ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
Навчально-науковий інститут економіки, управління, права та інформаційних технологій
Кафедра інформаційних систем та технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти магістр

на тему: **«Імітаційна модель програмного засобу інформаційної системи з обслуговування клієнтів компанії»**

Виконав: здобувач вищої освіти
за освітньою програмою
Інформаційні управляючі системи та технології
спеціальності 126 Інформаційні системи та технології
ступеня вищої освіти магістр
групи 126ІСТ_мд_2022
Ніколаєнко Валерій Олексійович
Керівник: Уткін Юрій Вікторович
Рецензент: Муравльов Володимир Вячеславович

Полтава – 2024 року

ВСТУП

Актуальність теми. Сучасні тенденції цифрової трансформації значно впливають на всі аспекти суспільного життя та економіки. Зокрема, у сфері стільникового зв'язку спостерігається стрімке зростання кількості абонентів і їхніх запитів, що створює серйозний виклик для операторів зв'язку. Для забезпечення ефективного обслуговування клієнтів компанії змушені постійно вдосконалювати свої інформаційні системи, інтегруючи новітні технології автоматизації та моделювання.

Зростаючий обсяг даних і необхідність їх обробки в реальному часі вимагають впровадження гнучких, надійних та адаптивних інформаційних систем. Ці системи мають не лише виконувати звичні операції з обслуговування клієнтів, але й передбачати можливі сценарії розвитку подій, оптимізуючи ресурси та мінімізуючи ризики збоїв. У цьому контексті імітаційне моделювання стає потужним інструментом для аналізу функціональних можливостей інформаційних систем до їхнього впровадження.

Імітаційне моделювання дозволяє відтворювати реальні бізнес-процеси у віртуальному середовищі, тестувати різні сценарії та передбачати можливі проблеми. Це особливо важливо у системах стільникового зв'язку, де оператори повинні забезпечувати безперебійне надання послуг, оперативну обробку запитів і адаптацію до змін ринку. Впровадження таких моделей дозволяє виявляти «вузькі місця» в роботі системи, тестувати нові функціональні можливості та оцінювати показники продуктивності.

Крім того, підвищення конкуренції між операторами зв'язку спонукає їх до оптимізації своїх бізнес-процесів. Використання імітаційних моделей на етапах проєктування інформаційних систем дає змогу скоротити час і витрати на розробку та впровадження нових послуг.

Вагомий внесок у теоретичні основи та практичне застосування методів імітаційного моделювання зробили роботи вчених [3-6], які досліджували способи інтеграції моделювання у сферу телекомунікацій. Їхні дослідження слугують фундаментом для створення сучасних інформаційних систем, орієнтованих на високу продуктивність і надійність.

Отже, тема дослідження є актуальною в умовах сучасного технологічного розвитку, оскільки спрямована на вдосконалення процесів автоматизації обслуговування клієнтів у стільниковому зв'язку через використання імітаційного моделювання.

Зв'язок роботи з науковими програмами, темами. Робота відповідає дослідженням в межах науково-дослідної роботи «Розвиток підприємництва: управлінські, економічні, інноваційна та правові аспекти» відповідно до договору №9 від 15.05.2023 р. між ТОВ «ПАФ Гарант» та Полтавським державним аграрним університетом.

Метою кваліфікаційної роботи є розробка та дослідження імітаційної моделі інформаційної системи «Центр обслуговування абонентів» для оптимізації її функціонування та підвищення надійності обслуговування.

Завданнями кваліфікаційної роботи є:

– аналіз сучасних моделей обслуговування абонентів у системах стільникового зв'язку;

– моделювання інформаційної системи за допомогою UML нотацій;

– розроблення імітаційної моделі програмного засобу інформаційної системи.

Об'єктом дослідження є інформаційні системи, що забезпечують автоматизацію обслуговування абонентів у стільниковому зв'язку.

Предметом дослідження є процеси моделювання та аналізу функціональних показників роботи інформаційних систем за допомогою імітаційних методів.

Методи дослідження – проведені в роботі дослідження базуються на методах структурного аналізу, об'єктно-орієнтованого моделювання, імітаційного моделювання, а також методах математичного аналізу та статистичної обробки даних.

Інформаційна база кваліфікаційної роботи складається з наукових статей, міжнародних аналітичних видань і звітів, матеріалів наукових конференцій, інтернет-ресурсів, що містять інформацію про сучасні тенденції розвитку інформаційних систем, їх проєктування та впровадження.

Елементи наукової новизни полягають у розробці та дослідженні імітаційної моделі, яка враховує марковські процеси, вплив часових параметрів і динамічних змін системи.

Практична значущість роботи полягає полягає в можливості повторного застосування та модифікації розробленого програмного коду для аналізу інших систем із подібною структурою. Отримані результати можуть бути корисними для операторів стільникового зв'язку, розробників програмного забезпечення та дослідників у галузі моделювання інформаційних систем.

Апробація результатів дослідження відбувалася шляхом оприлюднення доповідей на наукових конференціях, семінарах.

Публікації. За результатами проведеного дослідження опубліковано тези: «Моделювання інформаційної системи агропромислового комплексу з використанням об'єктно-орієнтованого та імітаційного методів», Стратегічний менеджмент агропродовольчої сфери в умовах глобалізації економіки: безпека, інновації, лідерство: матеріали II Міжнародної науково-практичної конференції, 27 вересня 2024 р. Полтава; «Розвиток інструментарію для моделювання та розробки інформаційних систем на основі об'єктно-орієнтованого підходу»: Студентські роботи за науковою тематикою кафедри інформаційних систем та технологій: матеріали XXI щорічного міждисциплінарного семінару, 20 листопада 2024 р. Полтава.

Структура та обсяг кваліфікаційної роботи логічно пов'язані з задачами досліджень. Робота містить перелік умовних позначень, вступ, три розділи основної частини, висновки, список використаних джерел, додатки. Загальний обсяг текстової частини дипломної роботи складає 63 сторінки формату А4. Вона містить 33 рисунки і 1 таблицю. У роботі використано 45 науково-технічних джерел.

РОЗДІЛ 1
АНАЛІЗ СУЧАСНИХ МОДЕЛЕЙ ОБСЛУГОВУВАННЯ АБОНЕНТІВ У СИСТЕМАХ СТІЛЬНИКОВОГО ЗВ'ЯЗКУ

1.1 Аналіз сучасних абонентських сервісів у сфері мобільного зв'язку

Мобільний зв'язок є формою радіозв'язку, яка відрізняється масовим обслуговуванням абонентів на певній території. Спочатку стільниковий зв'язок поступово розширював можливості телефонної мережі загального користування (ТфЗК) [8]. Термін «стільниковий зв'язок» зазвичай означає послугу, що надається за допомогою стільникових мереж рухомого зв'язку, побудованих на відповідних технологіях. Отже, поняття охоплює всі аспекти мобільного зв'язку.

На сьогодні ключовим завданням операторів мобільного зв'язку є розширення спектра послуг для абонентів. Це включає полегшення користування мобільними пристроями, вдосконалення способів зв'язку, забезпечення додаткових інформаційних сервісів, спрощення оповіщення та розрахунків. Зростання кількості абонентів стимулює впровадження сучасних автоматизованих систем, які допомагають зменшити витрати. Для абонентів ці системи стають зручним інструментом, перетворюючи телефон на автоматичного секретаря. У конкурентній боротьбі оператори пропонують передові рішення у сфері стільникового зв'язку.

У цьому дослідженні розглядається розробка імітаційної моделі програмного забезпечення для інформаційної системи «Центр обслуговування абонентів» (далі – Система).

Сучасні підприємства часто стикаються з необхідністю оптимізації своїх бізнес-процесів. Функціональне моделювання є одним із найпоширеніших методів аналізу, що дозволяє вивчати наявні процеси, виявляти їхні недоліки та створювати оптимальні моделі. Побудова таких моделей зазвичай відбувається поступово – від загальної схеми до детального опису конкретних процесів [12]. Однак, на етапі

високої деталізації, функціонального підходу може бути недостатньо для оптимізації, тому доцільно використовувати імітаційне моделювання.

Імітаційне моделювання дозволяє створювати моделі з урахуванням фактору часу. Отримані моделі можна аналізувати, «програючи» їх у часовому вимірі для збору статистики про хід процесів у реальному виконанні. Такий підхід забезпечує більш точний пошук оптимальних рішень, особливо за обмежених ресурсів, коли інші математичні методи надто складні.

Для створення моделі необхідно вивчити загальну характеристику підприємства, до якого застосовуватиметься імітаційна модель, а також врахувати специфіку управлінських завдань [14].

Центр обслуговування абонентів займається наданням телекомунікаційних послуг та подальшим обслуговуванням клієнтів. Це невелике підприємство, яке потребує якісної системи з єдиною та достовірною інформацією. Надійність даних має забезпечуватись системою управління, що сприятиме ефективному контролю роботи компанії, процесів та діяльності співробітників.

Система «Центр обслуговування абонентів» розроблена для автоматизації телекомунікаційних послуг. Вона надає операторам можливість створювати електронні договори з абонентами, вести облік клієнтів та забезпечувати якісне обслуговування. Електронні договори мають таку ж юридичну силу, як і паперові, що дозволяє значно скоротити обсяг паперової документації. За бажанням, копія договору може зберігатися у клієнта.

Типовий алгоритм (рис. 1.1) роботи оператора з Системою виглядає так: оператор запускає програму, обирає одного з доступних операторів стільникового зв'язку (оскільки кожен оператор має власний Центральний офіс і сервер), і приступає до роботи. В Системі підтримуються такі види електронних документів: договори на надання послуг зв'язку, заяви на заміну SIM-карти, зміну абонентського номера, деталізацію рахунку та поповнення рахунку.

Для створення документа оператор відкриває форму потрібного типу, заповнює всі необхідні поля та зберігає дані. Під час заповнення система

автоматично перевіряє правильність введених реквізитів і не дозволяє зберегти документ у разі помилки [15].

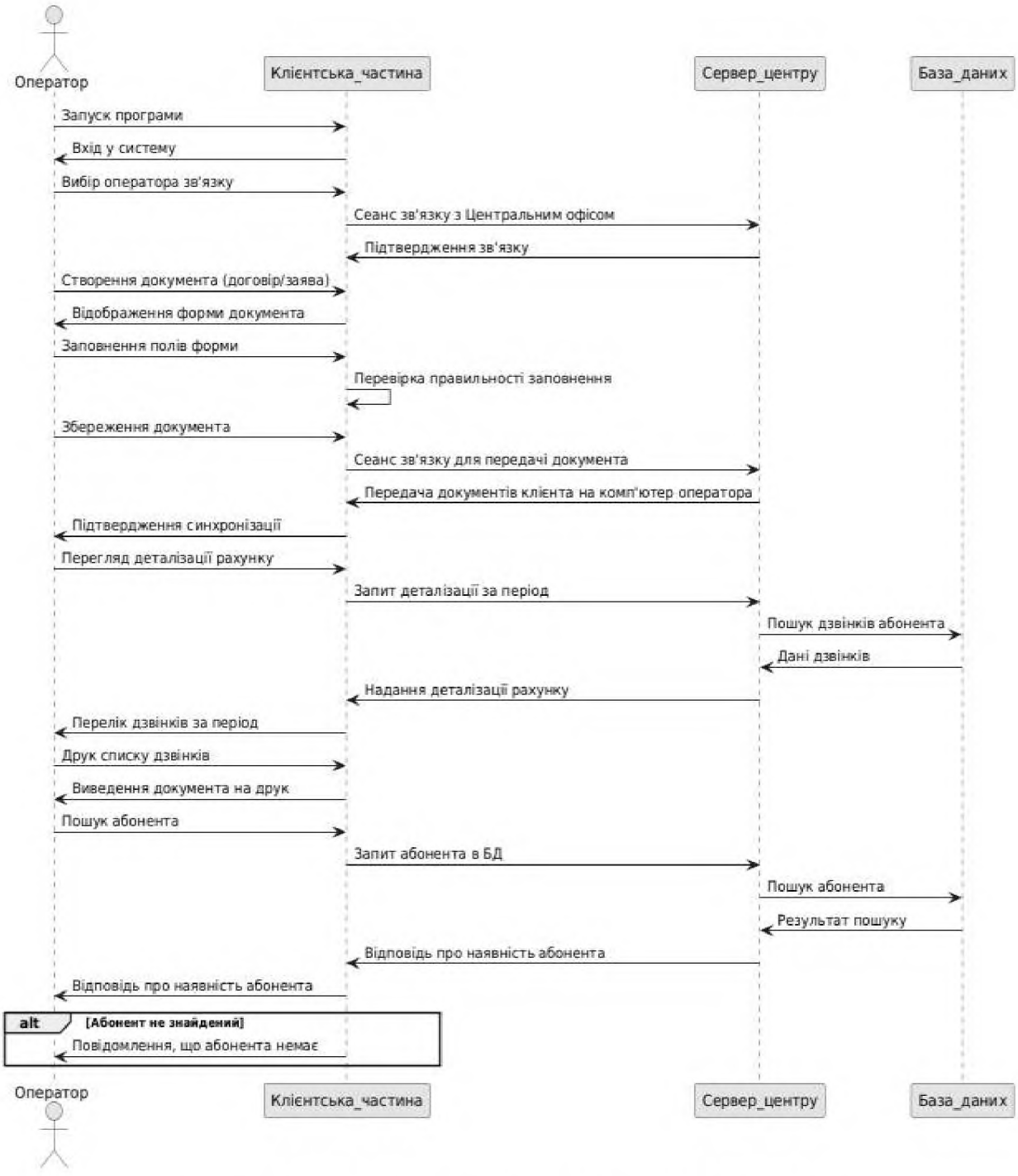


Рисунок 1.1 – Сценарій роботи оператора з Системою

Система має дві основні компоненти: серверну частину програмного забезпечення та клієнтську частину, яка використовується операторами. Взаємодія

компонентів базується на трирівневій архітектурі. Клієнтська частина забезпечує зв'язок між оператором і Системою.

Для початку роботи оператор проходить реєстрацію, обирає оператора зв'язку для роботи та ініціює сеанс зв'язку з Центральним офісом. У процесі цього сеансу всі необхідні дані передаються на сервер офісу, після чого оператор отримує доступ до функціоналу Системи.

Передача документів до центру здійснюється через сеанс синхронізації з сервером. У цьому процесі документи, створені оператором, завантажуються на сервер, а дані клієнта, вже збережені на сервері, передаються на комп'ютер оператора. Після синхронізації оператор може переглядати деталізацію рахунку за будь-який вибраний період. Система формує перелік дзвінків абонента за вказаний час, включаючи їхню тривалість, і дозволяє роздрукувати цей список у вигляді документа стандартного формату. Під час синхронізації система запитує ім'я абонента для пошуку у базі даних. Якщо абонент не знайдений, Система інформує про це, і сеанс зв'язку не починається.

Таким чином, Система забезпечує ефективну взаємодію між оператором і центральним сервером, автоматизуючи основні процеси обслуговування абонентів. Завдяки функції синхронізації оператори мають доступ до актуальної інформації про клієнтів, що значно підвищує якість обслуговування. Перевірка правильності даних та автоматизація документообігу зменшують ризик помилок і оптимізують робочий процес. Це робить Систему надійним інструментом для сучасних телекомунікаційних компаній.

1.2 Методи аналізу та моделювання інформаційних систем

Моделювання підприємств можна здійснювати за допомогою структурних або об'єктно-орієнтованих методів, кожен з яких містить кілька конкретних методик. На сьогодні найпоширенішими є структурні методи, що дозволяють

аналізувати систему або процес, починаючи із загального огляду й поступово деталізуючи кожен аспект.

Основні особливості структурних методів включають:

– розбиття складної системи на компоненти, представлені як «чорні ящики», кожен з яких реалізує певну функцію;

– ієрархічну організацію елементів системи з визначенням зв'язків між ними;

– використання графічних засобів для візуалізації взаємозв'язків.

Моделі, побудовані за структурними методами, зазвичай подаються у вигляді ієрархічних діаграм, які графічно демонструють функції системи та їх взаємозв'язки. Завдяки цьому підвищується наочність і спрощується сприйняття моделі. Такі діаграми створюються за суворими правилами і підтримуються спеціальним програмним забезпеченням.

Значним розвитком структурного аналізу стала поява CASE-технологій (Computer-Aided Software/System Engineering), які об'єднують методології аналізу, проєктування та розробки складних систем із підтримкою програмних інструментів. Вони замінюють ручну роботу системних аналітиків і розробників комп'ютерними засобами автоматизації.

Серед поширених методологій структурного аналізу варто виокремити:

– SADT (Structured Analysis and Design Technique) та її стандарт IDEF0;

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

– ERD (Entity-Relationship Diagrams) – діаграми «сутність-зв'язок»;

– STD (State Transition Diagrams) – діаграми переходів станів.

У цій роботі використано методологію DFD (Data Flow Diagrams), яка представляє процес у вигляді мережі, з'єднаної потоками даних. Основні елементи DFD – це процеси, потоки даних та сховища, що дозволяють зберігати інформацію між процесами. DFD демонструє, як кожен процес перетворює вхідні дані на вихідні, а також показує взаємозв'язки між процесами.

Ключовим компонентом у DFD є контекстна діаграма, яка відображає систему у найзагальнішому вигляді, з акцентом на її взаємодію із зовнішніми сутностями через інформаційні потоки. Декомпозиція DFD дозволяє деталізувати

кожен процес, підвищуючи структуризацію даних, наочність і читабельність діаграм.

Для побудови контекстної діаграми системи «Центр обслуговування абонентів» необхідно визначити її основну функцію, зовнішні сутності, з якими вона взаємодіє, та зв'язки між ними. Контекстна діаграма (рис. 1.2) забезпечує базове розуміння функціонування системи..

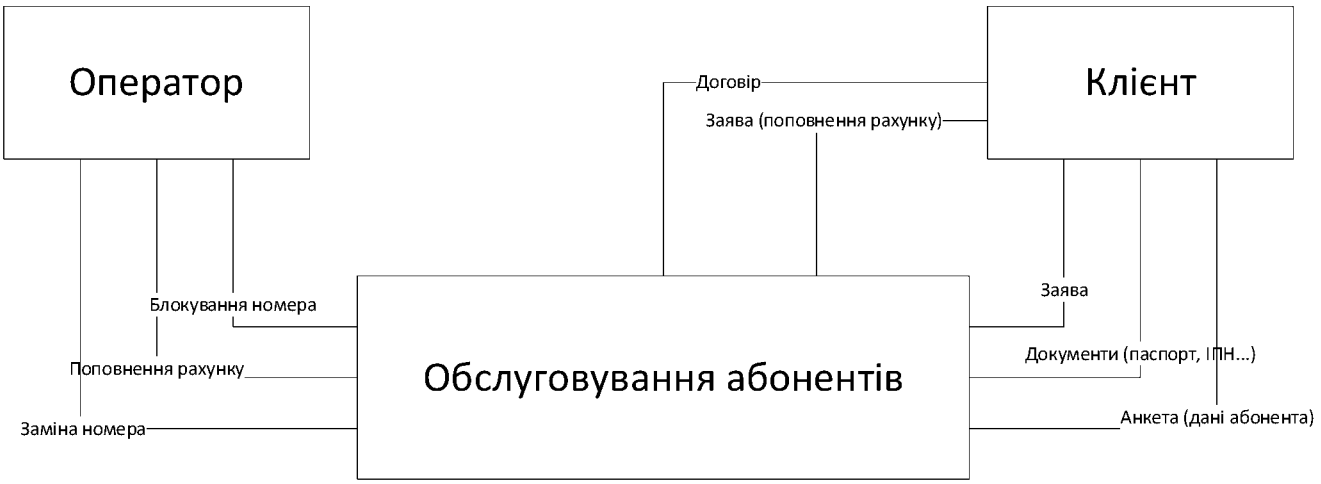


Рисунок 1.2 – Контекстна діаграма системи «Обслуговування абонентів»

Ключовим процесом функціонування системи є обслуговування абонентів, адже саме на нього спрямована вся діяльність системи. У процесі взаємодії із системою виділяються дві основні зовнішні сутності: Оператор і Клієнт. На початковому етапі клієнт надає необхідні документи та заповнює анкету, яка потім проходить перевірку оператором. У разі позитивного рішення укладається договір на надання послуг зв'язку. Якщо ж рішення негативне, клієнт отримує повідомлення про відмову.

На наступних етапах клієнт може звертатися до центру для заповнення заяв з різних питань, таких як деталізація рахунку, заміна абонентського номера, заміна SIM-карти чи поповнення рахунку. Оператор аналізує отримані заяви та ухвалює рішення: задовольнити запит чи відмовити у його виконанні. Ці процеси відображені на діаграмі потоків даних.

Створення контекстної діаграми стало відправною точкою для детального аналізу функцій системи та побудови діаграми деталізації першого рівня (рис. 1.3). Ця діаграма розкриває процеси, які можуть бути додатково декомпозовані в DFD нижчого рівня, формуючи ієрархію моделей, де контекстна діаграма займає кореневу позицію. Процес декомпозиції триває до того моменту, поки кожен елемент не стане можливим описати короткою мініспецифікацією (до однієї сторінки), яка детально пояснює обробку даних.

Рисунок 1.3 – Діаграма деталізації першого рівня системи «Обслуговування абонентів»

Обслуговування абонентів складається із семи базових функцій, які реалізуються в межах розроблюваної системи:

1. Розгляд анкети. Оператор розглядає анкету, заповнену клієнтом. Ця операція має два етапи: автоматичну перевірку системою та ручне доопрацювання оператором. Результатом є рішення щодо внесення інформації про абонента до системи.

2. Укладення договору. Успішна обробка анкети завершується укладенням договору та внесенням даних клієнта до системи.

3. Відмова в укладенні договору. У разі виявлення недоліків або порушення вимог оператор може відмовити клієнту у підписанні договору.

4. Підключення до мережі. Для реалізації цього етапу клієнт обирає оператора мережі, отримує SIM-карту та абонентський номер.

5. Розгляд заяви. У процесі користування послугами клієнт може подавати заяви, які стосуються різних аспектів обслуговування. Оператор перевіряє коректність заяви й ухвалює рішення.

6. Відмова у виконанні заяви. Якщо запит клієнта визнається недоцільним або таким, що не відповідає умовам договору, оператор повідомляє про відмову.

7. Виконання заяви. Після позитивного рішення заяву реалізують, забезпечуючи виконання вимог клієнта.

Методологія побудови діаграм потоків даних (DFD) дозволяє ефективно організувати процеси аналізу та декомпозиції системи. Графічне подання функціональних вимог і взаємозв'язків між елементами системи підвищує наочність і полегшує сприйняття моделі. Контекстна діаграма, як і наступні рівні деталізації, забезпечують чітке уявлення про функціонування системи та її взаємодію із зовнішнім середовищем. Грамотно структуровані моделі сприяють успішному впровадженню системи в реальних умовах, підвищуючи ефективність обслуговування клієнтів і точність виконання завдань.

Отже, побудова моделей інформаційних систем за допомогою структурних методів, зокрема діаграм потоків даних (DFD), дозволяє ефективно організувати процес декомпозиції та аналізу системи. Це підвищує наочність і полегшує розуміння функцій та взаємозв'язків між елементами системи. Використання таких методів забезпечує чітке структурування даних і процесів, що сприяє успішному впровадженню системи в реальних умовах.

1.3 Обґрунтування вибору інструментарію для створення UML-діаграм

PlantUML – потужний інструмент для створення UML-діаграм, що дозволяє моделювати програмні системи будь-якої складності.

Однією з головних переваг цього програмного засобу є можливість використовувати текстовий опис для створення діаграм UML, що робить процес

швидким та зручним. PlantUML фактично є текстовим редактором, який генерує графічні UML-діаграми на основі зрозумілого синтаксису.

У розпорядженні проєктувальника системи PlantUML надає підтримку таких типів діаграм, що дозволяють отримати повне уявлення про всю систему, яка проєктується, та її окремі компоненти:

– Use case diagram (діаграми прецедентів);

– Deployment diagram (діаграми розгортання);

– Statechart diagram (діаграми станів);

– Activity diagram (діаграми активності);

– Interaction diagram (діаграми взаємодії);

– Sequence diagram (діаграми послідовностей дій);

– Collaboration diagram (діаграми співпраці);

– Class diagram (діаграми класів);

– Component diagram (діаграми компонент).

PlantUML активно використовується для моделювання бізнес-процесів та програмних систем. Завдяки простому текстовому підходу він дозволяє моделювати систему до початку написання коду. Це дає змогу ще на етапі проєктування впевнитись в адекватності її архітектури та виявити потенційні недоліки. Завдяки цьому помилки виявляються на ранніх стадіях, коли їх виправлення потребує мінімальних витрат.

PlantUML дозволяє створювати сценарії використання та їх діаграми для візуалізації функціональних можливостей системи, використовуючи простий текстовий синтаксис.

PlantUML підтримується у багатьох сучасних програмних середовищах та редакторах. Серед них:

1. IntelliJ IDEA – інтеграція через плагіни дозволяє автоматично створювати UML-діаграми з тексту PlantUML.

2. Visual Studio Code – має плагін для роботи з PlantUML, що забезпечує автоматичну генерацію діаграм та їх перегляд у реальному часі.

3. Eclipse – плагін PlantUML додає функціональність створення UML-діаграм у цій IDE.

4. NetBeans – за допомогою відповідних розширень підтримує створення та візуалізацію діаграм PlantUML.

5. VSCodium – текстовий редактор з підтримкою PlantUML через розширення.

6. PlantText – онлайн-інструмент для створення діаграм на основі PlantUML.

7. WebSequenceDiagrams – спеціалізований сервіс для роботи з діаграмами послідовностей.

8. C4-PlantUML – бібліотека для створення C4-діаграм (архітектурних діаграм) у форматі PlantUML.

Ці програмні засоби роблять PlantUML зручним інструментом як для початківців, так і для професійних розробників, дозволяючи легко інтегрувати моделювання в сучасні робочі процеси.

Висновки до розділу 1

У першому розділі розглянуто сучасні моделі обслуговування абонентів у системах стільникового зв'язку, а також методи моделювання інформаційних систем. Проаналізовано ключові аспекти технологій стільникового зв'язку, зокрема впровадження автоматизованих систем для обслуговування клієнтів, що дозволяє знижувати витрати та підвищувати ефективність надання послуг. У роботі особлива увага приділена розробці імітаційної моделі програмного забезпечення інформаційної системи «Центр обслуговування абонентів», що включає в себе інтеграцію різних компонентів системи для обробки абонентських запитів, підключення та обслуговування клієнтів. Зазначено, що для оптимізації таких процесів доцільно використовувати методи імітаційного моделювання, оскільки вони дозволяють врахувати часові параметри та обмеження ресурсів, що є важливими для досягнення оптимального результату.

Визначено, що для побудови та аналізу інформаційних систем доцільно використовувати структурні методи, зокрема методологію DFD (Data Flow Diagrams), що дає змогу детально вивчити потоки даних і взаємозв'язки між різними компонентами системи. Діаграми потоків даних дозволяють наочно представити функціональні вимоги системи, що є необхідним для побудови ефективних та зрозумілих моделей, здатних забезпечити безперебійне обслуговування абонентів. Використання таких методів підвищує наочність процесів і допомагає краще зрозуміти, як саме інформація рухається через систему, а також як здійснюються різні операції, такі як укладення договорів або обробка заяв.

На основі проведеного аналізу сформульовано загальне завдання, яке полягає у розробці імітаційної моделі інформаційної системи. Загальне завдання розділено на три часткові задачі, дві з яких розглянуті у наступних розділах.

РОЗДІЛ 2
МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ЗА ДОПОМОГОЮ UML НОТАЦІЙ

2.1 Моделювання вимог до системи за допомогою діаграм варіантів використання

PlantUML і UML є універсальними інструментами, що ефективно використовуються для створення моделей бізнес-процесів та розробки діаграм варіантів використання (Use case diagrams). Цей тип діаграм дозволяє скласти перелік операцій, які виконує система, тому їх також називають діаграмами функцій. Завдяки таким діаграмам формується список вимог до системи та визначається набір її функцій.

Кожна діаграма варіантів використання (або Use case) описує сценарій взаємодії, якого дотримуються учасники (Actors). Цей тип моделювання використовується для опису бізнес-процесів у межах певної предметної області, що підлягає автоматизації, а також для визначення вимог до майбутньої системи. У діаграмах відображаються об'єкти як системи, так і предметної області, а також функції, які вони виконують.

Виділення суб'єктів взаємодії із системою здійснюється, відповідаючи на такі запитання:

– хто працює із системою або використовує її?

– хто передає або отримує інформацію в/із системи?

– хто перебуває поза межами системи, але взаємодіє з нею?

Прецеденти, або варіанти використання, є цілісними наборами функцій, що мають значення для суб'єкта. Їх можна визначити за допомогою запитання: «Які завдання виконує суб'єкт у межах системи і чого він очікує від неї?».

Кожен варіант використання задає послідовність дій, які виконує система під час взаємодії з актором. Водночас не деталізується, яким чином реалізовано цю

взаємодію чи виконання функцій. Прецеденти візуально зображаються у вигляді еліпсів, усередині або під якими зазначається їхня назва.

Рисунок 2.1 – Прецеденти системи «Обслуговування абонентів»

Документ, що описує прецеденти, має наступні ключові розділи:

1. Короткий опис.

2. Учасники (актори).

3. Передумови, необхідні для ініціювання прецеденту.

4. Деталізований опис потоку подій, який включає:

– Основний потік, що може бути розбитий на підпотоки для кращої читаємості документа.

– Альтернативні потоки для визначення виняткових ситуацій.

5. Післяумови, які визначають стан системи після завершення прецеденту.

Приклад документації прецедентів

Прецедент: Укладення договору

Короткий опис: Даний прецедент необхідний для реєстрації нового абонента в системі.

Суб’єкти: Оператор, клієнт.

Передумови: Оператор повинен ознайомити клієнта з наявними операторами зв’язку та надати форму анкети для заповнення.

Основний потік:

1. Клієнт обирає оператора та заповнює анкету.

2. Оператор перевіряє коректність заповнення анкети та вводить дані в систему.

3. Система запитує тип клієнта (фізична чи юридична особа) та відкриває відповідну реєстраційну форму.

4. Оператор вводить необхідні дані та зберігає договір, після чого система синхронізується із сервером.

– Альтернативний потік: У разі незаповнення всіх полів система видає повідомлення про помилку або дозволяє повторити спробу.

– Післяумови: Клієнта зареєстровано в базі даних системи.

Прецедент: Заміна абонентського номера

Короткий опис: Даний прецедент необхідний для зміни телефонного номера абонента.

Суб'єкти: Оператор, клієнт.

Передумови: Оператор має ознайомитися із заявою клієнта.

Основний потік:

1. Оператор знаходить абонента у базі даних, активує функцію «Заміна номера» та заповнює відповідну форму.

2. Після підтвердження система автоматично оновлює номер у базі даних, залишаючи інші дані без змін.

3. Зміни синхронізуються із сервером.

– Альтернативний потік: У разі незаповнення обов'язкових полів система видає повідомлення про помилку.

– Післяумови: Номер абонента успішно змінено в базі даних.

PlantUML дозволяє зручно моделювати бізнес-процеси та системи на етапі проєктування, що сприяє виявленню можливих помилок ще до написання коду. Завдяки текстовому підходу він забезпечує ефективну взаємодію між учасниками проєкту, полегшуючи створення та візуалізацію складних систем. Це робить PlantUML універсальним інструментом для різних галузей автоматизації.

2.2 Розробка діаграми прецендентів

Діаграми прецедентів дозволяють визначити перелік операцій, які виконує система. Їх також називають діаграмами функцій, адже вони формують основу для створення списку вимог до системи та опису її функціональних можливостей. Кожна така діаграма (або Use case) є описом сценарію поведінки, який реалізується за участю певних суб'єктів (Actors).

Діаграма прецедентів асоціює прецеденти із суб'єктами, а також надає можливість встановлювати зв'язки між прецедентами, якщо такі існують. Вона є графічним представленням суб'єктів, прецедентів та їхніх взаємозв'язків, доповненим відповідними специфікаціями. На діаграмі відображаються ключові функції системи, зовнішні суб'єкти, які взаємодіють із системою, і зв'язки між ними. У результаті діаграма прецедентів є не просто графічною схемою, а повноцінною документованою моделлю передбачуваної поведінки системи.

Основні переваги використання моделі варіантів використання:

– визначає користувачів системи та окреслює її межі;

– описує інтерфейс системи;

– полегшує комунікацію між користувачами та розробниками;

– використовується для розробки тестових сценаріїв;

– є основою для створення користувацької документації;

– підходить для різних підходів до проєктування, включаючи об'єктно-орієнтовані та структурні методи.

На діаграмі прецедентів можна представити суб'єктів та відповідні їм варіанти використання. Наприклад, на рисунку 2.2 зображено, як оператор ініціює такі дії, як розгляд заяв, заміна SIM-карти, деталізація рахунку, блокування номера тощо. Клієнт також виконує свої функції, наприклад, заповнення анкети чи вибір оператора зв'язку.

Взаємозв'язки між варіантами використання

– Include: Відношення, за якого один прецедент включає виконання іншого. Наприклад, підключення абонента передбачає вибір оператора зв'язку.

– Extend: Відношення, що додає до основного прецеденту додаткові можливості за певних умов. Наприклад, укладання договору можливе лише після розгляду анкети оператором, а розгляд заяви розширює прецеденти, такі як блокування номера чи заміна SIM-карти.

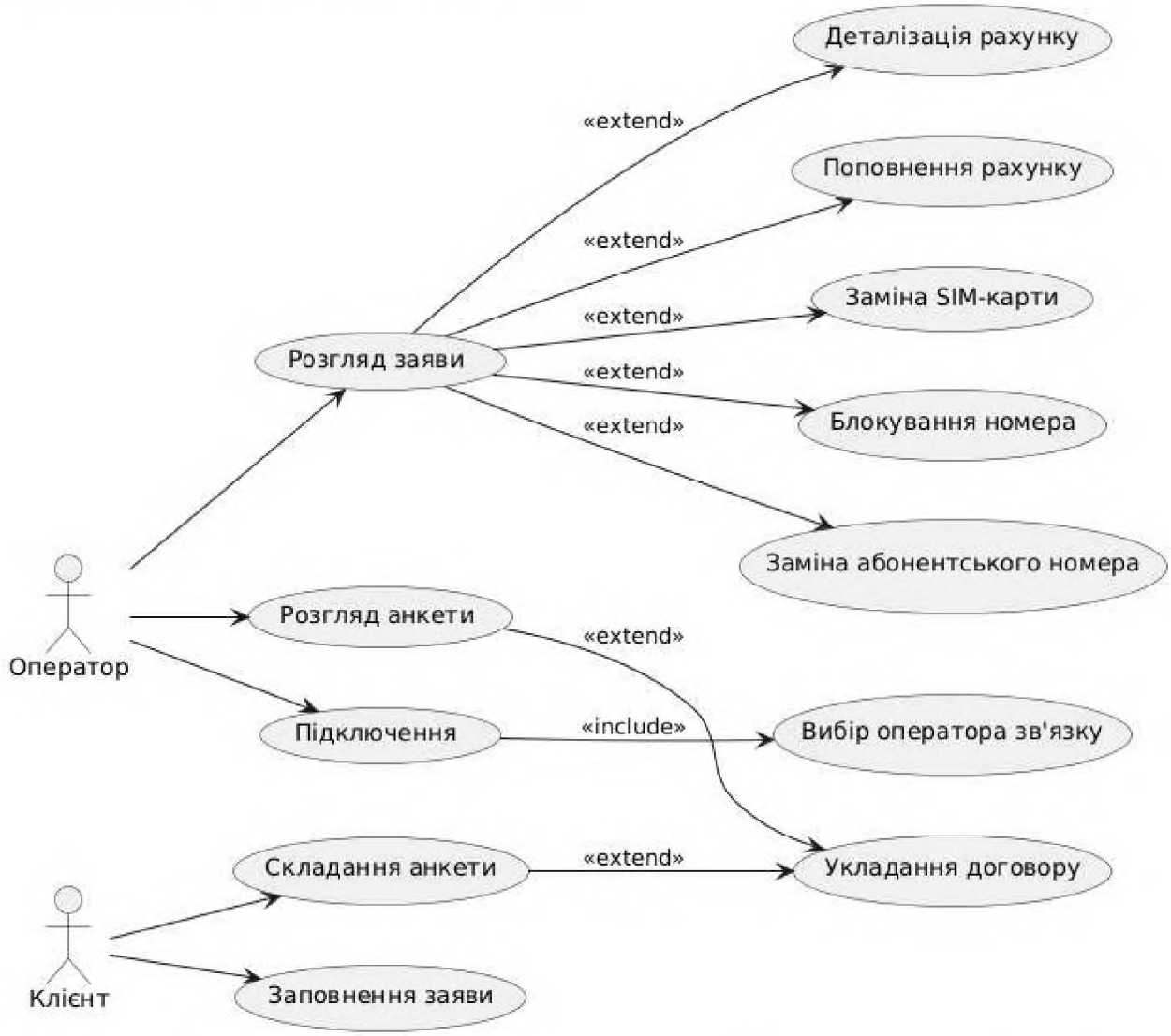


Рисунок 2.2 – Діаграма прецедентів системи «Обслуговування абонентів».

На рисунку 2.3 і рисунку 2.4 наведено детальні приклади відносин для суб'єктів «Оператор» та «Клієнт» відповідно. Зупинимося детальніше на деяких відносинах між варіантами використання.

Розглянемо деякі ключові відносини між варіантами використання, представлені на рис. 2.3. Зв'язок типу «include» означає, що один прецедент є

обов'язковою частиною іншого. Наприклад, підключення абонента обов'язково передбачає вибір оператора зв'язку.

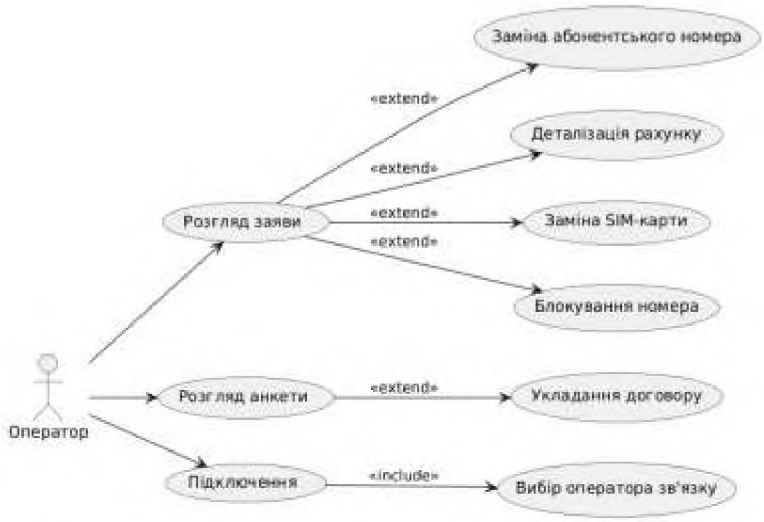


Рисунок 2.3 – Діаграма прецедентів для суб'єкта «Оператор».

Зв'язок «extend» вказує на те, що базовий прецедент може бути доповнений іншим за певних умов. Наприклад, розгляд анкети розширюється прецедентом укладання договору, оскільки укласти договір можливо лише після перевірки анкети оператором. Аналогічно, розгляд заяви може розширювати такі прецеденти, як блокування номера, заміна SIM-карти, деталізація рахунку або зміна абонентського номера. Отже, зв'язок «extend» використовується для позначення додаткових дій, які виконуються залежно від конкретної ситуації.

Рисунок 2.4 – Діаграма прецедентів для суб'єкта «Клієнт».

Підключення абонента включає вибір оператора зв'язку (рис. 2.4). Складання анкети «розширюється» варіантом використання укладання договору.

Діаграми прецедентів є потужним інструментом для моделювання поведінки системи. Вони забезпечують чіткий поділ ролей і функцій, сприяють кращому розумінню меж системи та її інтерфейсів. Завдяки такому підходу полегшується комунікація між розробниками та замовниками, що сприяє ефективній реалізації системи.

2.3 Особливості проєктування класів

Під час моделювання варіантів використання паралельно виконується ідентифікація так званих класів-сутностей. Ці класи разом із їх атрибутами та взаємозв'язками подаються у вигляді діаграми класів (Class diagram). Діаграма класів слугує для моделювання статичного представлення системи з точки зору її проєктування, формуючи логічну модель системи. У цій діаграмі не враховуються часові аспекти функціонування. Кожен клас аналізується в контексті кількох функціональних вимог.

Клас у мові UML є позначенням множини об'єктів із однаковою структурою, поведінкою та відношеннями з іншими класами. Клас-сутність визначає типи об'єктів системи та зв'язки між ними. Такі класи відіграють ключову роль у системі, оскільки вони зберігають частину бази даних, доступ до якої можливий для зчитування, зміни або оновлення. Аналіз предметної області дозволяє виявити класи-сутності. Зокрема, було визначено такі класи: Клієнт, Фізична особа, Юридична особа, Договір, Заява, Оператор. Вони подані на рисунку 2.5.

Клас «Клієнт» забезпечує збереження інформації про клієнтів системи (абонентів). До атрибутів цього класу належать: індекс, країна, місто, область, район, вулиця, будинок, корпус, квартира, e-mail, факс, телефон. Оскільки клієнтом може бути як фізична особа, так і організація, було створено класи «Фізична особа»

та «Юридична особа», що пов'язані з класом «Клієнт». Зв'язок між ними зображено на діаграмі класів.

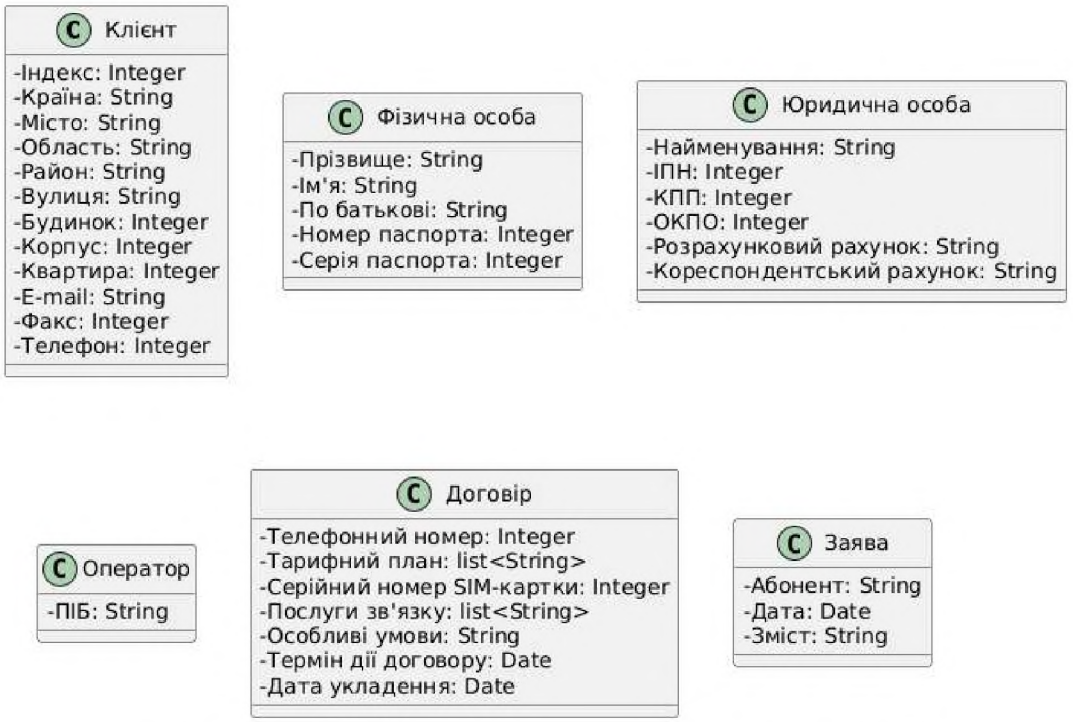


Рисунок 2.5 – Початковий вигляд класів-сутностей

Клас «Договір» є місцем збереження договорів про надання послуг зв'язку. Його атрибути включають: номер телефону, тарифний план, серійний номер SIM-картки, послуги зв'язку, особливі умови, термін дії договору, дату укладення. Послуги зв'язку описуються типом list<integer>, що дозволяє клієнту обирати одну або кілька послуг зі списку. Номер телефону представлений типом integer, оскільки складається з чисел.

Класи зазвичай не існують ізольовано, а взаємодіють між собою. На рисунку 2.6 показана діаграма класів, де зв'язки між класами «Фізична особа» – «Клієнт» та «Юридична особа» – «Клієнт» реалізовані як узагальнення. Узагальнення є відношенням між суперкласом (загальним класом) і підкласом (специфічним класом). Підклас успадковує атрибути й методи суперкласу. У цьому контексті «Клієнт» виступає як суперклас, тоді як «Фізична особа» і «Юридична особа» є підкласами, які успадковують усі атрибути класу «Клієнт».

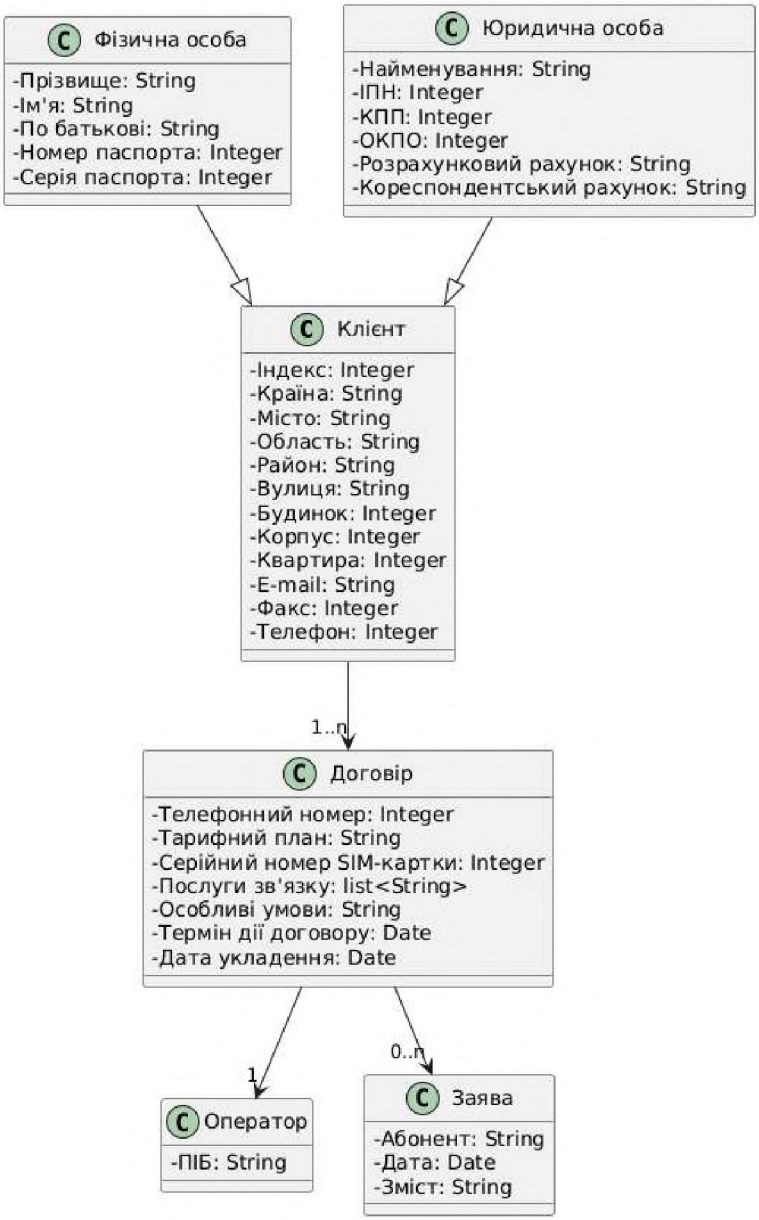


Рисунок 2.6 – Діаграма класів та зв’язків між ними

Ключовим елементом моделі є клас «Договір». На діаграмі показано зв’язки «Клієнт – Договір» і «Договір – Оператор». Кратність асоціацій пояснюється таким чином: клієнт може створити один або кілька документів ([1..n]), і кожен документ має бути введений оператором у систему. Клас «Договір» забезпечує збереження всіх договорів із однаковою структурою. Різниця між ними визначається атрибутом «тип платіжного документа».

Клас «Заява» відображає можливість створення клієнтом заяви за певних умов, однак це не є обов’язковим ([0..1]). Таким чином, побудова діаграми класів-сутностей дозволила визначити типи об’єктів для майбутньої бази даних і зв’язки

між ними. Проте ця діаграма не є повною статичною моделлю системи, оскільки в ній відсутні керуючі та інтерфейсні класи, необхідні для повноцінного моделювання.

Таким чином, формування діаграми класів-сутностей завершено. Було визначено типи об'єктів, які формуватимуть майбутню модель бази даних, а також зв'язки між ними. Однак створену діаграму класів не можна назвати повною статичною моделлю системи через відсутність таких важливих елементів, як керуючі та інтерфейсні класи.

2.4 Моделювання видів діяльності інформаційної системи

Діаграми видів діяльності (Activity Diagrams) використовуються для зображення загальної послідовності дій, які виконуються кількома об'єктами або варіантами використання. На цих діаграмах графічно відображено переходи потоку управління від однієї діяльності до іншої всередині системи. Цей вид діаграм відноситься до динамічних представлень системи та є надзвичайно корисним для моделювання функціонування, оскільки показує передачу потоку управління між об'єктами.

Ключовим елементом таких діаграм є діяльність. Інтерпретація цього терміна варіюється залежно від перспективи створення діаграми. Наприклад, на концептуальному рівні діяльність визначається як завдання, що виконується вручну або автоматизованим способом. У разі, коли діаграма створюється з позиції специфікації чи реалізації, діяльність відповідає певному методу для класу.

Діаграми видів діяльності забезпечують гнучкість у визначенні порядку виконання завдань, що особливо важливо для моделювання бізнес-процесів, які часто включають дії, виконання яких може бути як послідовним, так і паралельним. Така властивість сприяє ефективному відображенню складних бізнес-процесів.

Завдяки своїй здатності моделювати паралельні дії, ці діаграми також корисні у паралельному програмуванні. На них можна чітко показати всі гілки виконання

процесу та визначити момент їх синхронізації. Наприклад, синхронізаційна лінійка на діаграмі вказує, що вихідна діяльність починається лише після завершення кількох вхідних дій.

Ще одним важливим застосуванням діаграм видів діяльності є представлення потоків подій у варіантах використання. Хоча текстовий опис потоків подій може бути досить детальним, у разі складних сценаріїв із численними альтернативами графічне подання значно полегшує розуміння логіки процесів. Діаграми видів діяльності містять ту саму інформацію, що й текстовий опис, але подають її у зручній та зрозумілій формі.

Рисунок 2.7 – Діаграма для сценарію «Заповнення анкети»

Цей сценарій передбачає, що клієнт отримує анкету від оператора, заповнює всі її поля та передає оператору для перевірки (рис. 2.7). Якщо оператор знаходить помилки, клієнту видається нова форма для повторного заповнення. У разі правильно заповненої анкети укладається договір на надання послуг зв’язку.

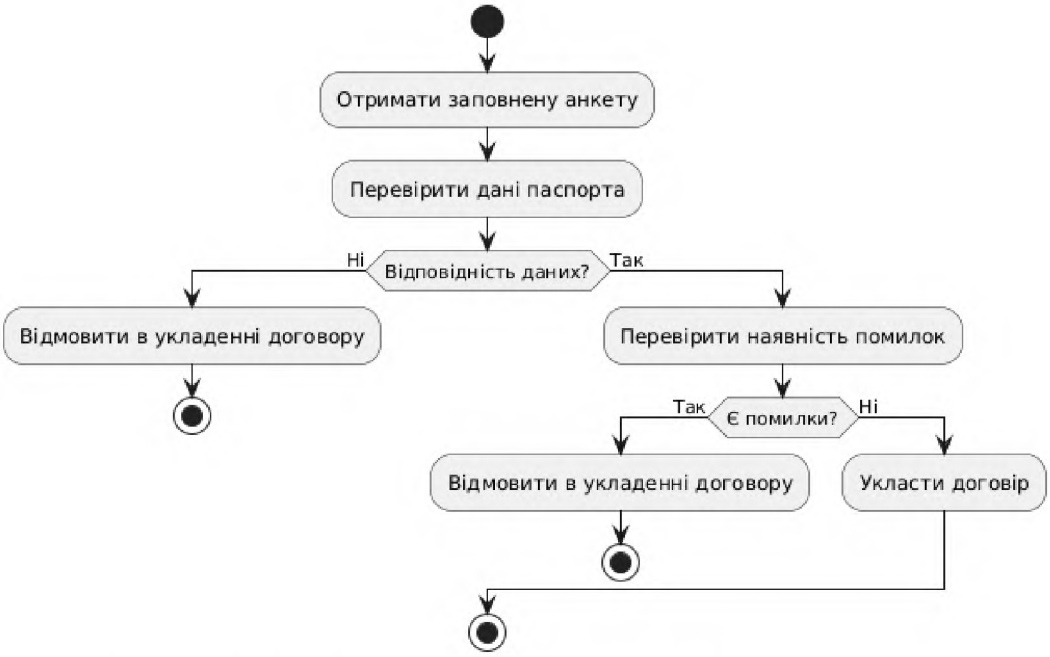


Рисунок 2.8 – Діаграма для сценарію «Розгляд анкети»

Оператор отримує заповнену клієнтом анкету, перевіряє всі її дані (рис. 2.8). Якщо помилки виявлені, клієнту відмовляють в укладенні договору. За відсутності помилок договір укладається.

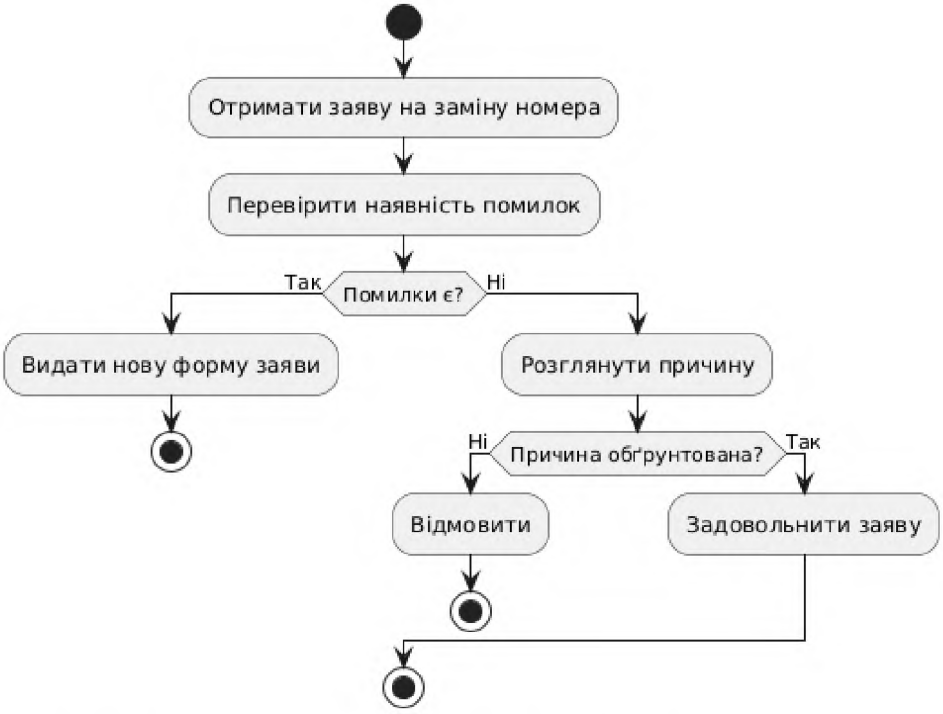


Рисунок 2.9 – Діаграма для сценарію «Заміна абонентського номера»

Сценарій починається із заяви клієнта на заміну номера. Оператор перевіряє правильність заповнення заяви (рис. 2.9). За наявності помилок клієнту видається нова форма для виправлення. Якщо заява заповнена правильно, розглядається причина заміни номера. У результаті причину або схвалюють, або відхиляють.

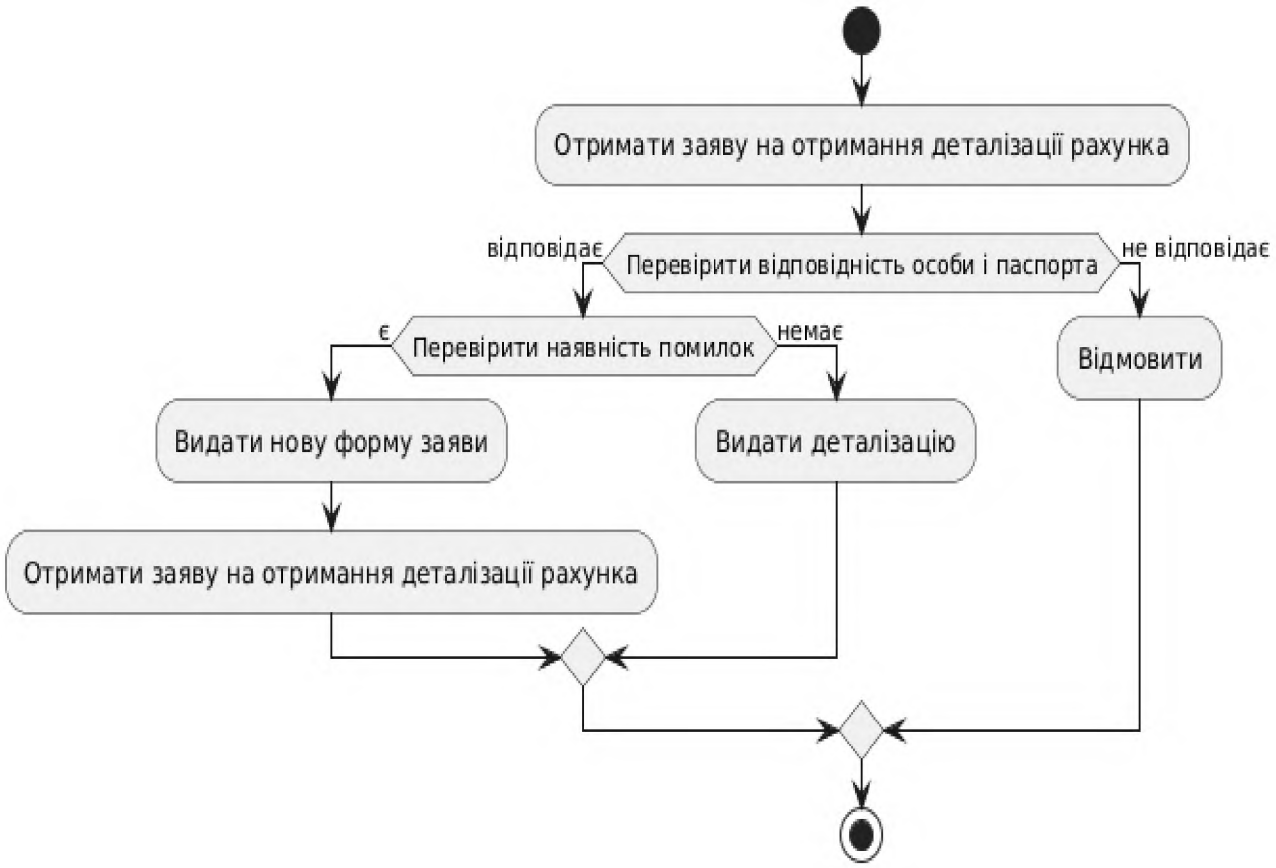


Рисунок 2.10 – Діаграма для сценарію «Деталізація рахунку»

Клієнт подає заяву на деталізацію рахунку, після чого оператор перевіряє відповідність особи та паспорта (рис. 2.10). Якщо документ не належить клієнту, запит відхиляється. За відсутності помилок у заяві клієнт отримує деталізацію рахунку за вказаний період. У разі помилок видається нова форма.

Клієнт подає заяву на заміну SIM-картки. Оператор перевіряє відповідність особи та паспорта клієнта (рис. 2.11). Якщо документи не відповідають, клієнту відмовляють. У разі відсутності помилок у заяві розглядається причина заміни, після чого картка змінюється, якщо причина є обґрунтованою.

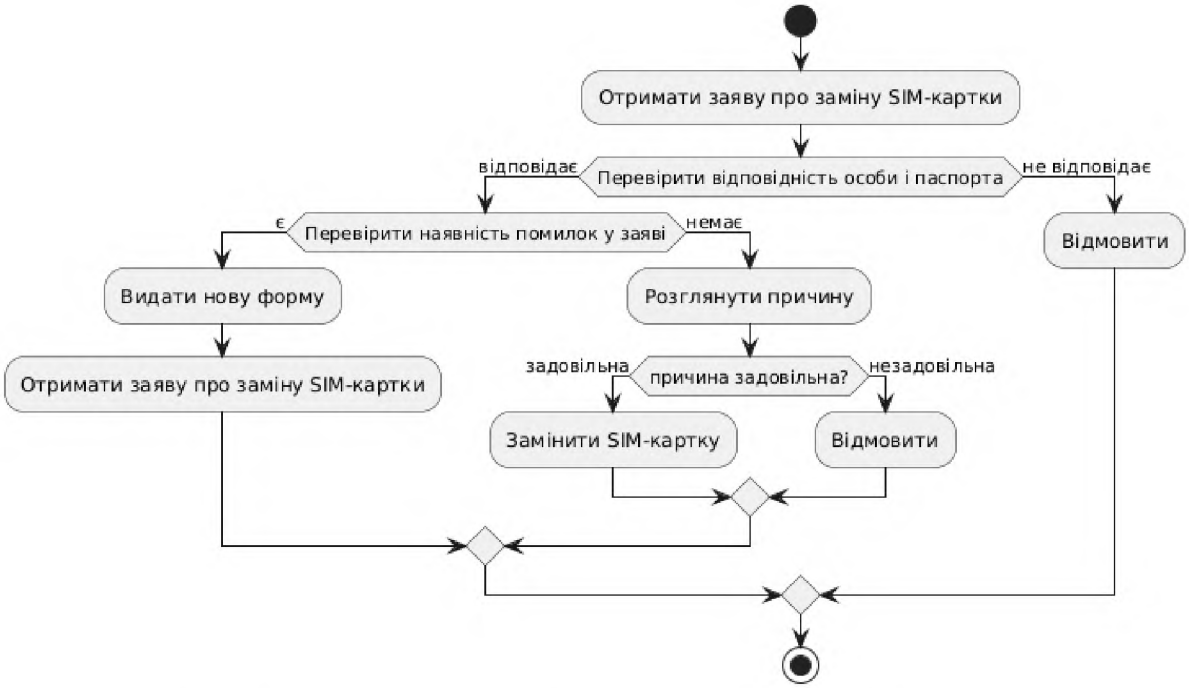


Рисунок 2.11 – Діаграма для сценарію «Заміна SIM-картки»

Рисунок 2.12 – Діаграма для сценарію «Вибір оператора зв’язку»

Клієнт аналізує доступні варіанти операторів і їхні тарифи (рис. 2.12). Після вибору підходящого оператора укладається договір, що дозволяє абоненту підключитися до мережі.

Таким чином, діаграми видів діяльності є ефективним засобом для моделювання процесів, що забезпечує їх структуроване, наочне та зрозуміле подання.

2.5 Моделювання взаємодій між об'єктами

Взаємодія між об'єктами в системі відбувається через прийом і передачу повідомлень, що надсилаються або отримуються об'єктами-клієнтами, а також їх обробку об'єктами-серверами. Щоб зафіксувати послідовність передачі повідомлень між об'єктами, можна побудувати модель динамічної взаємодії для кожного прецеденту використання. Модель взаємодії зазвичай подається у двох формах:

1. Діаграма послідовності (sequence diagram), яка фокусується на тимчасовій упорядкованості подій. На таких діаграмах зображуються об'єкти і повідомлення, що ними надсилаються чи отримуються. Зазвичай об'єкти на діаграмі представлені як екземпляри класів.

2. Діаграма кооперації (collaboration diagram), що акцентує увагу на структурі об'єктів, які приймають або надсилають повідомлення. Вона показує безліч об'єктів, їх взаємозв'язки та потоки повідомлень між ними.

Діаграми послідовностей і кооперації є частинами динамічної моделі системи. Вони є ізоморфними, тобто можна перетворювати одну форму в іншу без втрати інформації. Тому в даній роботі буде розглянута лише діаграма послідовності, оскільки з неї при потребі можна отримати діаграму кооперації.

Ключовим аспектом діаграми послідовності (рис. 2.13) є відображення динаміки взаємодії об'єктів в часі. Діаграма має два виміри: перший – горизонтальний, що зображує лінію життя (lifeline) кожного об'єкта, який бере участь у взаємодії, а другий – вертикальний, що є часом, позначеним віссю, спрямованою зверху вниз. Взаємодія між об'єктами здійснюється через

повідомлення, які передаються в тому порядку, в якому вони з'являються на діаграмі, тобто зверху вниз.

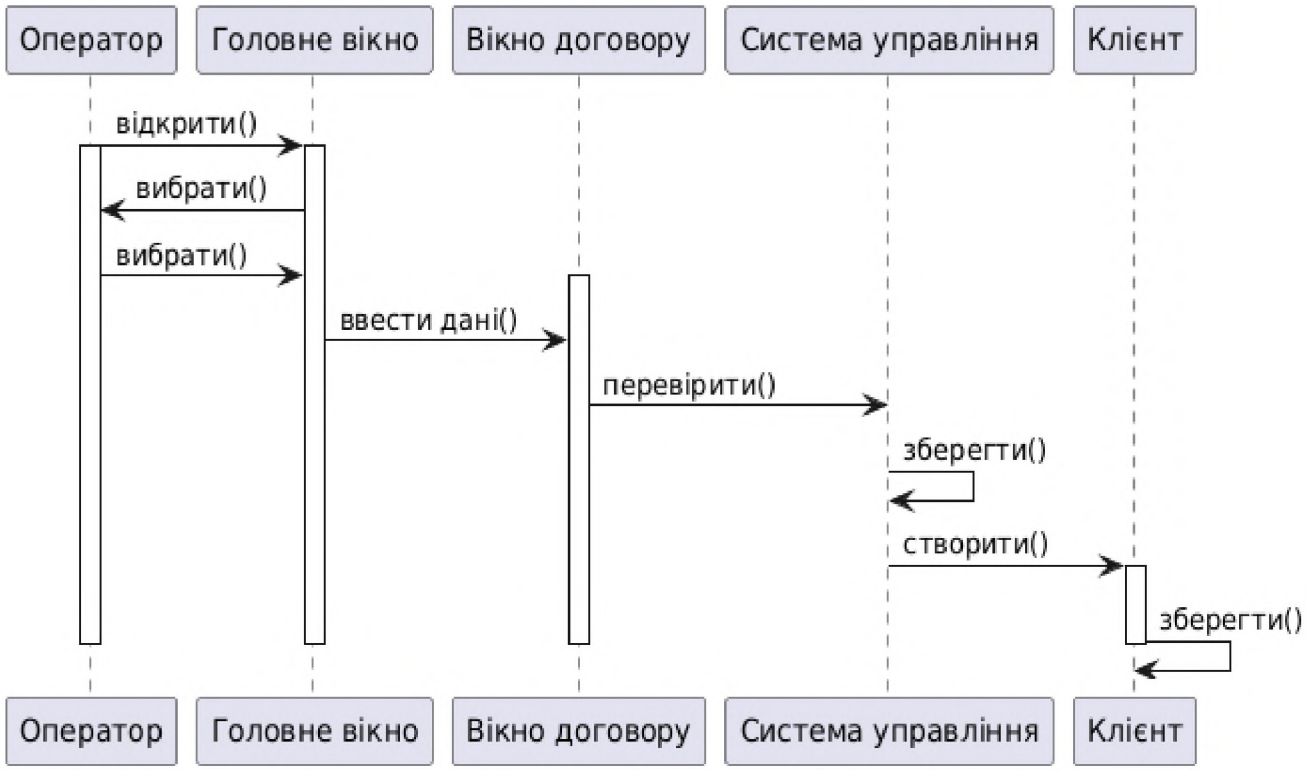


Рисунок 2.13 – Діаграма послідовності «Завершення договору»

На рис. 2.14 показано діаграму кооперації для процесу укладання договору.

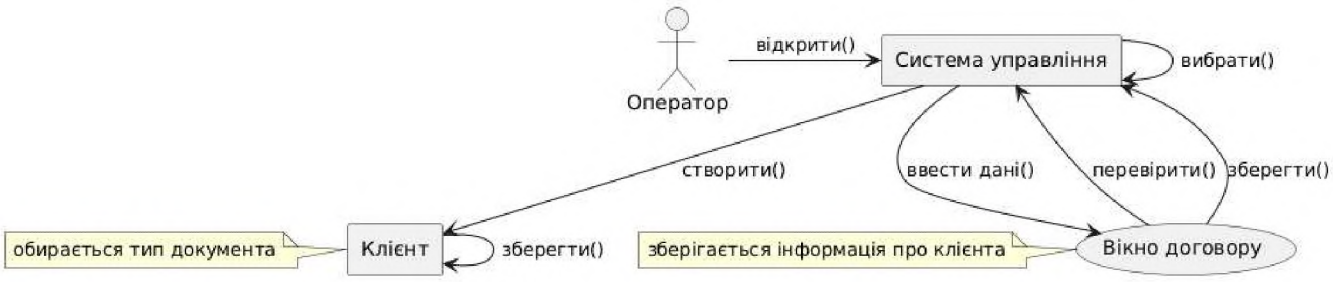


Рисунок 2.14 – Діаграма кооперації для укладання договору

Рисунок 2.15 ілюструє діаграму послідовності для процесу деталізації рахунку. Після завершення договору клієнт передає підписаний договір оператору, який відкриває головне вікно системи. Оператор вибирає реєстрацію, вводить дані в прикордонний клас Вікно реєстрації, де формується договір. Система перевіряє

введену інформацію та зберігає договір. Інформація про нового клієнта зберігається в контейнерному класі «Клієнт». Оператор відкриває вікно документа «Деталізація рахунку», формує звіт, передаючи повідомлення до класу «Система управління» для застосування критеріїв часу. Відображений список дзвінків можна надрукувати.

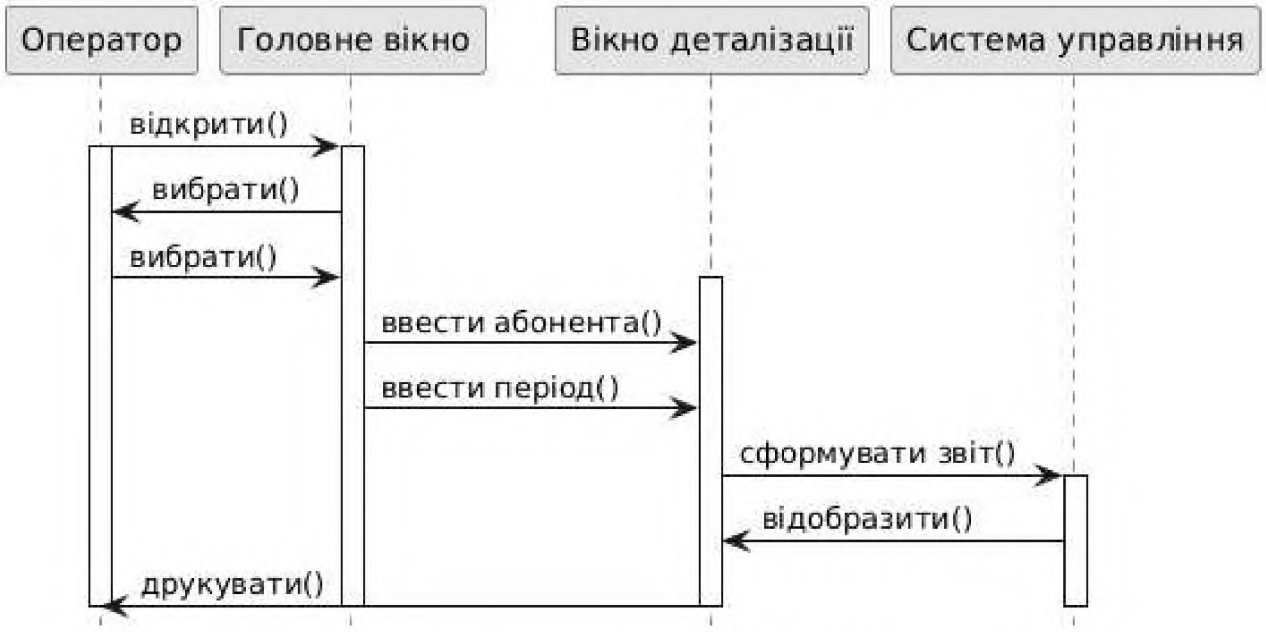


Рисунок 2.15 – Діаграма послідовності Деталізація рахунку

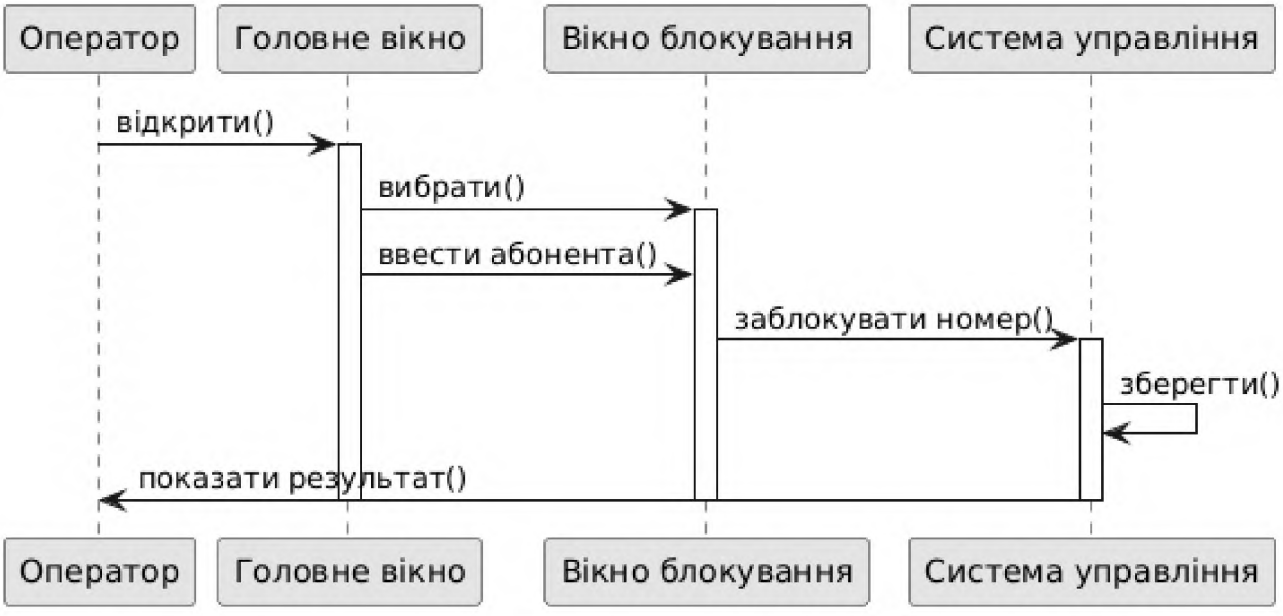


Рисунок 2.16 – Діаграма послідовності Блокування номера

Рисунок 2.16 демонструє діаграму послідовності для процесу блокування номера. Оператор відкриває головне вікно системи, потім вибирає вікно блокування номера, передає повідомлення класу «Система управління», яка блокує номер, зберігає операцію та повідомляє про результат. На рис. 2.17 зображена діаграма кооперації для цього процесу.

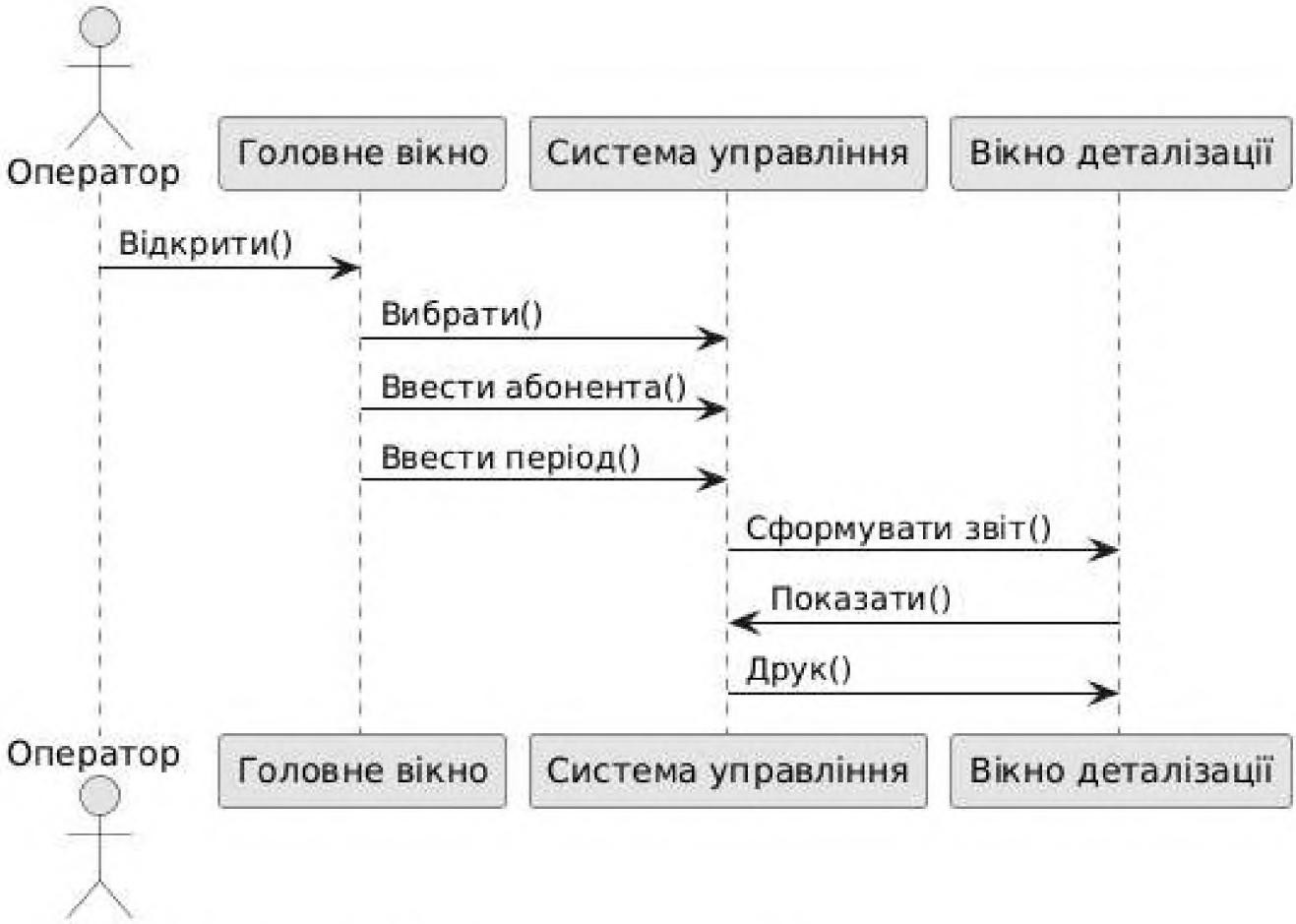


Рисунок 2.17 – Діаграма кооперації Блокування номера

Таким чином, моделювання взаємодій за допомогою діаграм послідовностей і кооперацій дає змогу ефективно відслідковувати процеси обміну повідомленнями в системі. Це допомагає візуалізувати динаміку взаємодії між об’єктами, структуру зв’язків і спрощує аналіз поведінки системи на різних етапах її функціонування. Завдяки можливості взаємного перетворення цих діаграм, вибір однієї з форм не обмежує подальші можливості для детальнішого аналізу та вивчення роботи системи.

2.6 Моделювання станів інформаційної системи

Наступним етапом розробки є процес моделювання станів, для якого необхідна інформація міститься в моделі класів-сутностей та моделі взаємодій. Модель класів ілюструє класи, що підлягають аналізу з метою виявлення особливостей їхньої поведінки, тоді як модель взаємодій допомагає визначити методи, що можуть впливати на зміну станів класів.

Стан можна визначити як конкретну ситуацію, коли виконується певна умова. Він може бути заданий у вигляді набору конкретних значень атрибутів класу або об'єкта. Зміна значень цих атрибутів відображатиме зміну стану класу або об'єкта, що моделюється. Важливо зазначити, що не кожен атрибут класу може бути використаний для характеристики його стану. Зазвичай лише ті властивості елементів системи, які відображають динамічні або функціональні аспекти їхньої поведінки, є важливими для опису стану.

Діаграми станів (Statechart Diagram) визначають всі можливі стани, в яких може перебувати конкретний об'єкт, а також процеси зміни цих станів внаслідок настання певних подій. У більшості об'єктно-орієнтованих методів діаграми станів розробляються для одного класу і показують динаміку поведінки цього класу, тобто, як змінюється стан окремого об'єкта.

Не варто створювати діаграми станів для кожного класу в системі; їх доцільно використовувати тільки для тих класів, чия поведінка є суттєвою для розуміння системи, а створення таких діаграм допомагає краще зрозуміти процеси, що відбуваються. Аналізуючи розроблювану систему та її вимоги, можна зазначити, що поведінка класу «Договір» є важливою для розробника, тому доцільно розглянути діаграму станів для цього класу (рис. 2.18).

Перш ніж договір буде створений, він має стан «Новий». Після того як обидві сторони підпишуть його, договір переходить у стан «Підписаний». Він залишається в цьому стані до завершення терміну дії, після чого може змінити свій стан на «Продовжений», «Протермінований» або «Розірваний», а в кінцевому підсумку перейде в стан «Видалений».

Рисунок 2.18 – Діаграма станів для класу «Договір»

Щодо класу «Заява», перед створенням і в процесі її оформлення заява має стан «Нова» (рис. 2.19). Після того, як клієнт підписує заяву, вона переходить у стан «Заповнена». При передачі заяви оператору вона набуває стану «На розгляді». Якщо виявлено помилки на етапі розгляду, заява переходить у стан «Відхилена», інакше – в стан «На обробці». Якщо заява задоволена, вона переходить у стан «Виконана», у протилежному випадку – в стан «Відхилена».

Рисунок 2.19 – Діаграма станів для класу «Заява»

Моделювання станів є важливим етапом у розробці інформаційних систем, оскільки дозволяє точно визначити процеси, які відбуваються при зміні станів об’єктів. Це допомагає розробникам зрозуміти динаміку роботи системи та забезпечити правильне управління її функціями. Завдяки такому підходу можна ефективно прогнозувати поведінку системи в різних ситуаціях і знизити ймовірність помилок при її реалізації.

2.7 Проєктування статичної структури, діаграми компонентів та розгортання

Статична структура інформаційної системи є кінцевою діаграмою класів, що демонструє взаємодію між сутностями, а також керуючими та граничними класами, виявленими на етапі початкового проєктування та під час моделювання взаємодій. Модель взаємодій дає інформацію не лише про те, які класи повинні бути присутні в системі, окрім сутностей, визначених на старті розробки, а й про способи їх взаємодії та взаємозв'язків, а також які методи їм властиві. На наведених діаграмах статична структура розділена на дві частини. Перша (рис. 2.20) показує взаємодію керуючого класу з сутностями, а друга (рис. 2.21) – взаємодію керуючого класу з граничними класами.

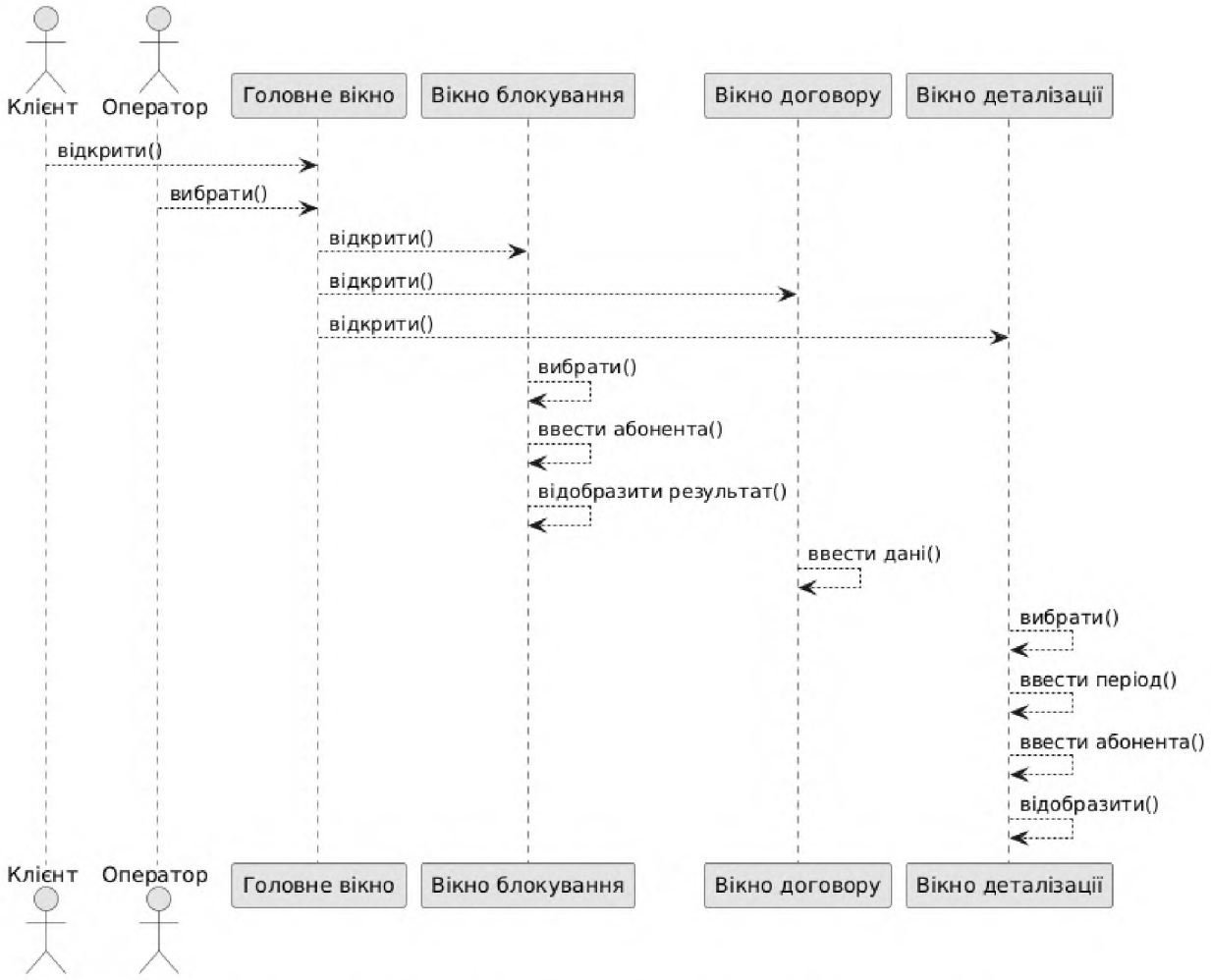


Рисунок 2.20 – Статична структура ІС (частина 1)

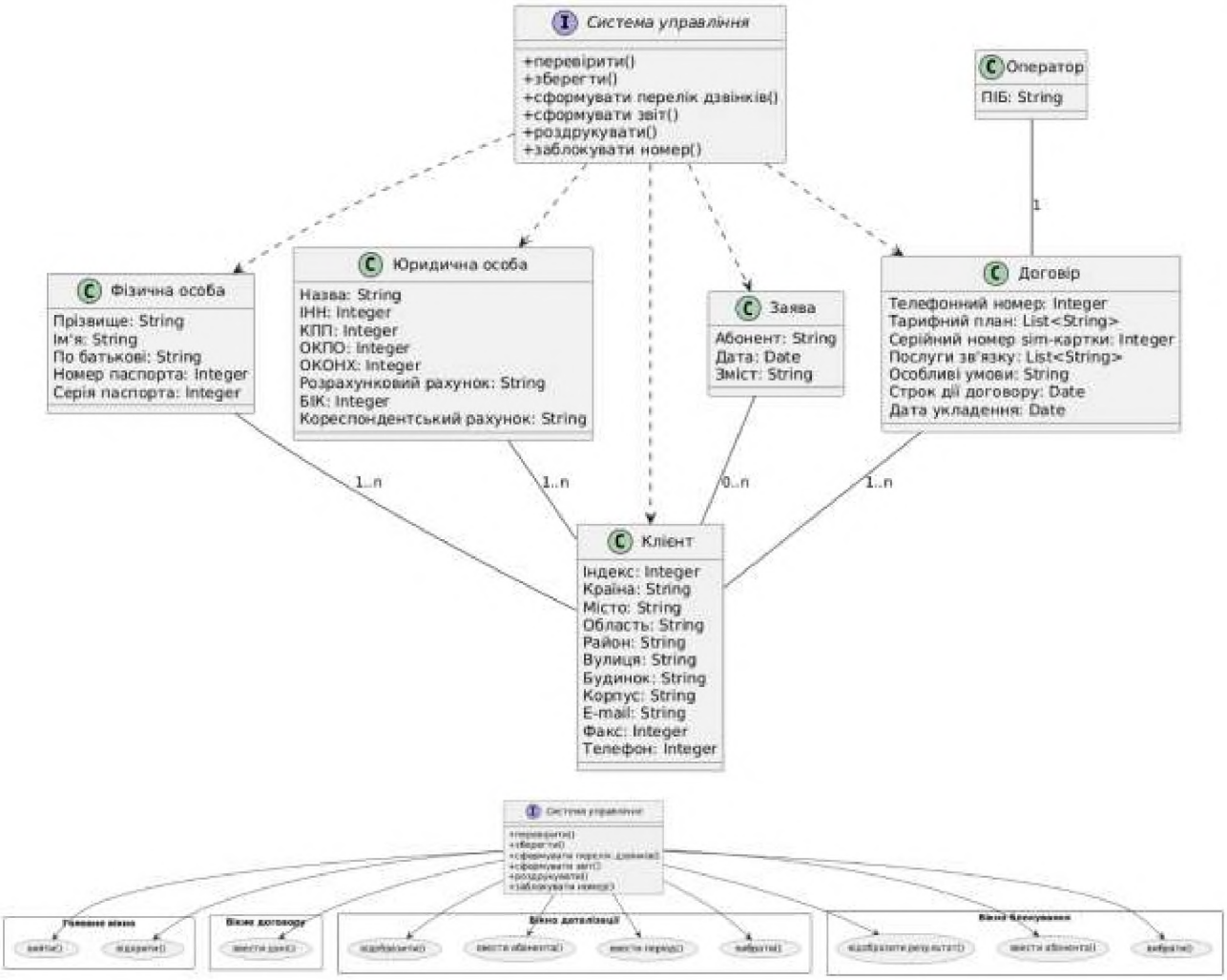


Рисунок 2.21 – Статична структура ІС (частина 2)

Не всі моделі послідовностей і кооперації були створені з урахуванням принципу трирівневої взаємодії, що, можливо, є певним недоліком цієї розробки, тому на кінцевій діаграмі класів мають бути присутні зв'язки між деякими граничними класами та сутностями. Однак, оскільки відображення всіх цих зв'язків на одній діаграмі значно ускладнило б її сприйняття, така діаграма не наводиться в роботі.

Для створення конкретної фізичної системи необхідно перетворити всі елементи логічного представлення на реальні матеріальні сутності. Для опису цих реальних сутностей призначене фізичне представлення моделі. Основними елементами фізичного представлення є компоненти, які в нотації UML представляють собою фізично існуючі частини системи, що забезпечують реалізацію класів, їхніх відносин та функціональної поведінки програмної системи,

що моделюється. На рис. 2.22 показано спробу визначення складу компонентів системи і представлення їх у вигляді діаграми компонентів.

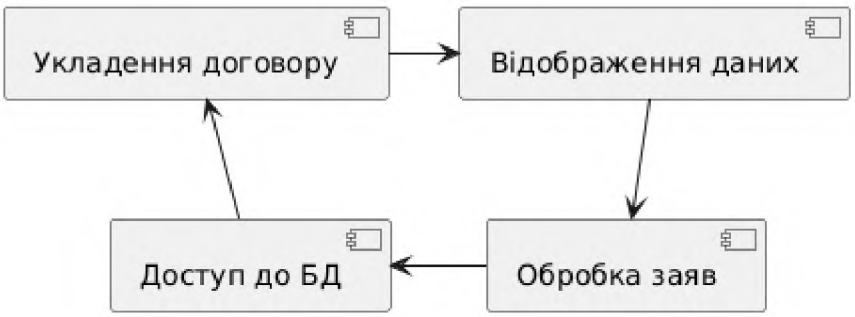


Рисунок 2.22 – Діаграма компонентів

Технологія «клієнт-сервер» є однією з основ, на яких побудована сучасна комп'ютерна мережа, і займає важливе місце в її розвитку. Для створення архітектури цього застосунку використано саме цю технологію. Коли йдеться про вибір між двошаровою або тришаровою архітектурою, необхідно ретельно оцінити обидва варіанти. Термін «клієнт-сервер» позначає архітектуру програмного комплексу, в якій його функціональні частини взаємодіють за принципом «запит-відповідь». Одна з частин, клієнт, виконує активну роль, ініціюючи запити, а інша частина, сервер, на них відповідає. У міру розвитку системи ролі можуть змінюватися: наприклад, певний програмний блок може одночасно виконувати функції як сервера для одного блоку, так і клієнта для іншого.

Варто зазначити, що кожна інформаційна система повинна містити мінімум три основні функціональні частини: модулі зберігання даних, обробки даних і взаємодії з користувачем (в даному випадку з оператором). Кожну з цих частин можна реалізувати окремо. Наприклад, змінюючи інтерфейс оператора, можна зберігати дані та алгоритми обробки без змін, при цьому відображати ті самі дані у вигляді таблиць, графіків або гістограм. Аналогічно, зміна алгоритмів обробки даних не потребує змін у програмах для зберігання або відображення даних. Крім того, можна змінити програмне забезпечення для зберігання даних, замінивши файлову систему, не змінюючи при цьому програми для обробки та відображення даних.

У класичній архітектурі «клієнт-сервер» три основні частини застосунку зазвичай розподіляються між двома фізичними модулями. Програмне забезпечення для зберігання даних зазвичай розташоване на сервері (наприклад, на сервері бази даних), інтерфейс користувача – на стороні клієнта, а обробка даних може здійснюватися як на стороні клієнта, так і на сервері. Однак двошарова архітектура має низку недоліків, що створюють певні складнощі в розробці клієнт-серверних систем.

При поділі алгоритмів обробки даних необхідно синхронізувати діяльність обох частин системи. Усі розробники повинні мати доступ до останніх змін у системі та розуміти ці зміни. Це ускладнює процес розробки, впровадження та підтримки клієнт-серверних систем, оскільки вимагає значних зусиль для координації роботи різних груп фахівців. Несумісність між розробниками затримує розвиток системи і змушує вносити зміни до вже перевірених і готових компонентів.

Щоб уникнути проблем з узгодженістю елементів архітектури, обробку даних часто намагаються виконувати або на стороні клієнта («товстий» клієнт), або на стороні сервера («тонкий» клієнт). Обидва підходи мають свої недоліки.

Недоліки моделі «товстий» клієнт:

– складність в адмініструванні;

– ускладнене оновлення програмного забезпечення, яке потрібно оновлювати одночасно в усій системі;

– труднощі в розподілі прав доступу, оскільки доступ регулюється таблицями, а не діями;

– перевантаження мережі через передавання необроблених даних;

– низький рівень захисту даних.

Недоліки моделі «товстий» сервер:

– складність реалізації, оскільки мови на кшталт PL/SQL погано підходять для розробки такого ПЗ;

– низька продуктивність програм, написаних на мовах типу PL/SQL;

– програми, розроблені для СУБД, не є переносними на інші платформи;

– помилки в таких програмах можуть спричинити збій у роботі всієї бази даних.

Для вирішення цих проблем використовується тришарова архітектура «клієнт-сервер» (рис. 2.23).

Рисунок 2.23 – Діаграма розгортання

У тришаровій архітектурі «тонкий» клієнт виконує лише функції представлення інформації, яка надходить із сервера застосунків. Такий підхід дозволяє точніше розподіляти повноваження операторів, що підвищує захищеність системи як від зовнішніх загроз, так і від помилкових дій персоналу.

Висновки до розділу 2

У другому розділі розглянуто процес моделювання інформаційної системи із застосуванням UML-нотацій. Зокрема, виконано аналіз вимог до системи через діаграми варіантів використання, які відображають функції, що виконує система, і взаємодії між суб'єктами. Досліджено основи проєктування діаграм прецедентів, що дозволяють встановлювати зв'язки між сценаріями використання і користувачами системи.

Також у розділі представлено виявлення класів-сутностей, які забезпечують статичне представлення системи та зв'язки між її об'єктами. Побудовано діаграми

класів, які демонструють зв'язок між клієнтами, операторами та іншими елементами системи, включаючи їх атрибути та методи.

Окремо проаналізовано моделювання видів діяльності через діаграми активностей, які описують основні сценарії функціонування системи, зокрема процеси заповнення анкет, укладання договорів і заміни номерів. Ці діаграми демонструють послідовність дій, а також можливість паралельного виконання процесів.

Розглянуто моделювання взаємодій між об'єктами через діаграми послідовностей і кооперації, що дозволяють відстежувати динаміку передачі повідомлень між елементами системи. Крім того, досліджено поведінку окремих класів за допомогою діаграм станів, які описують життєвий цикл ключових елементів, таких як договори та заяви.

Виконано проєктування статичної структури системи та фізичного представлення її компонентів. Побудовано діаграми розгортання для архітектури «клієнт-сервер» з аналізом двошарової і тришарової реалізацій. Робота сприяє формалізації моделювання складних інформаційних систем, забезпечуючи їх ефективне проєктування та впровадження.

РОЗДІЛ 3
РОЗРОБЛЕННЯ ІМІТАЦІЙНОЇ МОДЕЛІ ПРОГРАМНОГО ЗАСОБУ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Послідовність розробки базової імітаційної моделі інформаційної системи

На рис. 3.1 представлена однофрагментна модель ІС з використанням марковських процесів, що враховує перезавантаження програмного забезпечення після відмови.

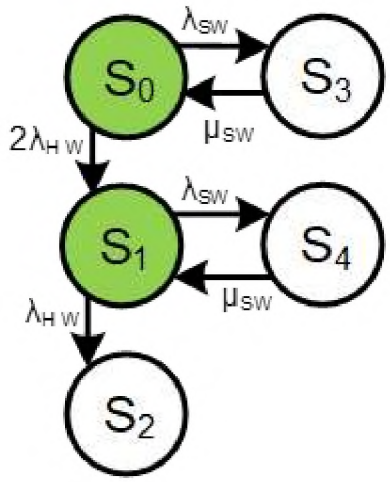


Рисунок 3.1 – Розмічений граф станів і переходів ІС

Для створення моделі оцінки готовності ІС без урахування оновлень програмного забезпечення були прийняті такі припущення [28]:

1) Потік подій, що переводить систему між функціональними станами, має стаціонарні, ординарні та безперервні властивості. Відповідно вхідні параметри моделі λ_{HW}, λ_{SW}, μ_{SW} приймаються постійними.

2) Кожен елемент ІС може перебувати у працездатному або непрацездатному стані у будь-який момент часу.

3) Припускається ідеальне функціонування засобів діагностування та контролю, що може призвести до трохи завищених значень показників готовності.

Проте це спрощення дозволяє зменшити обсяг обчислень, зважаючи на скорочення кількості розглядуваних функціональних станів.

Функціонування ІС узагальнено через деталізацію у вигляді графа станів, який відповідає моделі (рис. 3.1). У програмному інструментарії Matlab цей граф подається як об'єднання масиву вершин V і масиву дуг E, що відображають стани та переходи відповідно.

$$G = (V, E). \tag{3.1}$$

Для створення масивів V та E були розроблені власні програмні функції в Matlab у вигляді окремих m-файлів. Використання цих функцій для побудови базових імітаційних моделей зберегло логіку функціонування системи. Опис процесу створення ітерацій одного розіграшу базової імітаційної моделі, взятий за приклад з графа моделі, подано на рис. 3.1. Такий підхід відрізняється від того, що було запропоновано у [10], бо має недолік у зайвій кількості розіграшів на кожній ітерації, але при цьому не потребує додаткової пам'яті для зберігання невикористаних розіграшів.

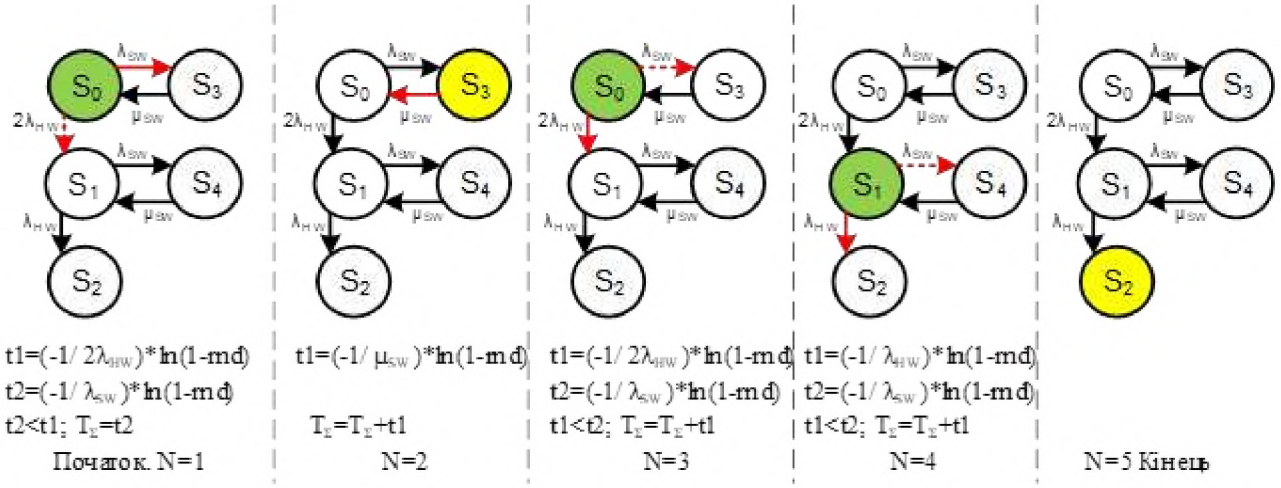


Рисунок 3.2 – Роз'яснення послідовності кроків базової імітаційної моделі ІС

У базовій імітаційній моделі часовий інтервал між послідовними подіями відтворюється випадковим чином з використанням експоненціального розподілу.

$$t_i = -\frac{1}{\pi_{j,k}} \ln(rnd), \tag{3.2}$$

де rnd – випадкове число, розподілене рівномірно на інтервалі [0..1];

$\pi_{j,k}$ – інтенсивність переходу від стану S_j до S_k.

У дослідженнях [10, 12] були запропоновані прямі формули для визначення часу за розподілом експоненти. У цій роботі для створення базових імітаційних моделей використовувалась розширена версія пакету Matlab Statistics Toolbox [15], включаючи функцію R = exprnd(mu). Формула (3.2) пройшла модифікацію:

$$t_i = \text{exprnd}(1/E[j,k]). \tag{3.3}$$

Під час кожної ітерації виконується розіграш k моментів часу ti, де k – це кількість зв'язків, що виходять з активної вершини графа (на рисунку 3.1 кольором позначені поточні вершини кожної ітерації). Якщо існує кілька зв'язків, дуга для переходу з поточного стану в новий обирається на основі мінімального часу розіграшу. У випадку наявності «поглинаючої» вершини у графі моделі (наприклад, вершина S2 на рис. 3.1), якщо система потрапляє в цю вершину під час ітераційного розіграшу, процес імітаційного моделювання припиняється, а розіграш вважається завершеним.

Іншим критерієм завершення повного розіграшу базової моделі є досягнення значення часу функціонування ІС T_Σ, яке відповідає або перевищує визначений інтервал дослідження. Узагальнений алгоритм побудови базової імітаційної моделі, який забезпечує виконання зазначеної кількості розіграшів показника готовності, включає наступні кроки.

Після запуску моделі вводяться значення інтенсивностей переходів та інші вхідні параметри. Одним із важливих параметрів є Tint – інтервал часу, під час якого імітується поведінка показника гарантованої працездатності ІС. Повний розіграш базової імітаційної моделі «відтворює» показник гарантованої працездатності протягом інтервалу [0..Tint]. За необхідності також визначаються

часткові показники гарантованої працездатності, такі як мінімальне значення функції готовності Amin, момент часу tconst, коли система потрапляє в поглинаючий стан тощо.

Процедура розіграшу виглядає так: на початковому етапі (час t=0) система перебуває в робочому стані S0, і поточний стан фіксується як n=1. Всі можливі переходи зі стану S0 в інші стани визначаються через вираз Q=E(find(E(:,1)==n),:), що формує вибірку з масиву E у вигляді масиву Q. Після проведення розіграшу однієї ітерації за допомогою функції ti=exprnd(1/Q(i)), де Q(i) – відповідна вага дуги графа, утворюється вектор t. Переможець ітераційного розіграшу визначає наступний поточний стан за допомогою виразу k=Q(q+find(t1==min(t1))). Якщо поточний стан є працездатним, обчислюється наробіток Tbf між відмовами, або ж наробіток Tbr відновлень і процедур багатоцільового обслуговування.

Наступним кроком перевіряється умова завершення повного розіграшу (якщо загальний час розіграшу не перевищив Tint або система не потрапила у поглинаючий стан). Якщо умова завершення не виконується, поточний стан змінюється на k, і розпочинається нова ітерація. Отже, повний розіграш показника гарантованої працездатності складається з ітераційних розіграшів, що забезпечують побудову його функції протягом інтервалу [0..Tint]. Якщо система потрапила в поглинаючий стан в момент часу tconst під час розіграшу, то в інтервалі [tconst..Tint] показник гарантованої працездатності приймає значення 1 (якщо поглинаючий стан є працездатним) або 0 (якщо поглинаючий стан викликає відмову).

В результаті одного запуску базової імітаційної моделі формуються два вектори: моментів часу tg фіксації розіграних подій та значень функції готовності A у відповідні моменти часу. Важливо враховувати, що з кожним розіграшем моменти часу набувають різних значень, тому перед застосуванням процедур усереднення необхідно визначити метод їх обробки (інтерполяція або формування узагальненої вибірки).

3.2 Імітаційна модель інформаційної системи для потоків подій, що зводяться до експоненціальних

На відміну від імітаційних моделей інформаційних систем, розглянутих у [37], де графи мали поглинаючі стани та працювали в умовах зміни вимог, параметрів середовища або неспецифікованих відмов, в даному випадку необхідно мати можливість вивчати зміну показника готовності з часом і визначати часткові показники гарантоздатності. Розіграш функції, а не постійного значення (і той факт, що в моделях з поглинаючими станами функція готовності має постійне значення, яке визначається однозначно), значно ускладнює можливість збільшення кількості розіграшів шляхом розширення часового інтервалу дослідження системи [11, 18]. Оскільки результати розіграшу показників готовності, отримані за допомогою імітаційної моделі, можуть мати ненормальний розподіл, кількість таких розіграшів визначається експертно та є вхідним параметром для моделі.

Блок-схема алгоритму імітаційного моделювання гарантоздатних інформаційних систем представлена на рисунку 3.3. Процес моделювання включає кілька етапів. Спочатку, після запуску скрипта, вводяться значення вхідних параметрів і значення Nr – кількості повних розіграшів імітаційної моделі. Один повний розіграш здійснюється за допомогою функції-скрипту базової імітаційної моделі, описаної в пункті 3.1. Після виконання кожного розіграшу формуються вектори часових моментів і значень статистичних оцінок показника гарантованої працездатності. Отримані оцінки можуть бути оброблені двома методами.

У першому методі часовий інтервал дослідження системи, time_interval, розбивається на NT рівних проміжків. На кінцях цих проміжків проводиться інтерполяція розіграшу функції готовності за допомогою виразу:

$$avgA1 = [avgA1; interp1(tg, Pg, Ti)].$$

Після виконання заданої кількості повних розіграшів (Nr) створюється матриця avgA1, яка містить Nr×NT значень інтерпольованих статистичних оцінок функції гарантійної працездатності. Усереднення цих оцінок за допомогою функції

nanmean(avgA1,1) дозволяє отримати результативні значення зміни показника гарантованої працездатності протягом експлуатації системи.

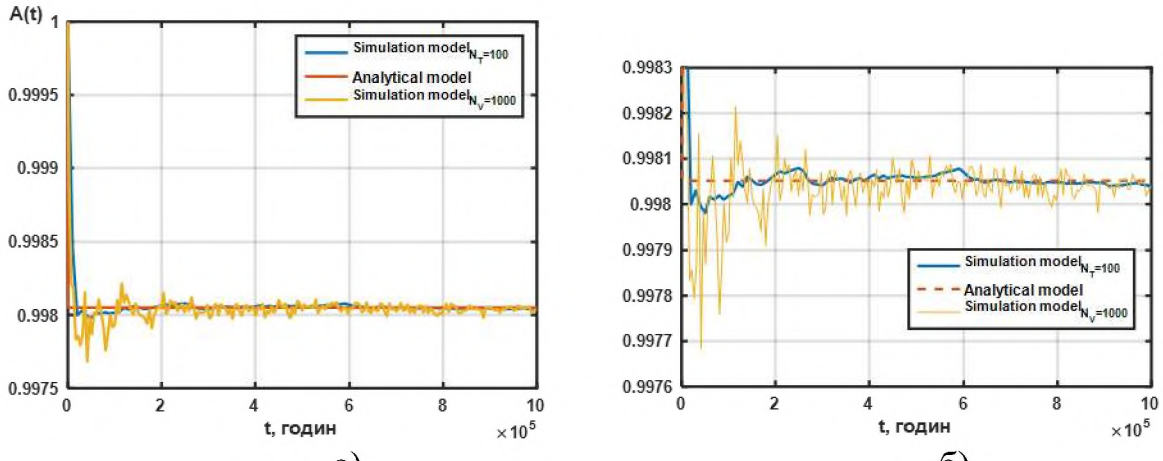
Рисунок 3.3 – Алгоритм імітаційного моделювання гарантоздатних ІС

Другий метод обробки отриманих значень розіграшів функціонального показника гарантійної працездатності (наприклад, функції готовності) описаний в [21]. Цей метод передбачає об’єднання всіх розіграшів функції гарантійної працездатності в один масив, де перший стовпчик містить часові відмітки, а другий

– розіграні значення функції гарантійної працездатності. Далі виконуються процедури обробки цього масиву, включаючи відсікання деяких детермінованих значень у точках t=0 і tconst<t<Taim_interval (A(t)=1). Для цього послідовно виконуються операції унікалізації елементів матриці Ai за допомогою unique(Ai,'rows') і відсікання перших та останніх рядків масиву: Ai(1,:)=[]; Ai(end,:)=[].

Після цього елементи матриці Ai розподіляються на окремі групи-вибірки (з розміром nv). У межах кожної групи виконується усереднення значень часових відміток і показника гарантійної працездатності за допомогою функції mean(X).

На рисунку 3.4 представлені результати порівняння запропонованих методів імітаційного моделювання функції готовності та результатів аналітичної моделі Мг2. Ці моделі не включають поглинаючих станів, проте ілюструють відхилення результатів імітаційного моделювання та їх варіації в залежності від розмірів вибірки даних.



а) б)

Рисунок 3.4 – Результати порівняння імітаційної моделі ІС та аналітичної моделі у вертикальному масштабі а) [0.9975…1]; б) [0.9976…0.9983]

Аналіз порівняння результатів імітаційного моделювання з використанням першого та другого методів побудови вибірок розіграшів на рис. 3.4,а та рис. 3.4,б показує більшу похибку у другому методі (побудові вибірок за допомогою нерівномірних часових відліків). Однак графіки на рисунку 3.4,в вказують на

неточність такого висновку. Справа в тому, що для зменшення похибки потрібно правильно підібрати розмір вибірки, і при збільшенні цього розміру на порядок, графік імітаційної моделі стає значно більш плавним.

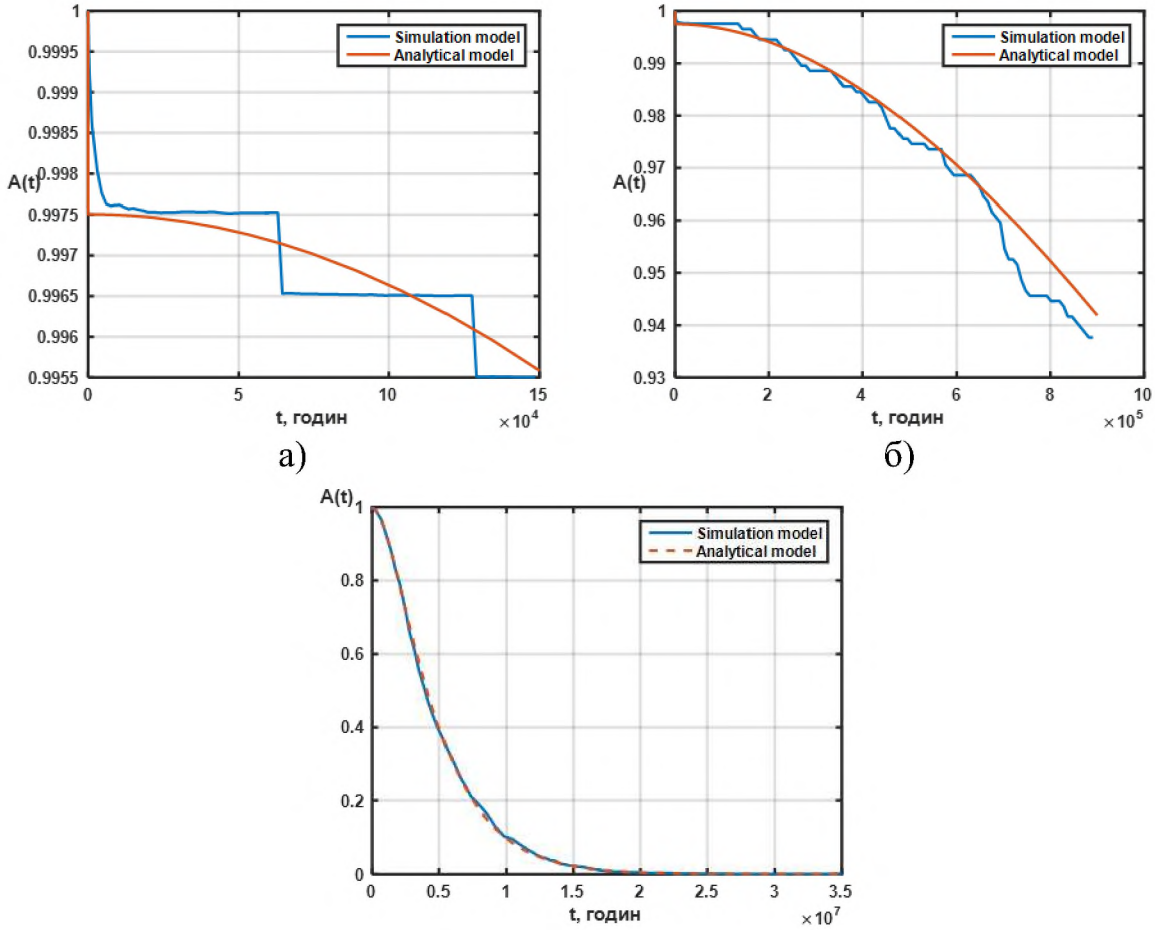


Рисунок 3.5 – Результати порівняння імітаційної моделі ІС та аналітичної моделі в горизонтальному масштабі а) [0…15*10^4]; б) [0…10^5]; в) [0…3,5*10^6]

На рисунку 3.5 представлені результати порівняння аналітичної та імітаційної моделей для систем із поглинаючими станами. Аналітична модель цієї системи – марковська багатофрагментна. Обчислення проведено на основі наступних вхідних даних. Деякі вхідні параметри моделі залишалися незмінними, тоді як інші змінювалися в межах заданих інтервалів. Незмінні параметри включають:

– інтенсивність відмови одного апаратного каналу $\lambda_{HW} = 3*10^{-7}$ (1/год);

– початкова інтенсивність відмов ПЗ $\lambda_{SW\,0} = 5*10^{-4}$ (1/год);

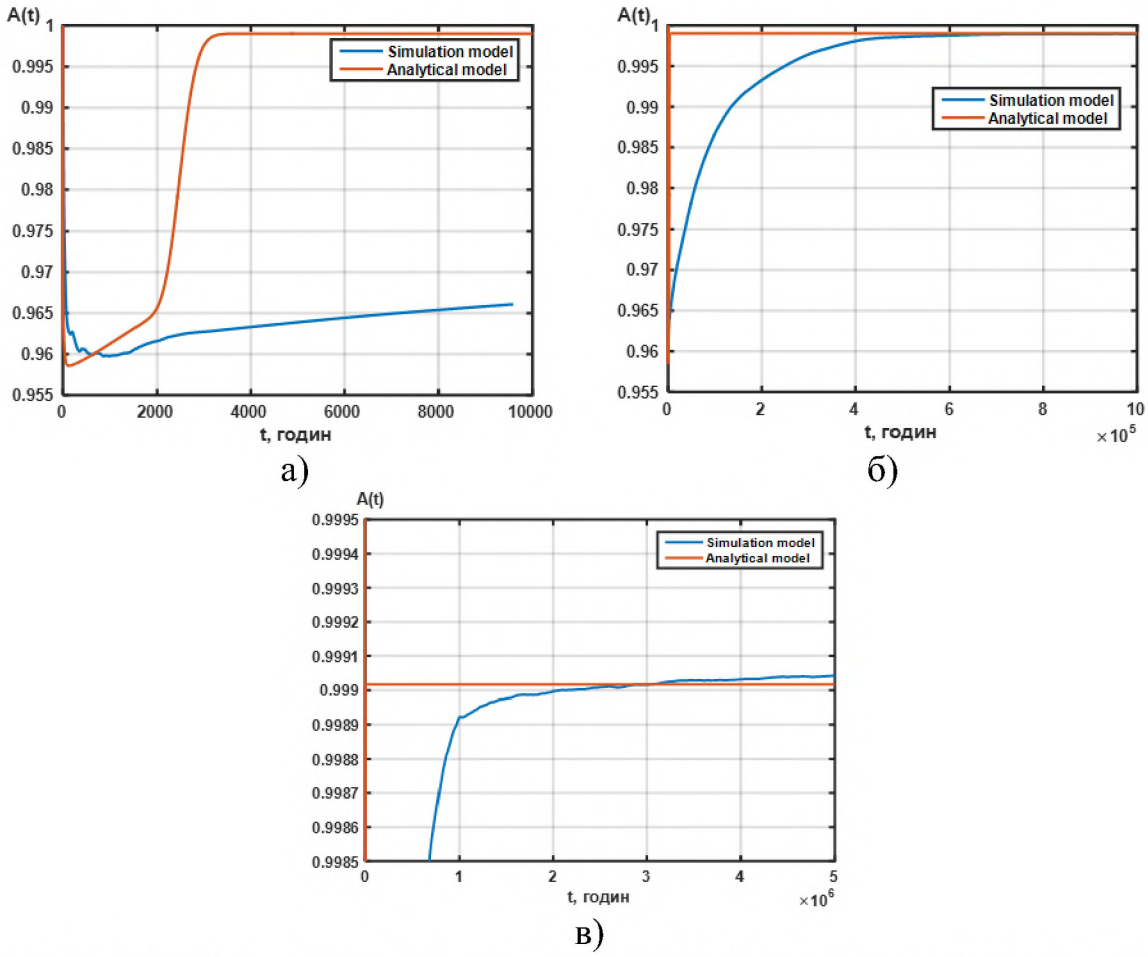
– інтенсивність відновлення системи після виявлення програмного дефекту (шляхом перезапуску ПЗ) $\mu_{SW} = 0{,}2$ (1/год);

– інтенсивність відмови ПЗ після усунення всіх дефектів дорівнює нулю $\lambda_{SW\,k} = 0$ (1/год).

Для вивчення готовності системи використовувалися змінні значення вхідних параметрів, які були узагальнені в табл. 3.1.

Таблиця 3.1 – Змінні значення вхідних параметрів моделі ІС

Параметр моделі	Значення параметра			
$\Delta\lambda_{SW}$ (1/год)	$3\cdot10^{-4}$	$2\cdot10^{-4}$	$1\cdot10^{-4}$	$9\cdot10^{-5}$
λ_{UP} (1/год)	$4{,}63*10^{-4}$	$2{,}31*10^{-4}$		$1{,}14*10^{-4}$
μ_{UP} (1/год)	0,33	0,5		1



а) б)

в)

Рисунок 3.6 – Результати порівняння імітаційної моделі гарантоздатних ІС та моделі Мг3 в горизонтальному масштабі а) [0…10^5]; б) [0…10^6]; у вертикальному масштабі в) [0.9985…0.9995]

Рисунок 3.6 демонструє порівняння аналітичної та імітаційної моделей для ІС, які працюють у середовищах змінних параметрів та проводять онлайн-верифікацію. У імітаційній моделі використано перший метод формування вибірок за допомогою рівномірних часових інтервалів, де інтервал дослідження системи був поділений на 1000 підінтервалів. Функціональний показник гарантованої працездатності (функція готовності) у імітаційній моделі був розіграний Nr=1000 разів.

Результати порівняння графіків аналітичної та імітаційної моделей свідчать про їхню значну відповідність на початку та в кінці дослідження, особливо щодо часткових показників гарантоздатності Amin та Aconst. Проте, визначення часових часткових показників гарантоздатності, таких як tconst, T1, T2, виявилося непридатним для імітаційної моделі через її інерційність.

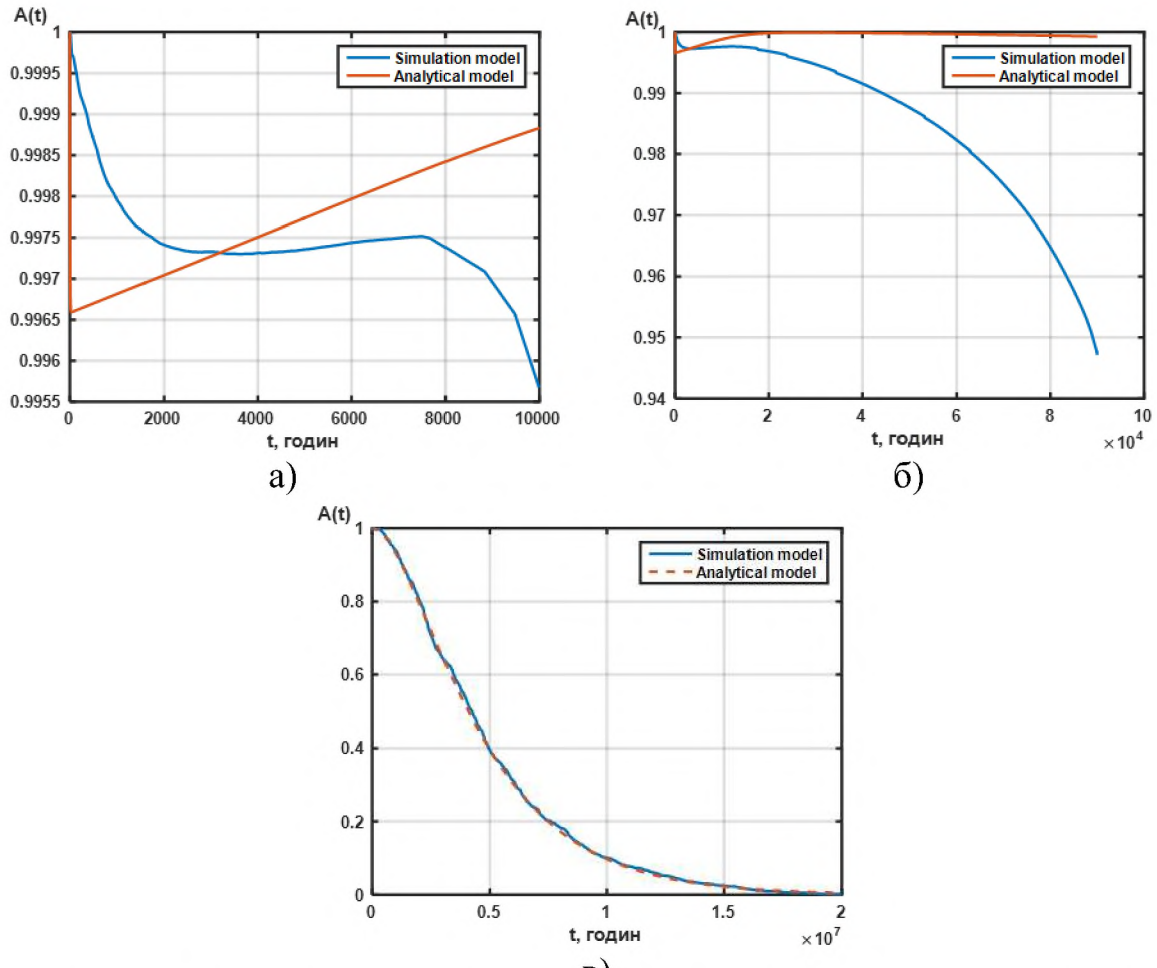


Рисунок 3.7 – Результати порівняння імітаційної моделі ІС та аналітичної моделі в горизонтальному масштабі а) [0…10^4]; б) [0…10^5]; в) [0…2*10^7]

Функціональний показник гарантованої працездатності (функція готовності) у імітаційній моделі був розіграний Nr=1000 разів. У моделі імітації також використано перший метод формування вибірок за допомогою рівномірних часових інтервалів. З графіків видно, що імітаційні моделі для оцінки функціональної гарантії мають особливість – «інерційність». За збільшення часового проміжку спостереження системи різниця між графіками аналітичної та імітаційної моделей стає менш помітною, оскільки масштаб вертикальної осі графіків зростає. Максимальне відхилення результатів імітаційної моделі від аналітичної склало ΔA= 0,0039.

На рисунку 3.7 представлені результати порівняння багатофрагментної аналітичної моделі з поглинаючими станами і результатів імітаційного моделювання ІС. На значному часовому проміжку (рис. 3.7,в) моделі показують помітну відповідність у визначенні функції готовності, з відхиленням ΔA=0.0225. Але на рис. 3.7,б видно, що інерційність імітаційної моделі проявляється на інтервалі дослідження в 10^5 годин. На рисунку 3.7,а також помітний відповідний збіг між аналітичною та імітаційною моделями у визначенні показника Amin. Проте визначення його за допомогою імітаційної моделі викликає деякі труднощі, оскільки крива функції готовності не завжди має перший точний перегин.

Отже, для ІС, які опиняються під впливом простих або експоненціальних потоків подій та не мають поглинаючих станів, були сформульовані такі рекомендації щодо використання імітаційних моделей:

1. Для моделей з простими потоками без поглинаючих станів застосовують ІМ для визначення часткового показника гарантоздатності Aconst.

2. У випадку моделей з простими потоками з поглинаючими станами за допомогою ІМ вивчається поведінка функції готовності на великих проміжках часу, що дає можливість визначити частковий показник tconst.

3. Для моделей з потоками, які мають експоненціальний характер та не мають поглинаючих станів, ІМ може допомогти оцінити часткові показники гарантоздатності Amin і Aconst.

4. У випадку моделей з експоненціальними потоками та поглинаючими станами за допомогою ІМ можна дослідити поведінку функції готовності на значних проміжках часу, використовуючи це для визначення часткового показника tconst. Іноді можна візуально оцінити частковий показник гарантоздатності Amin.

3.3 Розрахунок економічних затрат на розробку імітаційної моделі інформаційної системи

Розрахунок затрат на розробку моделей інформаційних систем базується на методології, описаній у [39]. Трудомісткість розробки імітаційних моделей, яка є складовою процесу створення програмного продукту, враховує витрати часу на такі етапи:

– формування технічного завдання;

– дослідження та розробка алгоритму вирішення задачі;

– створення блок-схеми алгоритму;

– програмування за блок-схемою;

– тестування та налагодження моделі у середовищі MATLAB;

– оформлення документації;

– виявлення помилок та їх усунення.

Загальну трудомісткість $T_{заг}$ у людино-днях можна розрахувати за допомогою формули:

$$T_{заг} = N_{час} \cdot k_{скл} \cdot k_{м} \cdot k_{станд} \cdot k_{станд.ПП}, \quad (3.4)$$

де $T_{заг}$ – загальна трудомісткість, людино-дні;

$N_{час}$ – базова норма часу (зазвичай 30 людино-днів);

$k_{скл}$ – коефіцієнт складності завдання (наприклад, 1,08);

$k_{м}$ – коефіцієнт, що враховує рівень мови програмування (для MATLAB приймається 1);

$k_{станд}$ – коефіцієнт застосування стандартних програмних компонентів (0,7);

$k_{станд.ПП}$ – коефіцієнт розробки стандартного коду (1,2).

Наприклад, якщо $N_{\text{час}} = 30$, то:

$$T_{\text{заг}} = 30 \cdot 1{,}08 \cdot 1 \cdot 0{,}7 \cdot 1{,}2 \approx 27{,}2 \approx 30 \text{ людино-днів.}$$

Кількість розробників $Ч_i$ визначається за формулою:

$$Ч_i = \frac{T_{\text{заг}}}{F_{\text{рч}} \cdot t_{\text{розроб}}/12}, \quad (3.5)$$

де $F_{\text{рч}}$ – річний фонд робочого часу (у 2023 році дорівнює 249 робочих днів);

$t_{\text{розроб}}$ – запланований термін розробки (у місяцях).

Для $T_{\text{заг}} = 30$, $F_{\text{рч}} = 249$, $t_{\text{розроб}} = 1$:

$$Ч_i = \frac{30}{249 \cdot \frac{1}{12}} \approx 1 \text{ особа.}$$

Фактична вартість розробки моделей розраховується через калькуляцію витрат, яка включає:

1. Виробничу собівартість (*Свир*):

$$Свир = S_{ЗП} + S_{ЄСВ} + Sзв + Sм, \quad (3.6)$$

де $S_{ЗП}$ – заробітна плата розробника (9000 грн);

$S_{ЄСВ}$ – єдиний соціальний внесок (1430 грн);

Sзв – накладні витрати (7455,21 грн);

Sм – витрати на матеріали (1491,04 грн).

Приклад розрахунку:

$$Свир = 9000 + 1430 + 7455{,}21 + 1491{,}04 = 19376{,}25 \text{ грн.}$$

2. Адміністративні витрати ($S_{адмін}$):

$$S_{адмін} = k_{адмін} \cdot S_{ЗП}, \quad (3.7)$$

де $k_{адмін} = 1{,}1$. Отже, $S_{адмін} = 1{,}1 \cdot 9000 = 9900$ грн.

3. Витрати на збут ($S_{збут}$):

$$S_{збут} = k_{збут} \cdot S_{вир}, \quad (3.8)$$

де $k_{збут} = 0{,}025$. Отже, $S_{збут} = 0{,}025 \cdot 19376{,}25 = 484{,}4$ грн.

Повна собівартість ($S_{пов}$):

$$S_{пов} = S_{вир} + S_{адмін} + S_{збут} = 19376{,}25 + 9900 + 484{,}4 = 29760{,}65 \text{ грн.}$$

У результаті аналізу встановлено, що найбільшу частку витрат займають адміністративні витрати, тоді як витрати на збут є найменшими. Інформація про структуру витрат дозволяє ефективно управляти ресурсами, знижувати собівартість розробки моделей і підвищувати їхню конкурентоспроможність.

Висновки до розділу 3

У розділі розглянуто процес розробки імітаційної моделі програмного забезпечення інформаційної системи із застосуванням методів марковських процесів і аналізу потоків подій. Виконано побудову базової моделі через граф станів і переходів, що враховує перезавантаження після відмов. Досліджено підходи до моделювання інтервалів між подіями з використанням експоненціального розподілу та алгоритмів, реалізованих у Matlab.

Детально описано послідовність створення та ітераційного моделювання показників гарантоздатності, таких як функція готовності, із застосуванням процедур усереднення результатів. Проаналізовано використання альтернативних методів формування вибірок для оцінки показників готовності.

У роботі також розглянуто можливості застосування імітаційних моделей для систем із поглинаючими станами та змінними параметрами середовища. Виконано порівняльний аналіз результатів імітаційного моделювання з аналітичними моделями для оцінки похибок і відповідності результатів. Запропоновано рекомендації для адаптації моделей до різних потоків подій та умов функціонування.

Крім того, виконано розрахунок економічних затрат на розробку імітаційних моделей, включаючи трудомісткість, адміністративні витрати та собівартість. Систематизовано підходи до побудови, аналізу й економічного оцінювання імітаційних моделей, що забезпечує комплексний підхід до розробки програмного забезпечення для інформаційних систем.

ВИСНОВКИ

У роботі була поставлена і вирішена актуальна задача створення імітаційної моделі інформаційної системи «Центр обслуговування абонентів» для забезпечення її ефективного функціонування.

1. Виконано аналіз сучасного стану моделей обслуговування абонентів у системах стільникового зв'язку. Розглянуто актуальні методи моделювання інформаційних систем, зокрема структурні методи й методологію DFD, що забезпечують деталізацію процесів обслуговування.

2. Розглянуто методи моделювання інформаційної системи з використанням UML-нотацій. Побудовано діаграми варіантів використання, класів, активностей, а також діаграми взаємодій і станів, що дозволяють формалізувати основні процеси функціонування системи.

3. Розроблено імітаційну модель інформаційної системи на основі марковських процесів із використанням програмного інструментарію Matlab. Представлено алгоритми для побудови базових імітаційних моделей, досліджено функцію готовності та інші показники системи.

4. Проведено порівняльний аналіз результатів імітаційного та аналітичного моделювання для оцінки відповідності отриманих результатів. Запропоновано методи інтерполяції та усереднення даних для покращення точності моделювання.

5. Розглянуто економічні аспекти створення імітаційної моделі, включаючи трудомісткість і собівартість розробки, що дозволяє оптимізувати витрати.

6. Проведений аналіз функціональних, структурних і економічних аспектів розробки інформаційної системи дозволив сформувати висновки щодо ефективності використання імітаційних моделей для автоматизації обслуговування абонентів. Моделі демонструють високу точність у прогнозуванні готовності системи та можливості її адаптації до змінних умов.

На основі отриманих висновків запропоновано рекомендації щодо впровадження трирівневої архітектури «клієнт-сервер» для підвищення безпеки та гнучкості системи, а також використання імітаційних моделей для оптимізації бізнес-процесів.

Таким чином, поставлені задачі розв'язано у повному обсязі. Напрямком подальших досліджень є інтеграція розроблених моделей із сучасними технологіями проєктування ІС для покращення адаптивності системи до нових викликів.