіантів використання (прецедентів) зазвичай відображають основний потік, діаграми робастності дозволяють відобразити одночасно альтернативні потоки і тому зазвичай будуються за повними текстовими описами сценаріїв прецедентів.

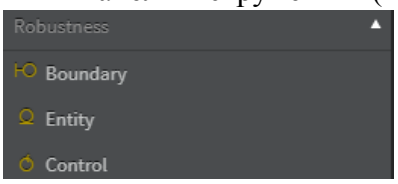
Побудова діаграм робастності виконується відповідно до простих правил:

1. Актори пов'язані лише з граничними об'єктами.
2. Граничні об'єкти пов'язані лише з акторами та об'єктам-управління.
3. Об'єкти-сутності пов'язані лише з об'єктам-управління.
4. Об'єкти-управління (Контролери) пов'язані з граничними об'єктами, об'єктами-сутностями та іншими контролерами, але з акторами не пов'язані.

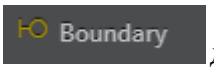
Використання діаграм робастності дозволяє уточнити діаграму класів для моделі предметної області та більш точно побудувати діаграми взаємодії.

Для того, щоб створити діаграму робастності виконайте наступні дії:

1.1. Створіть нову діаграму класів. Надайте ім’я діаграмі класів за допомогою ModelExplorer.

1.2. В нижній частині Панелі інструментів (Toolbox) розташовані елементи діаграми робастності (Robustness) 

.

2.1. На панелі інструментів натисніть кнопку 

для створення граничних об’єктів, на кнопку 

- для створення об’єктів сутностей, 

- ддя створення об’єктів, що управляють.

2.2. Нанесіть на діаграму усі об'єкти.

3. Для нанесення зв’язків між об’єктами виконайте наступні дії:

3.1. На панелі інструментів натисніть необхідну кнопку, яка відображає зв'язок між об’єктами

, .

Приклад. Командою розробників було визначено класи аналізу: граничні класи, класи сутності і класи, що управляють для прецеденту "Реєстрація".

Основний (успішний) сценарій:

1. Для реєстрації на сайті інтернет-магазину користувач натискає кнопку «Реєстрація», що знаходиться на початковій сторінці.

3. Система відображає сторінку реєстрації.

4. Користувач вводить свій ідентифікатор і пароль, після цього клацає на кнопці ОК.

5. Система визначає роль користувача. Для цього система порівнює введену інформацію з даними облікового запису персоналу і повертає користувача на початкову сторінку. Якщо дані користувача співпадають з даними облікового запису персоналу, то система вважає, що реєструється Персонал, інакше – Покупець.

6. Система зберігає реєстраційні дані Персонал чи Покупець у БД.

7. Завершення прецеденту.

Готова діаграма кооперації повинна виглядати як на рисунку 4.1. Це тільки одна з діаграм, необхідних для моделювання варіанту використання "Реєстрація". Вона відповідає успішному сценарію. Для опису того, що трапиться, якщо виникне помилка, або якщо користувач вибере інші дії із запропонованих, доведеться розробити інші діаграми. Кожен альтернативний сценарій варіанта використання може бути промоделирован за допомогою окремих діаграм робастності.

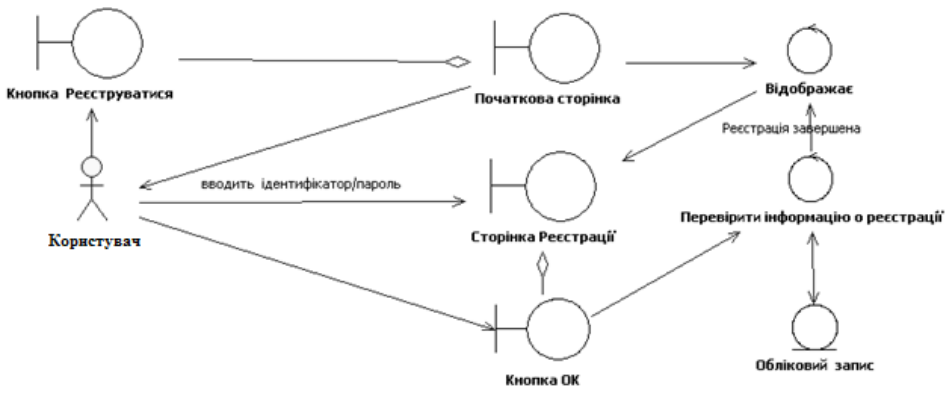


Рисунок 4.1 −Діаграма робастності для прецеденту "Реєстрація".

Завдання до практичної роботи:

1. Ознайомитись з теоретичним матеріалом – вивчити поняття діаграм робастност, їх призначення та правил побудови.
2. Для кожного основного сценарію до прецедентів діаграми варіантів використання для CRM-системи у StarUML побудувати діаграми робастності (з тексту сценарію виділити всі об’єкти, що братимуть участь у процесі; класифікувати їх як граничні об’єкти, об’єкти-сутності або об’єкти-управління; добавити зв’язки між об’єктами).
3. Перевірити відповідність кожної створеної діаграми робастності сценарію варіанта використання.
4. Оформити звіт, який має містити:

− побудовані діаграми робастності для основних сценаріїв до прецедентів діаграми варіантів використання для CRM-системи;

− висновки.

Контрольні питання:

1. Для чого створюють у системі діаграму робастності?
2. Як створити діаграму робастност у StarUML?
3. Як додати об'єкти на діаграму робастності у StarUML?
4. Як додати зв’язки на діаграму робастності у StarUML?
5. Що таке об’єкт-сутність? Які дані він відображає?
6. Яка роль об’єкта-управління (контролера) у діаграмі робастності?
7. Які правила зв’язків між об’єктами необхідно враховувати при побудові діаграми робастності?
8. Як можна використовувати діаграми робастності для валідації вимог до системи?

Практичне заняття 5

Побудова UML діаграми класів.

Мета заняття: Навчитися виявляти елементи діаграм класів, їх властивості. Вивчення типів відношень між класами та здобуття навичок побудови та узгодження діаграм класів.

Зміст заняття: Визначення класів та їхніх характеристик на основі предметної області та сценаріїв використання; побудова UML-діаграми класів для прикладу програмної системи; узгодження діаграми класів із діаграмами використання та робастност; аналіз побудованої діаграми та виправлення можливих помилок.

Теоретичні основи

Моделювання класів – процес не детермінований, не існує єдиної методики його виконання. Підхід на основі використання іменних груп припускає, що аналітик шукає іменні групи в описі вимог. Кожен іменник є потенційним класом. Потім отриманий список класів розділяють на три групи:

1. Релевантні класи – це класи, які безумовно належать до проблемної області.
2. Нечіткі класи – це класи, які неможна впевнено та безсумнівно визнати релевантними.
3. Нерелевантні класи – це класи, які виходять за межі проблемної області.

Керівні принципи для визначення класів:

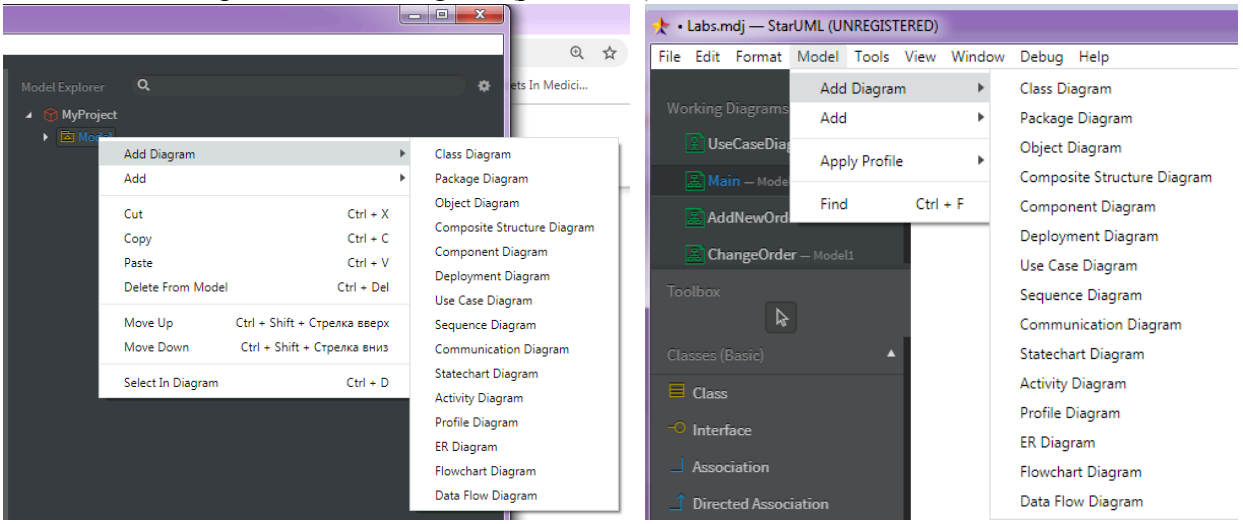
1. Для кожного класу має бути ясно сформульовано його призначення в системі.
2. Кожен клас – це шаблон для опису множини об’єктів. Одиничні класи, для яких можна уявити існування лише одного об’єкта, малоймовірні серед «бізнес-об’єктів».
3. Кожен клас має містити набор атрибутів.
4. Клас має відрізнятися від атрибуту. Але визначення того, чим є поняття (класом чи атрибутом) залежить від предметної області застосунку.
5. Клас має містити набор операцій.

Знаходження основних асоціацій є побічним ефектом процесу виявлення класів. Під час виконання пробного прогону варіантів використання з’ясовують шляхи взаємодії між класами. Ці взаємодії підтримують асоціації.

Хід виконання роботи:

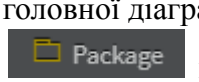
5.1 Створити головну діаграму класів.

5.1.1 Клацніть на піктограмі Model, з контекстного меню обрати Add Diagram і далі вибираємо Class Diagram (рис. 5.1 а). Або обираємо пункт меню Model, далі Add Diagram і далі вибираємо Class Diagram (рис. 5.1 б).



а) б)

Рисунок 5.1 – Створення головної діаграми класів.

5.1.2 За допомогою кнопки Package (Пакет) 

панелі інструментів помістите на діаграму новий пакет. У вкладці Властивості (Properties) вкажіть ім'я пакету Entities (Сутність).

5.1.3 Створіть акети Boundaries (Граничні) і Control (Управління). Головна діаграма Класів повинна виглядати як на рисунку 5.2.

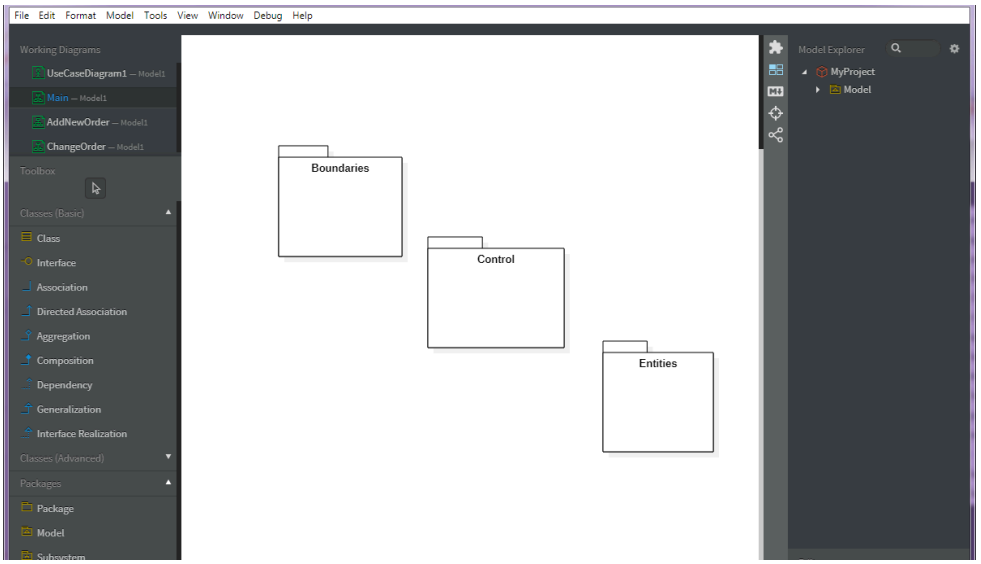


Рисунок 5.2 _ Головна діаграма Класів системи обробки замовлень.

5.2 Приклад створення діаграми Класів для прецеденту "Реєстрація" зі всіма класами.

Основний (успішний) сценарій:

Етап 1. Для реєстрації на сайті інтернет-магазину користувач натискає кнопку «Реєстрація», що знаходиться на початковій сторінці.

Етап 2. Система відображає сторінку реєстрації.

Етап 3. Користувач вводить свій ідентифікатор і пароль, після цього клацає на кнопці ОК.

Етап 4. Система визначає роль користувача. Для цього система порівнює введену інформацію з даними облікового запису персоналу і повертає користувача на початкову сторінку. Якщо дані користувача співпадають з даними облікового запису персоналу, то система вважає, що реєструється Персонал, інакше – Покупець.

Етап 5. Система зберігає реєстраційні дані Персонал чи Покупець у БД.

Етап 6. Завершення прецеденту.

Діаграма робастності показана на рисунку (5.3):

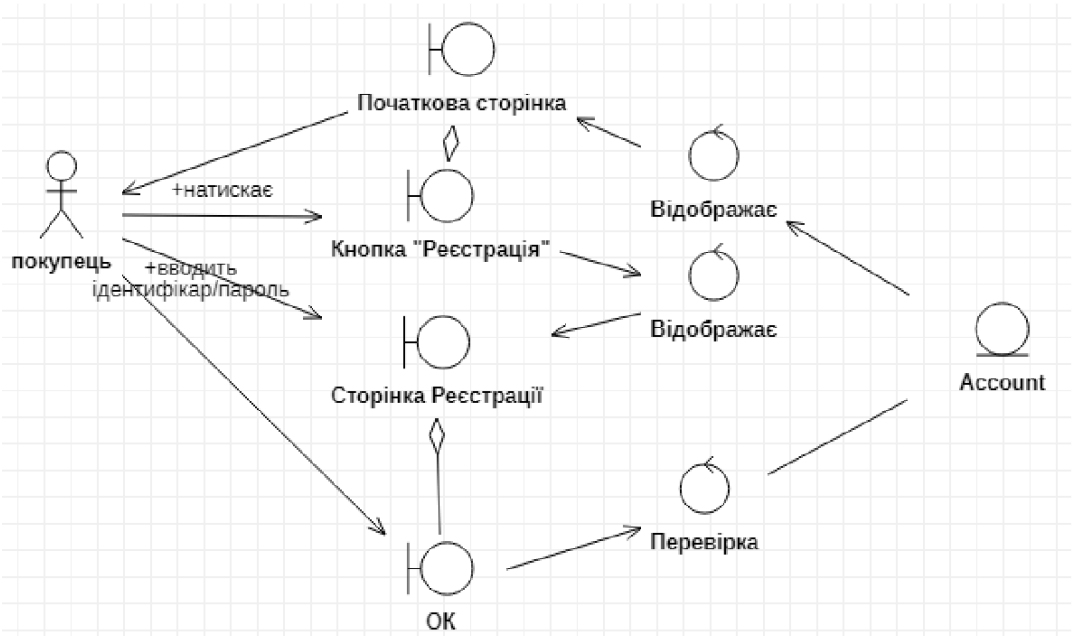


Рисунок 5.3 – Діаграма робастност для прецеденту "Реєстрація"

5.2.1. Створюємо нову діаграму класів для Прецеденту «Реєстрація», як прописано у пункті **5.1.1.**

5.2.2. У вкладці Властивості (Properties) введіть ім'я нової діаграми Класів Registration (Реєстрація).

5.2.3. Клацніть в браузері на цій діаграмі, щоб відкрити її.

5.2.4. Створіть у вікні діаграми класів класи: Початкова сторінка, Сторінка Реєстрації, Обліковий опис, Контролер реєстрації, за допомогою кнопки 

на панелі інструментів. Ім'я класу вводиться з вкладки Властивості (Properties).

5.2.5. Додайте операції, вказані на діаграмі класів рис. 5.4.

Для цього виділіть відповідний клас, з контекстного меню обираємо Add і далі Operation.

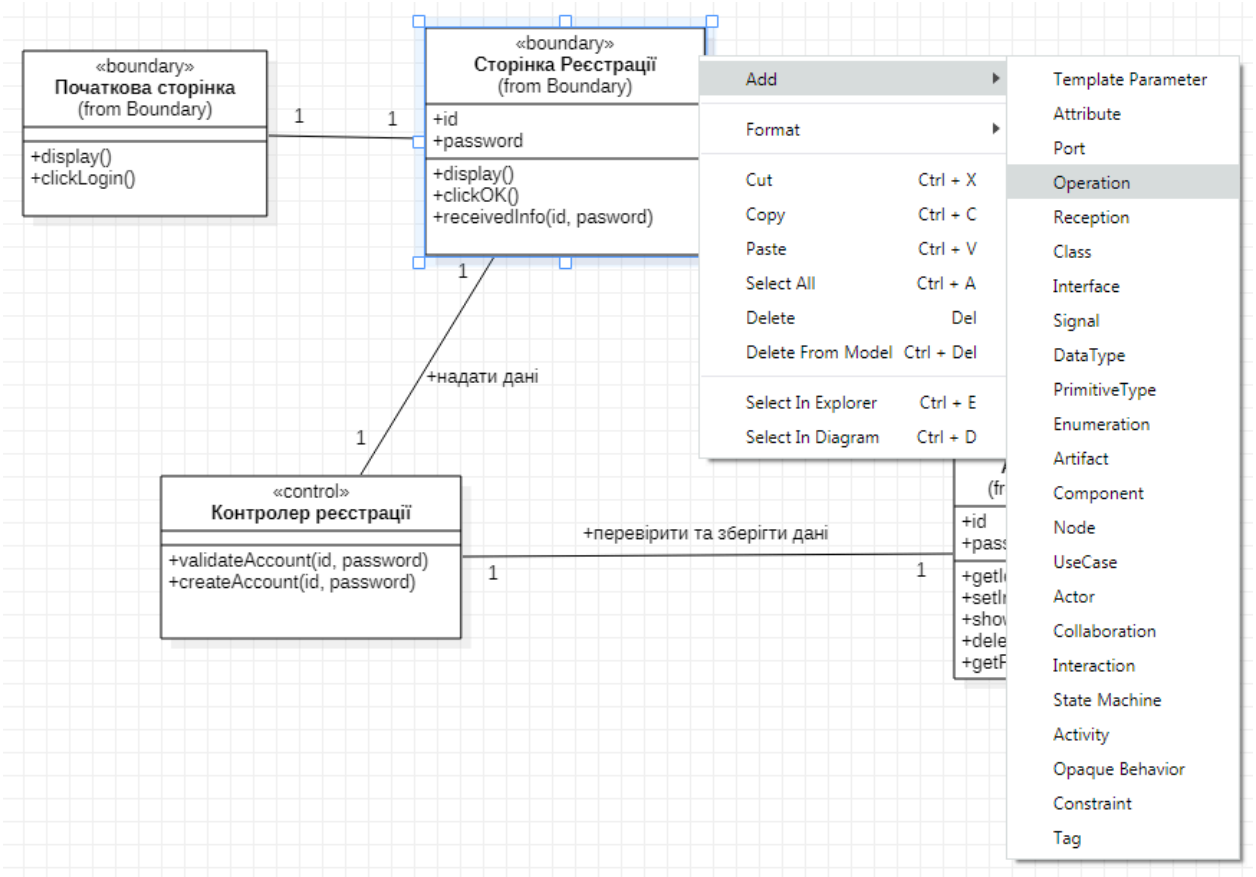


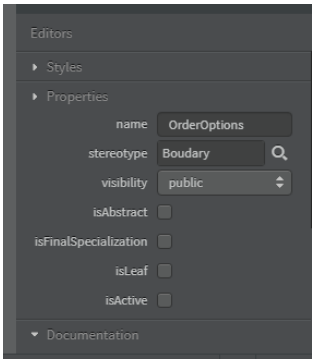
Рисунок 5.4 – Додавання операцій до класу.

5.3. Додавання стереотипів до класів

5.3.1. Двічі клацніть мишею на класі **Початкова сторінка** діаграми.

5.3.2. Додайте у полі імені класу стереотип << Boundary>>.

5.3.3. У списку, що розкривається, в полі стереотипів тепер буде стереотип Boundary.



.

5.3.4. Клацніть мишею на класі **Сторінка Реєстрації** діаграми.
5.3.5. Додайте у полі імені класу стереотип << Boundary>>.
5.3.6. Додайте стереотипи Control для класу **Контролер реєстрації**, а для класу **Account** - стереотип <<Entity>>.

Тепер діаграма Класів повинна виглядати як на рисунку 5.5.

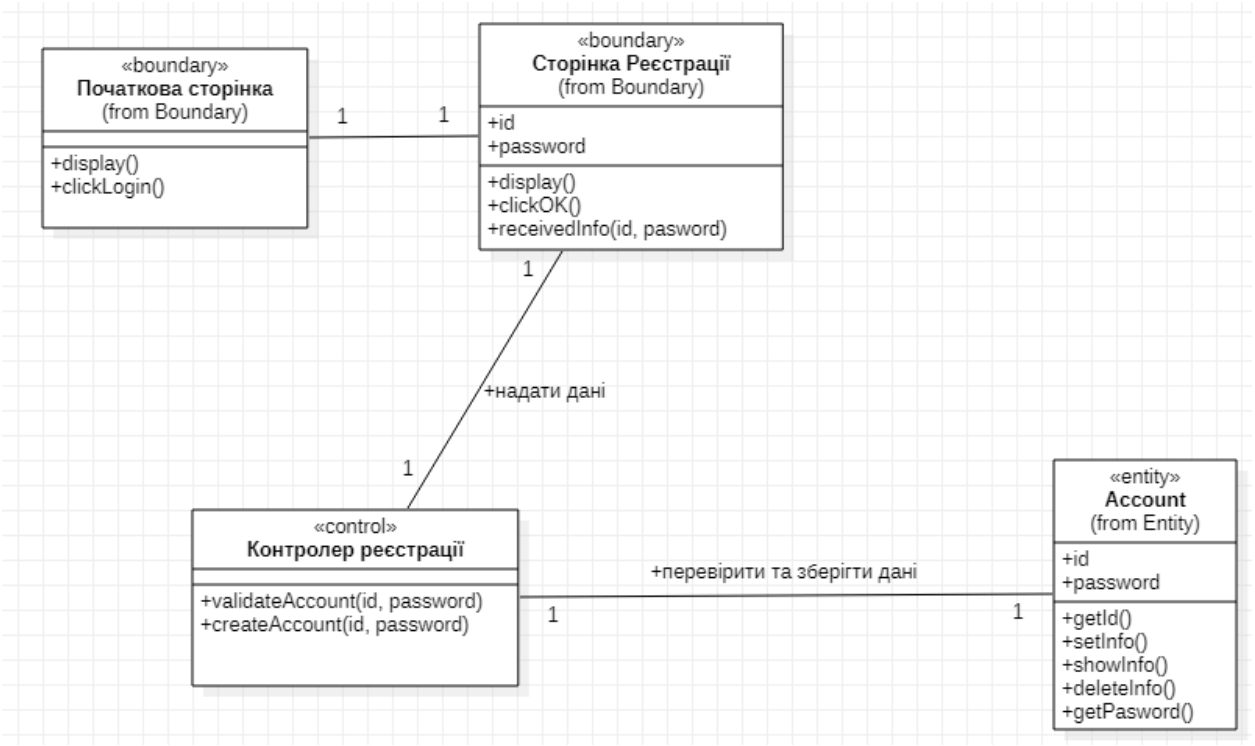


Рисунок 5.5 - Діаграма Класів Registration.

5.4 Об'єднання класів в пакети

5.4.1. Перетягніть в браузері клас **Початкова сторінка** до пакету Boundaries.
5.4.2. Перетягніть клас **Сторінка Реєстрації** до пакету Boundaries.
5.4.3. Перетягніть клас **Контролер реєстрації** до пакету Control.
5.4.4. Перетягніть клас **Account** до пакету Entities.

Завдання до практичної роботи:

1. Створити головну діаграму класів у StarUML. Додати на діаграму пакети Entities (Сутність), пакети Boundaries (Граничні) та Control (Управління).
2. Створити діаграми класів для всіх основних варіантів використання **Діаграми**

варіантів використання для CRM-системи.

3. До класів додати стереотипи << Boundary>>, <<Entity>>, <<Control>>.
4. Об'єднати класи в пакети за відповідними стереотипами.
5. Ознайомитись з теоретичним матеріалом – вивчити поняття діаграм робастност, їх призначення та правил побудови.
6. Оформити звіт, який має містити:

− побудовані діаграми класів;

− висновки.

Контрольні питання:

1. Що таке клас у UML? Які основні елементи він містить?
2. Чим клас відрізняється від об’єкта?
3. Підходи до виявлення класів.
4. Що таке пакет у UML та для чого його використовують?
5. Які стереотипи використовуються у діаграмах класів?
6. Як у StarUML будуються Діаграми класів?
7. Як у StarUML будуються Діаграми пакетів?

Практичне заняття 6

Побудова UML діаграми взаємодій.

Мета заняття: Вивчення особливостей діаграм взаємодії, здобуття практичних навичок побудови діаграм взаємодій та забезпечення їх узгодженості з діаграмами класів та сценаріями варіантів використання.

Зміст заняття: Відображення сценаріїв варіантів використання на діаграмі послідовностей; побудова діаграми послідовностей для прикладу програмної системи; узгодження діаграм послідовностей із діаграмами класів та сценаріями використання.

Теоретичні основи

6.1.Створення діаграм взаємодій.

Будування діаграми послідовностей можна виконати в такий спосіб:

1. Створити нову діаграму взаємодії (Sequence Diagram) з назвою Прецеденту.
2. Визначити сцену взаємодії відповідно до сценарію, з’ясувавши, які об’єкти беруть в ній участь.
3. Перенести об’єкти з браузера (Model Explorer) в поле діаграми взаємодії, розпочинаючи з об’єкту, який ініціює взаємодію.
4. Наносимо повідомлення між об’єктами, розпочинаючи з повідомлення, що ініціює взаємодію. Всі наступні повідомлення розташовуємо згори вниз між лініями життя об’єктів.
5. Перевіряємо, що для кожного повідомлення на діаграмі взаємодій існує відповідна операція в класі, який відповідає об’єкту-приймачу повідомлення, якщо операція не існує додаємо відповідне повідомлення на діаграму взаємодії, тим самим ми добавляємо операцію у відповідний клас.

6.2. РОЗРОБКА ДІАГРАМ ПОСЛІДОВНОСТІ В СЕРЕДОВИЩІ StarUML

6.2.1 Діаграми послідовності

Діаграма послідовності є формою візуалізації взаємодій в моделі і в контексті UML

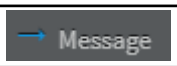
описує динамічний аспект взаємодії об'єктів при реалізації окремих варіантів використання. Активізувати робоче вікно діаграми послідовності можна двома способами:

−Виконати операцію головного меню: Model→Add Diagram→Sequence Diagram.

−Виконати операцію контекстного меню Add Diagram→Sequence Diagram в вікні Model Explorer.

При цьому з'являється нове вікно із чистим робочим аркушем діаграми послідовностей і спеціальна панель інструментів, що містить кнопки із зображенням графічних примітивів, необхідних для розробки діаграми послідовності (табл.6.1).

Таблиця 6.1 – Призначення основних кнопок спеціальної панелі інструментів діаграми послідовності

Зображення кнопки	Призначення кнопки
	Перетворює зображення курсору у форму стрілки для наступного виділення елементів на діаграмі
	Додає на діаграму об'єкт
	Додає на діаграму просте повідомлення
	Додає на діаграму рефлексивне повідомлення

6.2.1.1 Додавання об'єкта на діаграму послідовності та редагування його властивостей

Додати об'єкт на діаграму послідовності можна як стандартним образом за допомогою відповідної кнопки на спеціальній панелі інструментів, так і більше зручним способом – за допомогою перетаскування зображення піктограми класу з браузера моделі на вільне місце робочого аркуша діаграми послідовності.

Як результаті цих дій на діаграмі послідовності з'явиться зображення об'єкта з іменем класу і вертикальною пунктирною лінією, що означає лінію життя цього об'єкта (рис.6.1).

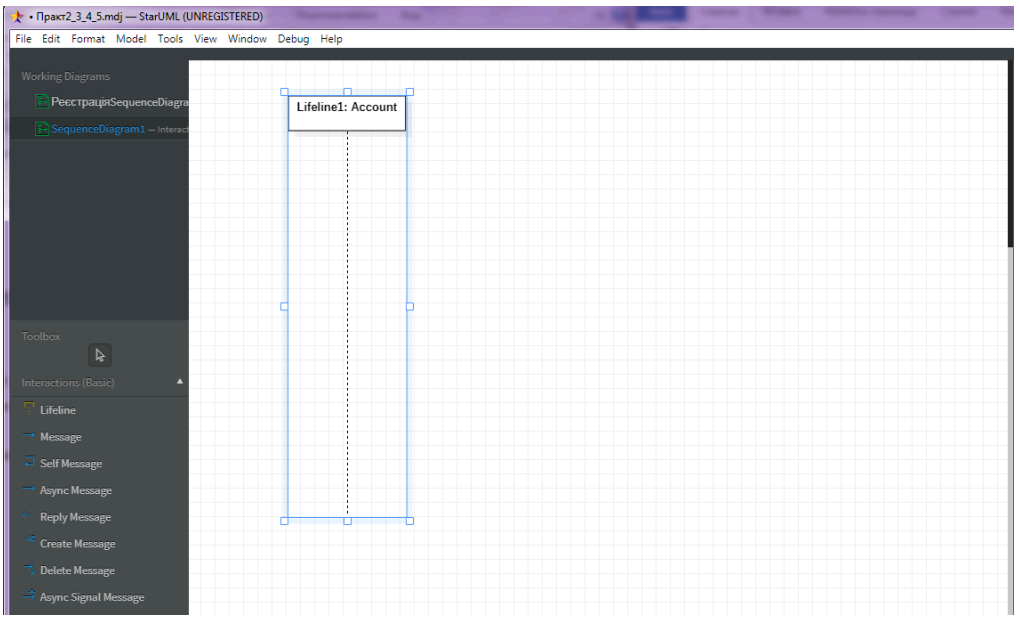


Рисунок 6.1 − Діаграма послідовності після додавання анонімного об'єкта класу Account

Для діаграми послідовності кожний об’єкт, що додається, як усталено вважається

анонімним. При необхідності можна задати власне ім'я об'єкта, для чого подвійним кліком на зображенні об'єкта на діаграмі кооперації слід перейти в режим редагування властивостей об'єкта (рис.6.2).

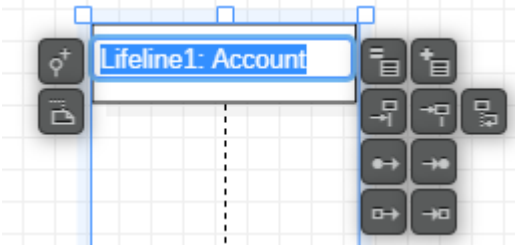


Рисунок 6.2 − Режим редагування властивостей об’єкта

Крім того, змінити властивості об’єкта можна в вікні редагування властивостей елементів.

6.2.1.2 Додавання повідомлення на діаграму послідовності

Для додавання повідомлення між раніше розташованими на діаграмі об'єктами потрібно за допомогою лівої кнопки миші нажати кнопку із зображенням повідомлення на спеціальній панелі інструментів, відпустити ліву кнопку миші, клацнути лівою кнопкою миші на зображенні лінії життя одного об'єкта на діаграмі та відпустити її на зображенні лінії життя другого об'єкта.

Як результаті цих дій на діаграмі з'явиться зображення повідомлення. При цьому зображення лінії життя у об'єкта, якому передається повідомлення, зміниться на зображення фокуса управління (рис.6.3).

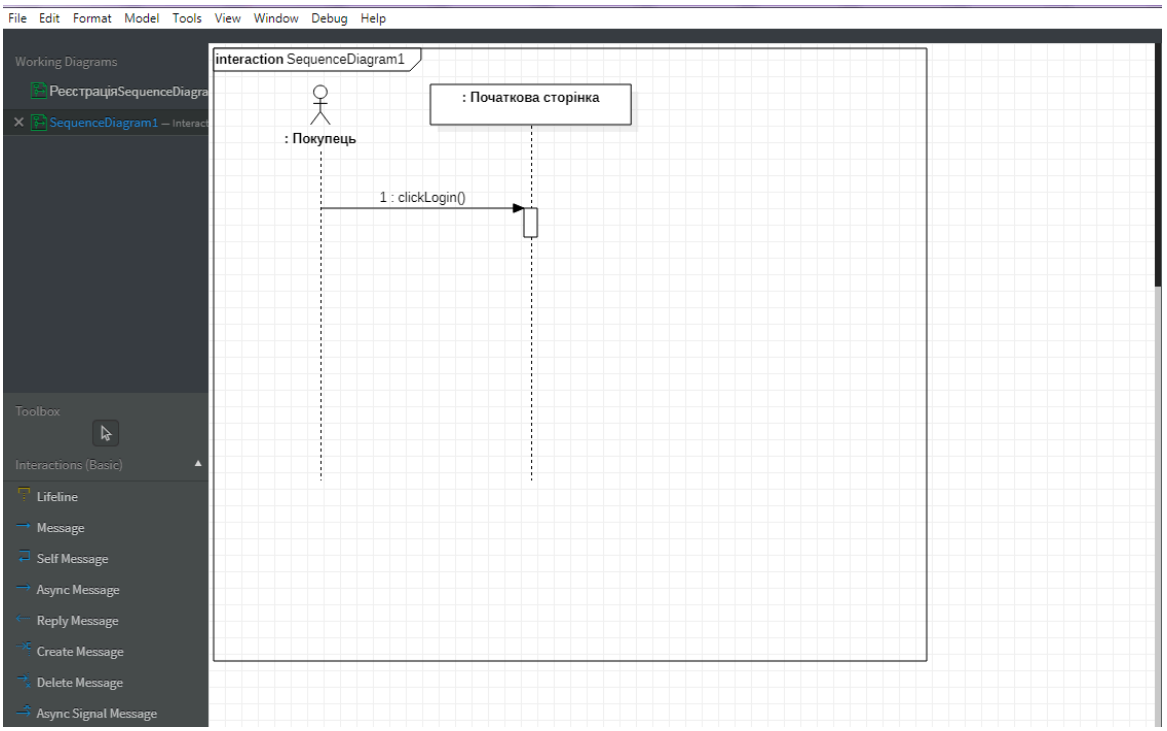


Рисунок 6.3 − Діаграма послідовності після додавання повідомлення

Для специфікації властивостей доданого повідомлення призначене спеціальне вікно, яке автоматично відкривається при доданні нового повідомлення. Ім'я повідомлення можна вибрати в текстовому полі або вибрати зі списку операцій відповідного об’єкту класу (рис.6.4).

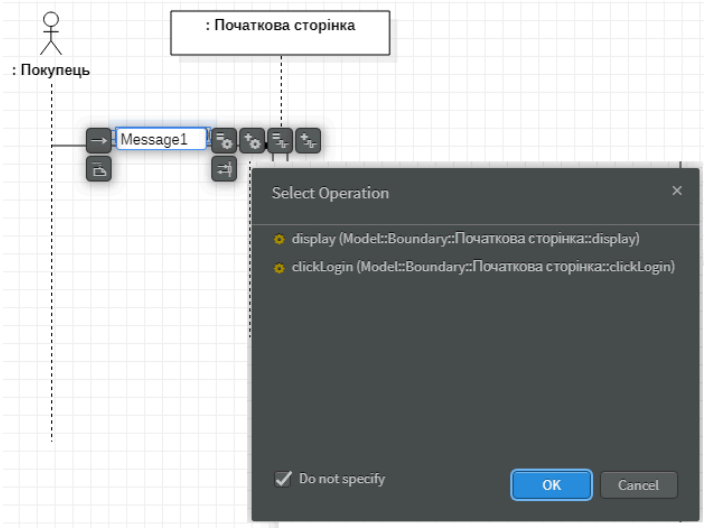


Рисунок 6.4 − Діалогове вікно вибору операції

Будування діаграми послідовності зводиться до додавання та редагування властивостей окремих об'єктів і повідомлень. Для редагування властивостей відповідних елементів можна скористатися з можливостей вікна властивостей. При додаванні повідомлень на діаграму послідовності вони одержують як усталено свій номер в загальній послідовності повідомлень.

Якщо необхідно змінити порядок проходження повідомлень, то з двох діаграм взаємодії цю дію зручніше виконати на діаграмі послідовності, ніж на діаграмі кооперації. В цьому випадку досить нажати ліву кнопку миші на стрілці відповідного повідомлення і, не відпускаючи її, перетягнути вертикально нагору або вниз це повідомлення.

За допомогою вікна редактора властивостей можна задати ім'я зв'язку, стереотип, видимість відповідної пари об'єктів і наявність загальних ролей.

На рисунку 6.5 зображена діаграма послідовностей для прецеденту Реєстрація.

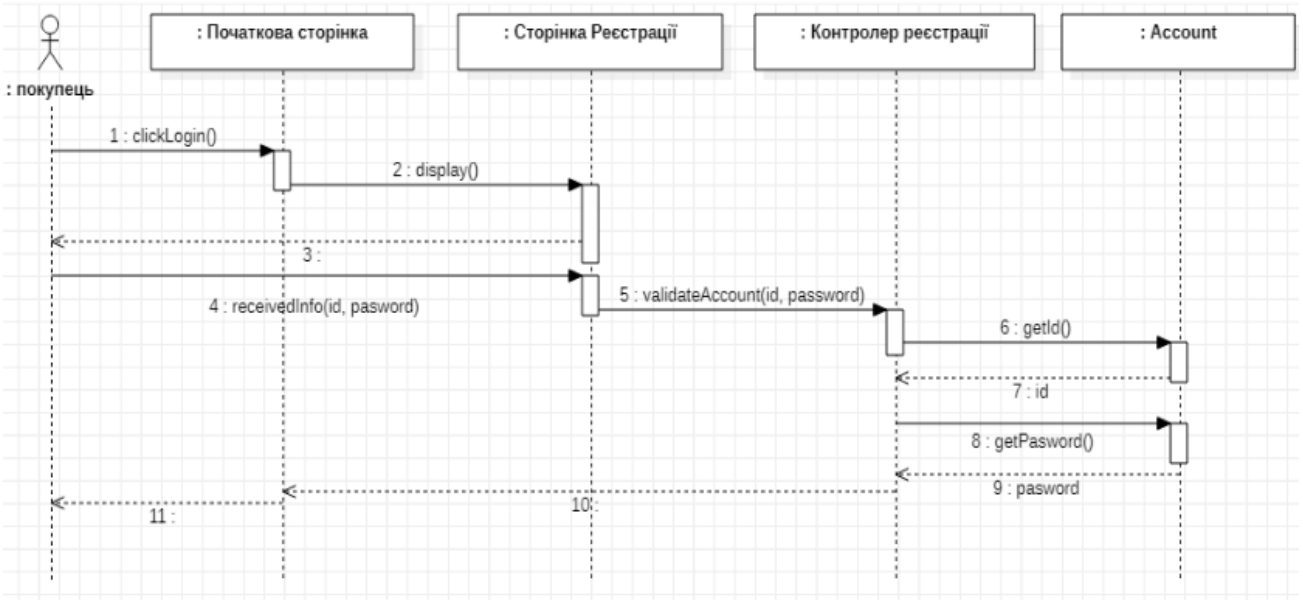


Рисунок 6.5 − Діаграма Послідовності з показаними на ній операціями.

6.2.2 Кооперативна діаграма

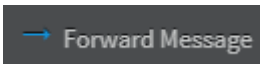
Етапи розробки Кооперативної діаграми в StarUML:

- Виконати операцію головного меню: Model→Add Diagram→ Communication Diagram.

− Виконати операцію контекстного меню Add Diagram→ Communication Diagram в вікні Model Explorer.

При цьому з'являється нове вікно із чистим робочим аркушем діаграми кооперацій і спеціальна панель інструментів, що містить кнопки із зображенням графічних примітивів, необхідних для розробки діаграми послідовності (табл.6.2).

Таблиця 6.2 – Призначення основних кнопок спеціальної панелі інструментів діаграми кооперації

Зображення кнопки	Призначення кнопки
	Перетворює зображення курсору у форму стрілки для наступного виділення елементів на діаграмі
	Додає на діаграму об'єкт
	Додає на діаграму зв'язок між об’єктами
	Додає на діаграму повідомлення
	Додає на діаграму рефлексивний зв'язок

6.2.2.1 Додавання об'єкта на діаграму кооперації та редагування його властивостей

Додати об'єкт на діаграму кооперації можна як стандартним образом за допомогою відповідної кнопки на спеціальній панелі інструментів, так і більше зручним способом – за допомогою перетаскування зображення піктограми класу з браузера моделі на вільне місце робочого аркуша діаграми послідовності.

Як результаті цих дій на діаграмі кооперації з'явиться зображення об'єкта з іменем класу (рис.6.6).

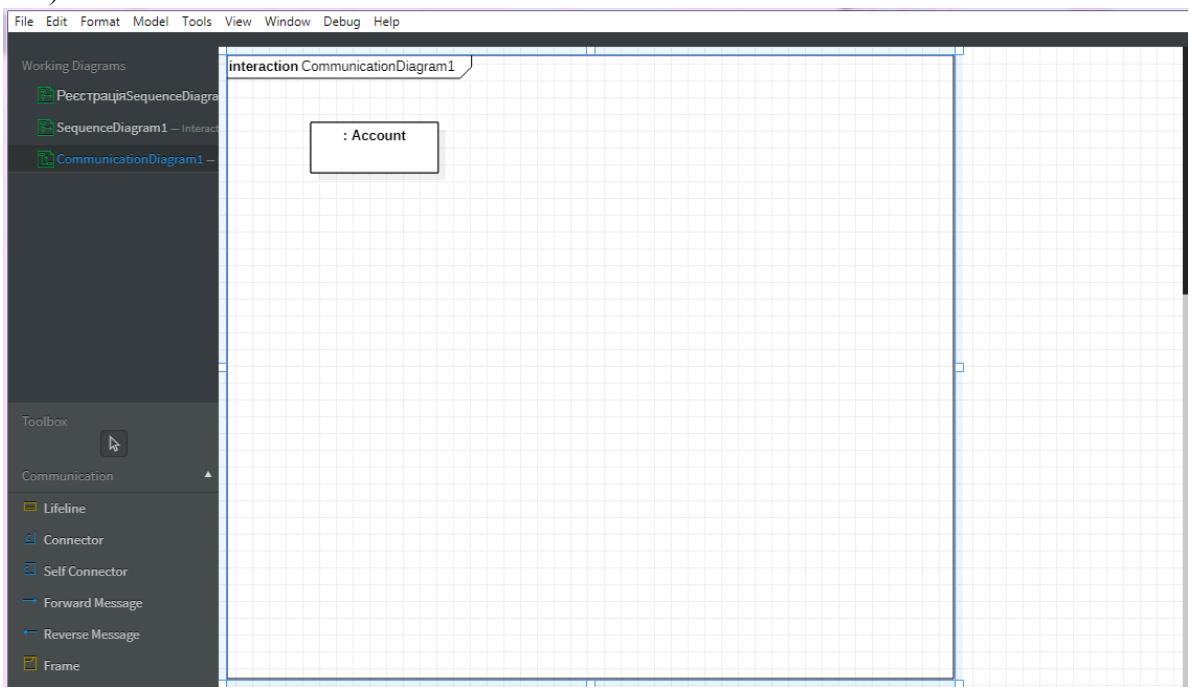


Рисунок 6.6 − Діаграма кооперації після додавання анонімного об'єкта класу Account

Для діаграми кооперації кожний об’єкт, що додається, як усталено вважається анонімним. При необхідності можна задати власне ім'я об'єкта, для чого подвійним кліком на зображенні об'єкта на діаграмі кооперації слід перейти в режим редагування властивостей об'єкта (рис.6.7).

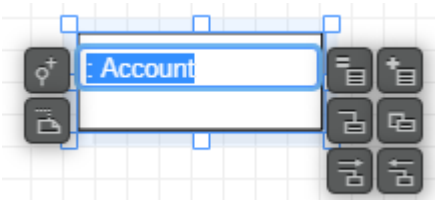


Рисунок 6.7 − Режим редагування властивостей об’єкта на діаграмі кооперації

6.2.2.2 Додавання повідомлення на діаграму послідовності

Для додавання повідомлення між раніше розташованими на діаграмі об'єктами потрібно за допомогою лівої кнопки миші нажати кнопку із зображенням повідомлення на спеціальній панелі інструментів, відпустити ліву кнопку миші, клацнути лівою кнопкою миші на зображенні першого об'єкта на діаграмі та відпустити її на зображенні другого об'єкта.

Як результаті цих дій на діаграмі з'явиться зображення повідомлення (рис.6.8).

Для специфікації властивостей доданого повідомлення призначене спеціальне вікно, яке автоматично відкривається при доданні нового повідомлення, якщо обрати  (Select operation). Ім'я повідомлення можна вибрати зі списку операцій відповідного об’єкту класу (рис.6.8).

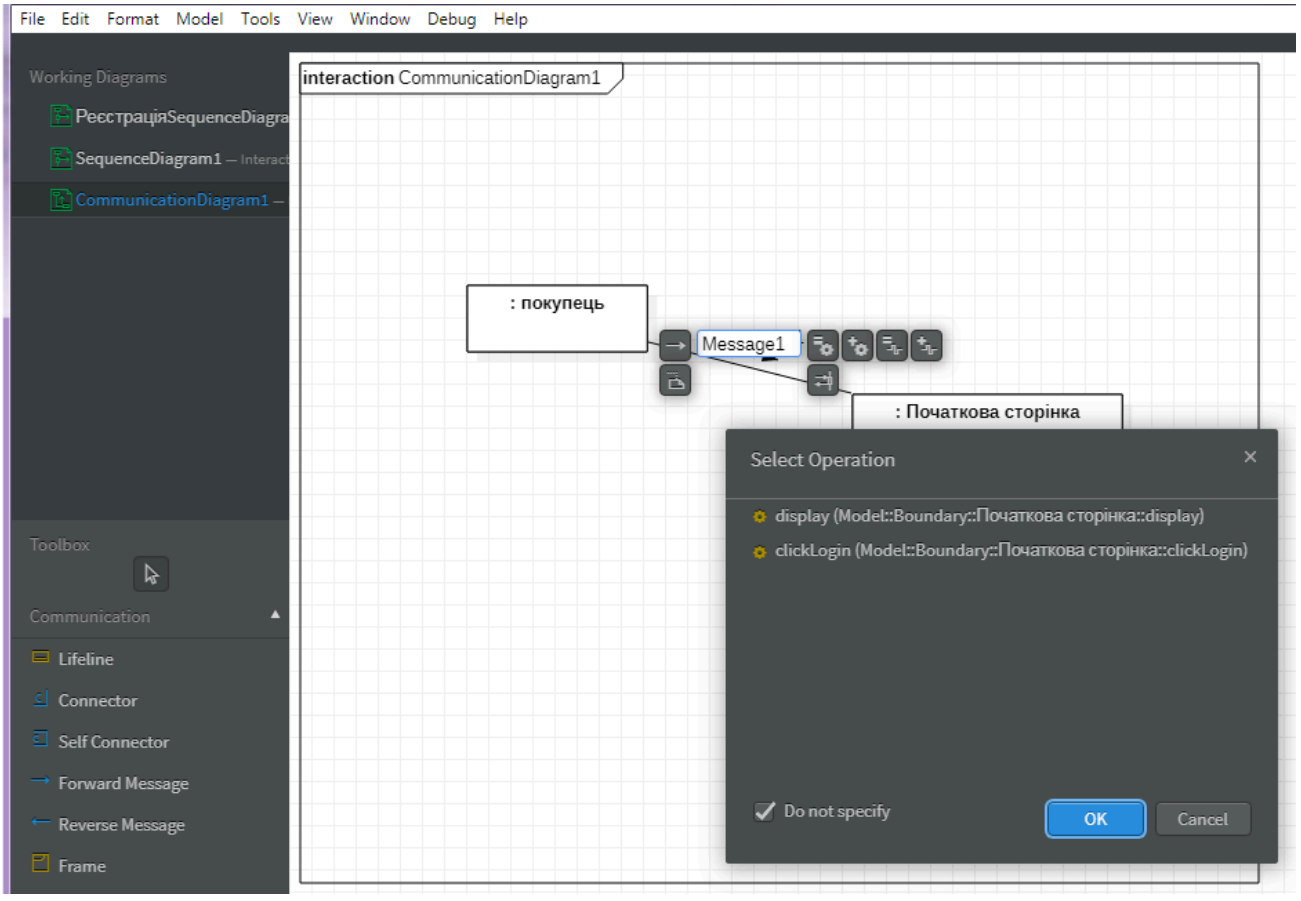


Рисунок 6.8 − Діаграма кооперації після додавання повідомлення

Будування діаграми кооперації зводиться до додавання та редагування властивостей окремих об'єктів і повідомлень. Для редагування властивостей відповідних елементів можна скористатися з можливостей вікна властивостей. При додаванні повідомлень на діаграму кооперації вони одержують як усталено свій номер в загальній послідовності повідомлень.

За допомогою вікна редактора властивостей можна задати ім'я зв'язку, стереотип, видимість відповідної пари об'єктів і наявність загальних ролей.

На рисунку 6.9 зображена діаграма кооперації для прецеденту Реєстрація.

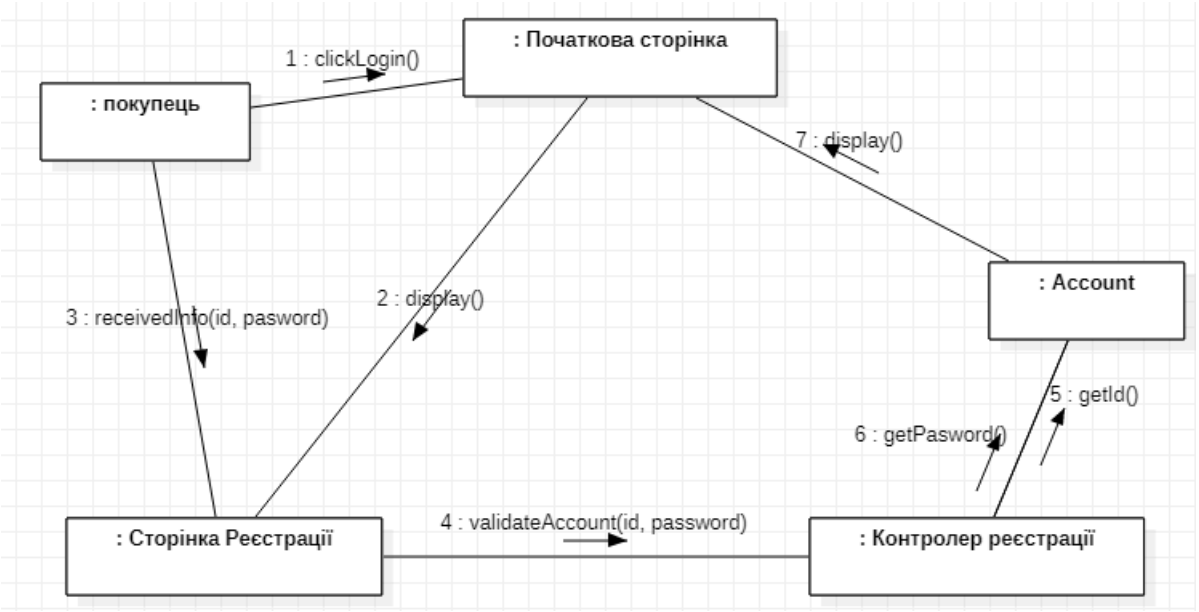


Рисунок 6.9 − Діаграма кооперації з показаними на ній операціями.

6.3 Доповнення діаграми класів наборами операцій для кожного класу

Якщо з діаграм взаємодій (послідовностей або кооперації) зрозуміло, що певний клас має виконати додаткову операцію, то додайте цю операцію до діаграми класів виконав додавання операції на діаграмі взаємодій (обрати  Create operation) (рис. 6.10-6.11).

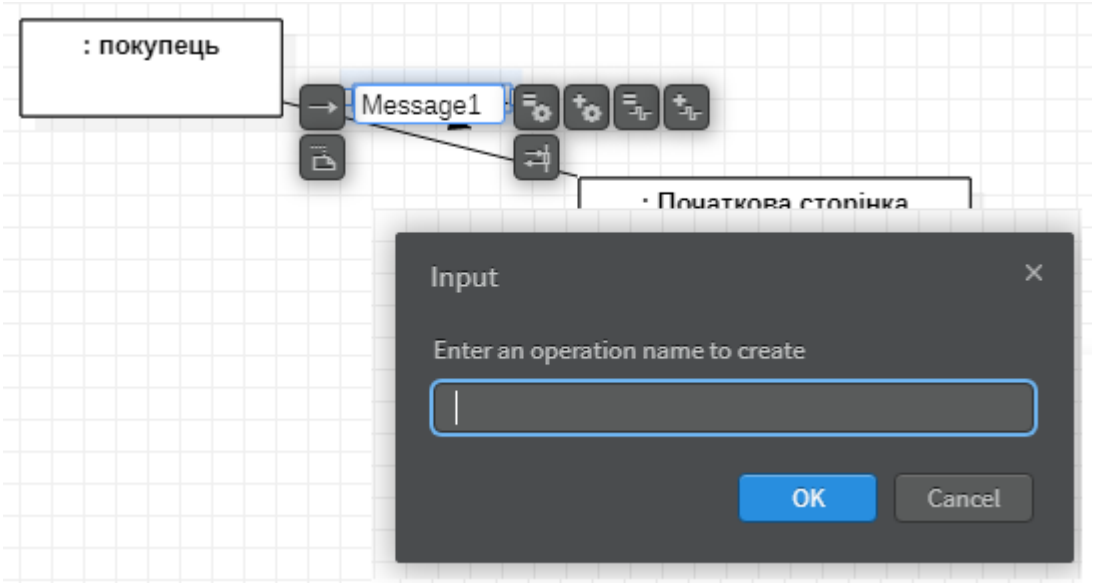


Рисунок 6.10 − Діалогове вікно створення нової операції на діаграмі кооперації

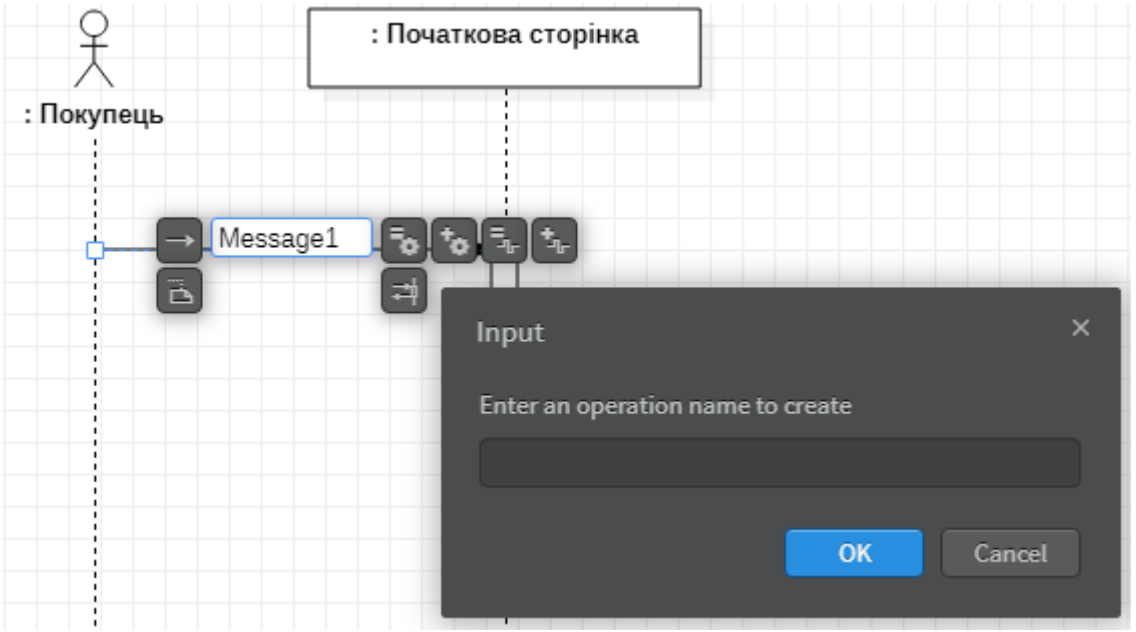


Рисунок 6.11 – Діалогове вікно створення нової операції на діаграмі послідовностей

Завдання до практичної роботи:

1. Побудувати діаграми взаємодій у StarUML для всіх сценаріїв варіантів використання для CRM-системи.

1.1. Побудувати діаграми послідовностей для всіх сценаріїв варіантів використання.

1.2. Побудувати Кооперативні діаграми для всіх сценаріїв варіантів використання.

2. З використанням діаграм взаємодій доповнити (у разі необхідності) діаграму класів наборами операцій для кожного класу.

3. Оформити звіт, який має містити:

– побудовані діаграми послідовностей;

– побудовані Кооперативні діаграми;

– висновки.

Контрольні питання:

1. Що таке діаграма послідовностей?
2. Для чого створюють у системі діаграму послідовностей?
3. Як створити діаграму послідовностей?
4. Як додати об'єкти на діаграму послідовностей?
5. Як додати повідомлення на діаграму послідовностей?
6. Як додати на діаграму додаткові об’єкти?
7. Як на діаграмі послідовності відобразити послідовну інформацію?
8. Що таке діаграма кооперації?
9. Для чого створюють у системі діаграму кооперації?
10. Як створити діаграму кооперації?
11. Як додати об'єкти на діаграму кооперації?
12. Як додати повідомлення на діаграму кооперації?
13. Як додати на діаграму додаткові об'єкти кооперації?
14. Як на діаграмі кооперації відобразити послідовну інформацію?

Практичне заняття 7

Побудова діаграм діяльності.

Мета завдання: Навчитись моделювати потоки виконання в межах системи для представлення логіки процесів або функціональних сценаріїв та перевіряти їх узгодженість з діаграмами класів та сценаріями варіантів використання.

Зміст завдання: побудувати UML-діаграму діяльності для заданого сценарію використання, відобразивши послідовність дій та перевіривши її узгодженість із діаграмами класів і варіантів використання.

Теоретичні основи

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. Зазвичай для цієї мети використовувалися блок-схеми або структурні схеми алгоритмів.

Для моделювання процесу виконання операцій в мові UML використовуються так звані діаграми діяльності. Вживана в них графічна нотація багато в чому схожа на нотацію діаграми станів, оскільки на діаграмах діяльності також присутні позначення станів і переходів. Кожне перебування на діаграмі діяльності відповідає виконанню деякої елементарної операції, а перехід в наступний стан спрацьовує тільки при завершенні цій, операції в попередньому стані. Графічно діаграма діяльності представляється у формі графа діяльності, вершинами якого є стани дії, а дугами — переходи від одного стану дії до іншого.

Основним напрямом використання діаграм діяльності є візуалізація особливостей реалізації операцій класів, коли необхідно представити алгоритми їх виконання. При цьому кожен стан може бути виконанням операції деякого класу або її частини, дозволяючи використовувати діаграми діяльності для опису реакцій на внутрішні події системи.

У контексті мови UML діяльність (activity) є деякою сукупністю окремих обчислень, що виконуються автоматом. При цьому окремі елементарні обчислення можуть приводити до деякого результату або дії (action). На діаграмі діяльності відображається логіка або послідовність переходу від однієї діяльності до іншої, при цьому увага фіксується на результаті діяльності. Сам же результат може привести до зміни стану системи або повернення деякого значення.

7.1. Розробка діаграм діяльності в середовищі StarUML

Активізувати робоче вікно діаграми діяльності в середовищі StarUML можна двома способами:

- Виконати операцію головного меню: Model→Add Diagram→Activity Diagram.
- Виконати операцію контекстного меню Add Diagram→ Activity Diagram в вікні Model Explorer.

При цьому з'являється нове вікно із чистим робочим аркушем діаграми послідовностей і спеціальна панель інструментів, що містить кнопки із зображенням графічних примітивів, необхідних для розробки діаграми послідовності (табл.7.1).

Таблиця 7.1 – Призначення основних кнопок спеціальної панелі інструментів діаграми діяльності

Зображення кнопки	Призначення кнопки
	Використовується для позначення конкретної дії чи кроку алгоритму

Initial	(Початкова точка) з якої починається процес
Final	(Кінцева точка) позначає завершення процесу.
Fork	Розгалуження використовується для поділу потоку на кілька паралельних.
Join	Об’єднання паралельних гілок потоку в один
Merge	(Об’єднання) – ромб для злиття альтернативних потоків в один
Decision	(Ромб) – для розгалужень: «if/else», вибір умов

7.1.1 Діаграма діяльності в середовищі StarUML

Побудуємо діаграму діяльності для прецеденту "Реєстрація" в середовищі StarUML (рис. 7.1).

Основний (успішний) сценарій:

Етап 1. Для реєстрації на сайті інтернет-магазину користувач натискає кнопку «Реєстрація», що знаходиться на початковій сторінці.

Етап 2. Система відображає сторінку реєстрації.

Етап 3. Користувач вводить свій ідентифікатор і пароль, після цього клацає на кнопці ОК.

Етап 4. Система визначає роль користувача. Для цього система порівнює введену інформацію з даними облікового запису персоналу і повертає користувача на початкову сторінку. Якщо дані користувача співпадають з даними облікового запису персоналу, то система вважає, що реєструється Персонал, інакше – Покупець.

Етап 5. Система зберігає реєстраційні дані Персонал чи Покупець у БД.

Етап 6. Завершення прецеденту.

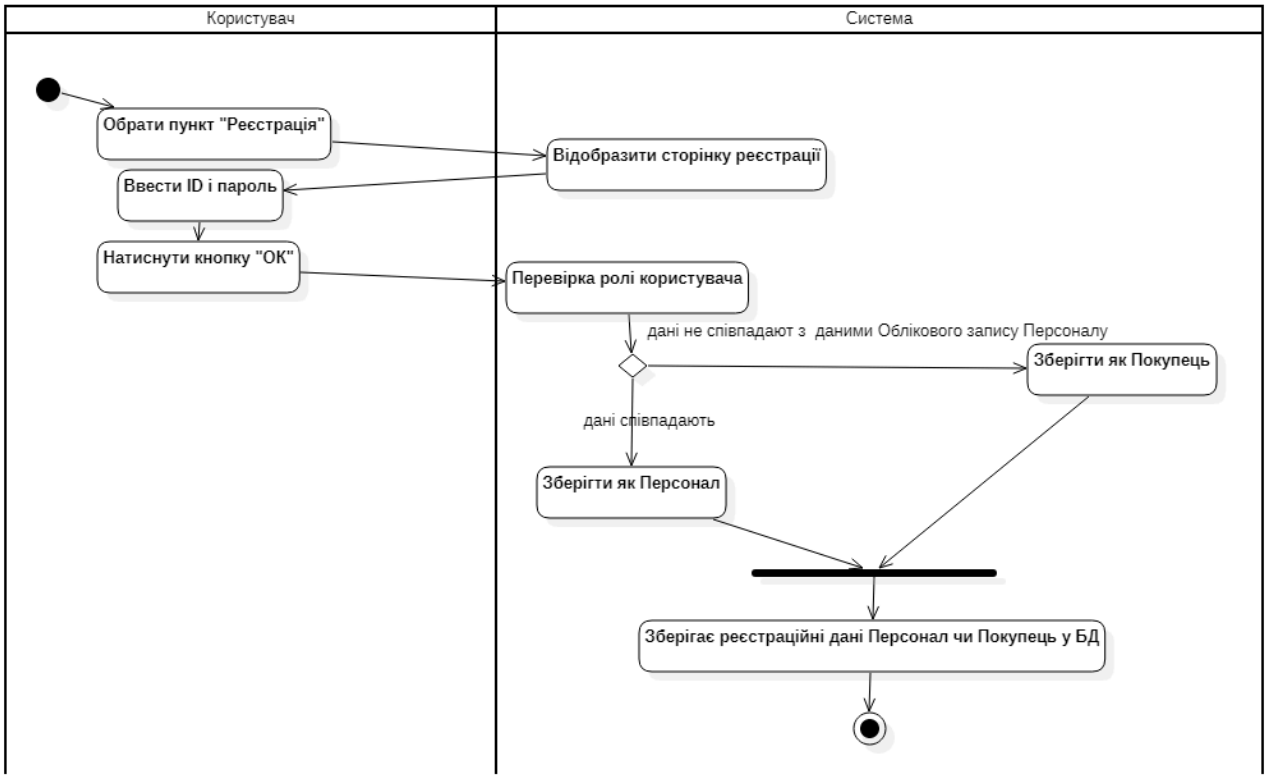


Рисунок 7.1 − Діаграма діяльності для прецеденту "Реєстрація" в середовищі StarUML

Завдання до практичної роботи:

1. Побудувати діаграми діяльності для всіх сценаріїв варіантів використання для CRM-системи в середовищі StarUML.

1.1Оформити звіт, який має містити:

− побудовані діаграми діяльності;

− висновки.

Контрольні питання

1. Для чого використовується діаграма діяльності?
2. Як графічно зображується діаграма діяльності в в середовищі StarUML?
3. Що таке стан діяльності. Зобразите.
4. Як зображується галуження?
6. Який символ використовують для розпаралелювання обчислень?
7. Як вирішують програму синхронізації діяльності?
8. Що таке «плавальна доріжка»?

Рекомендована література

1. Mall, Rajib. Fundamentals of software engineering. PHI Learning Pvt. Ltd., 2018. 612 p.
2. Цибульник, Сергій Олексійович, and Катерина Сергіївна Барандич. Технології розроблення програмного забезпечення. Частина 1. Життєвий цикл програмного забезпечення. Підручник, 2022, 270 с.
3. Жихаревич В.В. Професійна практика програмної інженерії : навчальний посібник. Чернівці : Чернівецький національний університет, 2015. 384 с.
4. Maciaszek, Leszek. Requirements analysis and system design. Pearson Education, 2007. 656 p.
5. Grady Booch, Ivar Jacobson, and James Rumbaugh. The Unified Modeling Language User Guide. Second Edition. The Addison-Wesley Object Technology Series, 2005, 475 p.
6. Fowler, Martin. UML distilled: a brief guide to the standard object modeling language. Addison-Wesley Professional, 2018, 208 p.

Додаткова література

1. Радуш, В. В., Радуш, В. В., Лебедєва, О. Ю., Лебедева, Е. Ю., Кушніренко, Н. І., Кушниренко, Н. И., ... & Зорило, В. В. (2021). Моделювання організаційних заходів для створення політики безпеки організації з використанням бізнес-процесів. Інформатика та математичні методи в моделюванні, 11(3), 239-246.
2. Лавріщева К.М. ПРОГРАМНА ІНЖЕНЕРІЯ.–К.– 2008.–319 с.
3. Michael Blaha, James Rumbaugh. Object-Oriented Modeling and Design with UML. Second Edition. Pearson, 2004, 496 p.
4. Kruchten, Philippe. The rational unified process: an introduction. Addison-Wesley Professional, 2004, 240 p.
5. Craig Larman. Applying UML and patterns: an introduction to object-oriented analysis and design, Prentice-Hall, Inc., United States, 1997, 507 p.
6. E. Gamma, R. Helm, R. Johnson, J. Vlissides, D. Patterns. Elements of reusable object-oriented software. Design Patterns, 1, 1995, 417 p. URL: https://www.javier8a.com/itc/bd1/articulo.pdf.
7. Войцеховська, О. В. Шаблони проектування програмного забезпечення : навчальний посібник [Електронний ресурс] / О. В. Войцеховська, О. І. Черняк. – Вінниця : ВНТУ, 2022. – (PDF, 100 с.) URL: https://files.znu.edu.ua/files/Bibliobooks/Inshi78/0058358.pdf

Навчальне видання

Методичні рекомендації до практичних занять з дисципліни «Основи програмної інженерії» для здобувачів першого (бакалаврського) рівня вищої освіти за спеціальністю 121 – Інженерія програмного забезпечення

Укладачі:Волкова Наталія Павлівна
Крісілов Віктор Анатолійович