ЛАБОРАТОРНА РОБОТА 4
РОЗРОБКА UML–ДІАГРАМ ЛОГІЧНОГО РІВНЯ ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ: МОДЕЛЮВАННЯ ДИНАМІЧНИХ АСПЕКТІВ

Мета роботи – ознайомитися із методикою побудови UML-діаграм логічного рівня проектування ПЗ, які застосовуються для моделювання динамічних аспектів функціонування ПЗ.

4.1. Загальні відомості

Після побудови діаграми класів (class diagram) та діаграми об’єктів (object diagram) (див. л/р 3), які визначають статичні аспекти побудови відповідних КПР, для моделювання їх динамічних аспектів (тобто для відображення взаємодії компонентів) використовуються такі діаграми UML [3–4]:

- діаграми кінцевого автомату (state machine diagram);
- діаграми діяльності (activity diagram);
- діаграми комунікації (communication diagram);
- діаграми композитної структури (composite structure diagram);
- діаграми обзору взаємодії (interaction overview diagram).

Діаграма кінцевого автомата (state machine diagram)

Діаграма кінцевого автомата описує процес зміни станів одного екземпляра певного класу деякої програмної системи (ПС), тобто моделює всі можливі зміни в стані конкретного програмного об’єкта. При цьому зміна стану об’єкта може бути викликана зовнішніми діями з боку інших об’єктів. Головне призначення цієї діаграми – описати можливі послідовності станів і переходів, які у сукупності характеризують поведінку певного елемента UML-моделі ПС протягом його життєвого циклу.

Для того щоб у середовищі Visual Paradigm перейти до режиму побудови діаграми кінцевого автомата, потрібно на вкладці UML вибрати пункт State Machine Diagram (рис .4.1).

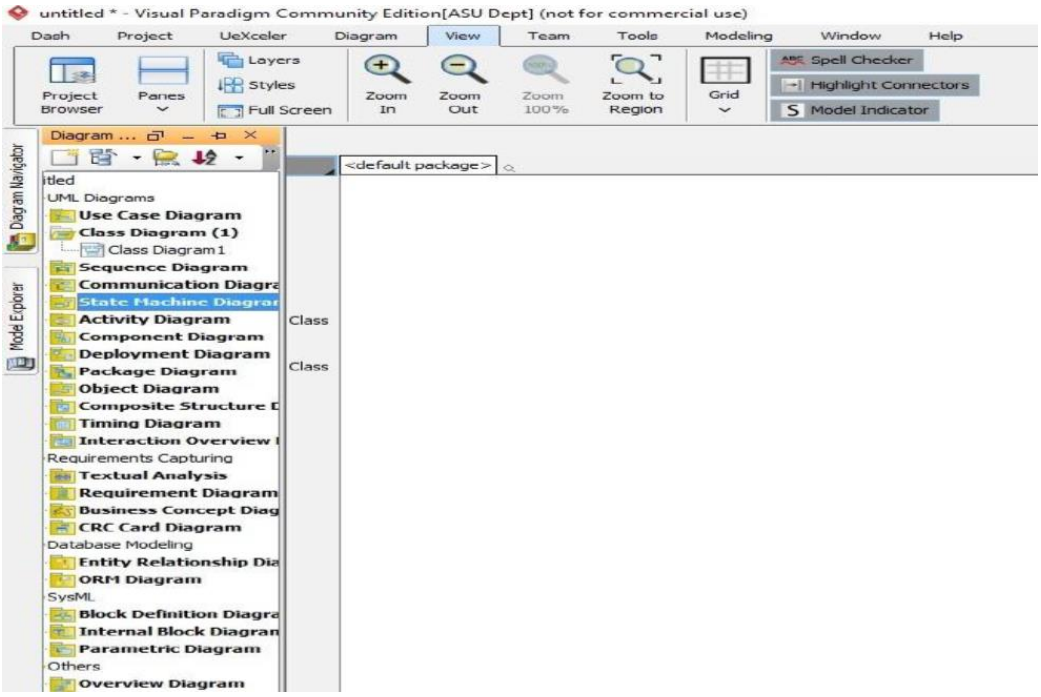

Рисунок 4.1 – Інтерфейс у середовищі Visual Paradigm при побудові діаграми кінцевого автомата

Стан (state) об’єкта подається у вигляді набору конкретних значень його атрибутів, при цьому зміна їх окремих значень відображатиме зміну його стану.

Стан на діаграмі зображується прямокутником з вершинами, що є округленими. Цей прямокутник, у свою чергу, може бути розділений горизонтальною лінією на дві секції. Якщо вказана лише одна секція, то у ній записується лише назва стану, інакше в першій з них записується назва стану, а в другій – список деяких внутрішніх дій. Назва стану є рядком тексту, який розкриває змістовний сенс даного стану. Назва завжди записується з великої літери.

У списку внутрішніх дій міститься перелік дій, які виконуються в процесі знаходження об’єкта в цьому стані. Кожна з дій записується у вигляді окремого рядка і має такий формат: «позначка дії/опис дії».

Позначка дії вказує на обставини або умови, при яких виконуватиметься відповідна дія. Перелік позначок дії має фіксовані значення, а саме:

• entry – ця позначка вказує на дію, яка виконується у момент входу в цей стан (вхідна дія);

• exit – ця позначка вказує на дію, яка виконується у момент виходу з цього стану (вихідна дія);

• do – ця позначка специфікує діяльність, що виконується, яка виконується протягом усього часу, поки об’єкт знаходиться у цьому стані, або доти, поки не закінчиться обчислення, специфіковане наступним за нею виразом дії;

• event – ця позначка визначає дію, яка виконується у момент, коли в ПС наступає певна подія (наприклад, користувач натискає клавішу Escape, тощо).

Початковий стан є окремим випадком стану об’єкта ПС, який не містить жодних внутрішніх дій. Графічно початковий стан у мові UML позначається у вигляді зафарбованого кола, з якого може лише виходити стрілка, відповідна першому переходу ПС у наступний стан.

Кінцевий стан є також окремим випадком стану об’єкта ПС, який також не містить жодних внутрішніх дій. Графічно цей стан позначається у вигляді зафарбовано-го кола, поміщеного в коло, в яке може лише входити стрілка, відповідна останньому переходу ПС.

Зміст поняття переходу та його позначення на діаграмі

Простий перехід (simple transition) є відношенням між двома послідовними станами, яке вказує на факт зміни одного стану іншим. Перебування об’єкта у першому стані може супроводжуватися виконанням деяких дій, а перехід в другий стан буде можливий після завершення цих дій, а також після задоволення деяких додаткових умов. У цьому випадку говорять, що відповідний перехід спрацьовує.

Перехід спрацьовує при настанні деякої події у ПС: це може бути, наприклад, закінчення виконання певної дії (do activity), отримання об’єктом повідомлення та ін. На переході вказується назва події. Крім того, на переході можуть вказуватися дії, що виробляються об’єктом у відповідь на зовнішні події при переході з одного ста-ну в інший. Спрацьовування переходу може залежати не лише від настання деякої події, але і від виконання певної умови, так званої сторожової умови (guard condi-tion). Об’єкт перейде з одного стану в інший у тому випадку, якщо сталася вказана подія і сторожова умова набула значення «true/істина».

Перехід може бути направлений у той же стан, з якого він виходить. У цьому ви-падку його називають переходом в самого себе, а вихідний і цільовий стани переходу збігаються. При переході в себе об’єкт покидає вихідний стан, а потім знову входить в нього. При цьому кожного разу виконуються внутрішні дії, специфіковані по-значками entry та exit.

На рис. 4.2 показано початковий стан деякої ПС та перехід у стан System start, що відповідає появі на екрані комп'ютера діалогового вікна (або стартової HTML-сторінки). При вході в цей стан це вікно створюється ПС, при виході з цього стану воно закривається.

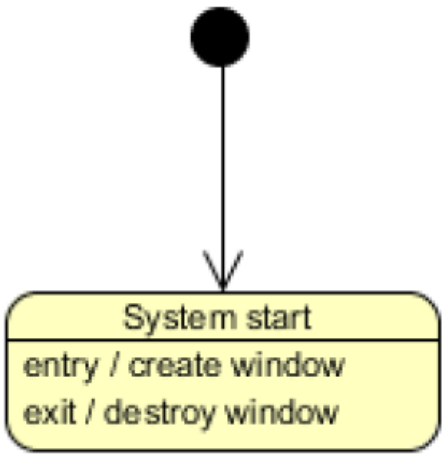

Рисунок 4.2 – Графічне зображення окремого стану з секцією опису внутрішніх дій

На рис. 4.3 наведена діаграма станів для прикладу розробки ПЗ процедури авторизації користувача у ПС (див. л/р 3–4). При вході в стан «System start» створюється відповідне діалогове вікно (при виході воно закривається).

З цього стану є два можливих переходи в інші стани. При виконанні дії «sign in» система переходить до стану «Login and password input» в якому буде виконано введення даних щодо імені користувача (LoginName) та його пароля (PW).

При вході у цей стан (позначка дії: entry/input field initialization) виконується ініціалізація полів вводу даних (позначка дії do/get Symbol) і виконується введення символів у поля вводу, а при отриманні зовнішньої дії натисканням користувачем клавіші «Escape» (позначка дії: event Escape button / Close window) вікно закривають.

Із стану «Login and password input» існує лише один перехід до стану «Authorization», цей перехід буде виконано після виконання дії «Input data submit».

При вході у стан «Authorization» буде встановлено з'єднання з базою даних ПС. У цьому стані виконується перевірка імені та пароля користувача (позначка дії «do / check login and password»). У разі успішного входу – це позначено сторожовою умо-вою [is LoginPasswordCorrect=true] – система переходить у кінцевий стан, у

проти-лежному випадку – це позначено сторожовою умовою [is LoginPasswordCorrect=false] – система має повернутися у стан «Login and password input» (рис. 4.3).

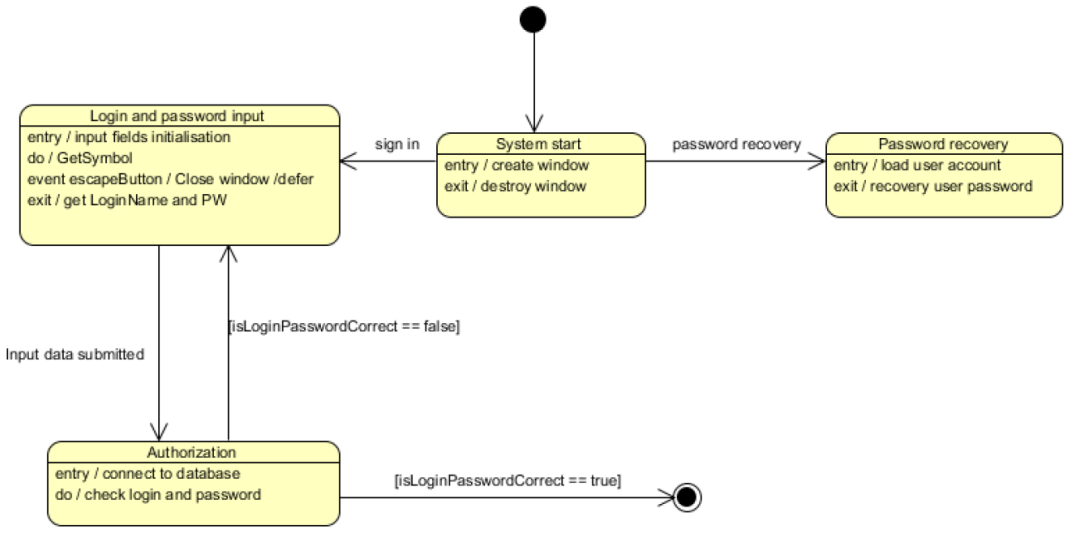

Рисунок 4.3 – Приклад побудови діаграми кінцевого автомата

Із стану «System start» при виконанні дії «password recovery» буде виконано пере-хід до стану «Password recovery». При вході в цей стан буде виконано пошук у БД облікового запису користувача (позначка дії: entry/load user account), протягом дії цього стану буде виконано відновлення пароля користувача (позначка дії: do / recov-ery user password).

Діаграми діяльності (activity diagram)

При моделюванні поведінки ПС виникає необхідність не лише представити процес зміни її станів, але і деталізувати особливості алгоритмічної реалізації виконуваних системою операцій. Для моделювання процесу виконання операцій у мові UML використовуються так звані діаграми діяльності. Їх графічна нотація багато в чому схожа на нотацію діаграми станів, оскільки на діаграмах діяльності також присутні позначення станів і переходів. Відмінність полягає в семантиці станів, які використовуються для визначення саме дій (тобто це так звані стани дії), у відсутності на переходах сигнатури опису подій. Кожне перебування на діаграмі діяльності відповідає виконанню деякої елементарної операції, а перехід у наступний стан спрацьовує лише при завершенні цієї операції в попередньому стані.

Перехід до режиму побудови діаграми діяльності в середовищі Visual Paradigm виконується так як і для діаграми кінцевого автомата (рис. 4.1).

Стан дії (action state) є спеціальним випадком стану з деякою вхідною дією і принаймні одним переходом, що виходить зі стану. Цей перехід неявно передбачає, що вхідна дія вже завершилась. Стан дії не може мати внутрішніх переходів, оскільки воно є елементарним. Звичайне використання стану дії полягає в моделюванні одного кроку виконання алгоритму (процедури) або потоку управління. Графічно стан дії зображується прямокутником, бокові сторони якого замінені округленими дугами (рис. 4.4.). Усередині цієї фігури позначається опис дії (action-expression), який має бути унікальним у межах однієї діаграми діяльності. Дія може бути записана розмовною мовою на деякому псевдокоді або мові програмування. Кожна діаграма діяльності повинна мати лише один початковий та один кінцевий стан (вони мають такі ж позначення, як і на діаграмі станів (рис. 4.2–4.3).

При побудові діаграми діяльності використовуються лише не тригерні переходи, а такі, які спрацьовують відразу після завершення відповідної дії. Цей перехід пере-водить систему в подальший стан дії відразу, як тільки закінчиться дія в попереднь-ому стані. На діаграмі такий перехід зображується суцільною лінією зі стрілкою. На рис. 4.4, а показано окрему дію «System start», яка виконується в ПС безпосередньо після початкового стану.

Якщо зі стану дії виходить єдиний перехід, то він може бути ніяк не помічений. Якщо таких переходів декілька, то спрацювати може лише один з них.

Саме у цьому випадку для кожного з таких переходів має бути явно записана сторожова умова в прямих дужках. При цьому для всіх переходів, що виходять з деякого стану, повинна виконуватися вимога істинності лише одного з них. Подібний випадок зустрічається тоді, коли послідовно виконувана діяльність повинна розділитися на альтернативні гілки залежно від значення деякого проміжного результату.

Така ситуація отримала назву альтернативного розгалуження або просто розгалуження (decision), а для її позначення застосовується спеціальний символ – невеликий ромб, усередині якого немає жодного тексту (рис. 4.4, б). У цей ромб може входити лише одна стрілка від того стану дії, після виконання якого потік управління має бути продовжений по одній з гілок, що взаємно виключають. Прийнято вхідну

стрілку приєднувати до верхньої або лівої вершини символу розгалуження. Стрілок, що виходять, може бути дві або більше, але для кожної з них явно вказується відповідна сторожова умова у формі логічного виразу.

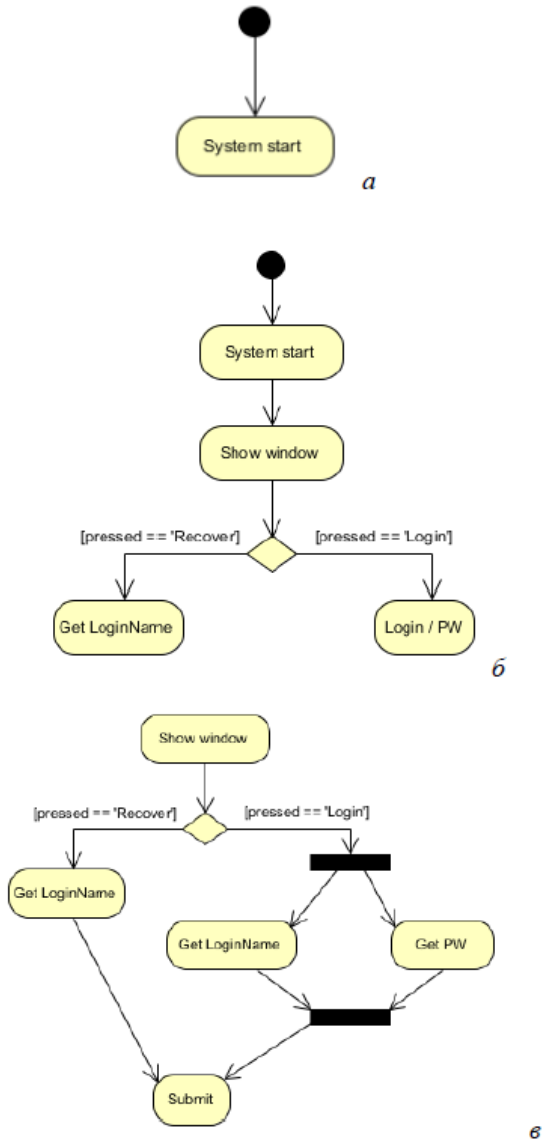

Рисунок 4.4 – Зображення основних графічних елементів діаграми дії: а – окрема дія; б – альтернативне розгалуження; в – паралельне розділення та злиття (або синхронізація)

Один із найбільш значних недоліків звичайних блок-схем або структурних схем алгоритмів пов’язаний з проблемою зображення паралельних гілок окремих обчислень. У мові UML для цієї мети використовується спеціальний символ для розділення і злиття паралельних обчислень або потоків управління в ПС. Це зображується відрізком горизонтальної лінії, товщина якої декілька ширше за основні суцільні лі-

нії діаграми діяльності. При цьому розділення (concurrent fork) має один вхідний перехід і декілька, що виходять, а злиття (join) або синхронізація, навпаки, має декі-лька вхідних переходів і один вихідний (рис.4.4, в).

У діаграмах діяльності можливо відобразити той факт, що певні сукупності дій у ПС можуть виконуватися деякими суб'єктами, наприклад, окремі бізнес-процеси є асоційованими з конкретними підрозділами компанії або дії щодо обробки даних розподіляються між клієнтським додатком та сервером.

Для моделювання цих особливостей у мові UML використовується спеціальна конструкція, що отримала назву доріжки (swimlanes). Мається на увазі візуальна аналогія з плавальними доріжками в басейні, якщо дивитися на відповідну діаграму. При цьому всі стани дії на діаграмі діяльності поділяються на окремі групи, які відділяються один від одного вертикальними лініями. Дві сусідні лінії і утворюють доріжку, а група станів між цими лініями виконується окремим суб'єктом: відділом компанії, групою користувачів, програмним додатком тощо. Приклад використання цього механізму відображено на рис. 4.5. На цій діаграмі присутні дві доріжки (swimlane). Доріжка Client відповідає виконанню клієнта, доріжка Server – виконанню сервера. Після старту системи (System start) виводиться діалогове вікно (Show window). Після цього можуть бути два варіанти виконання системи.

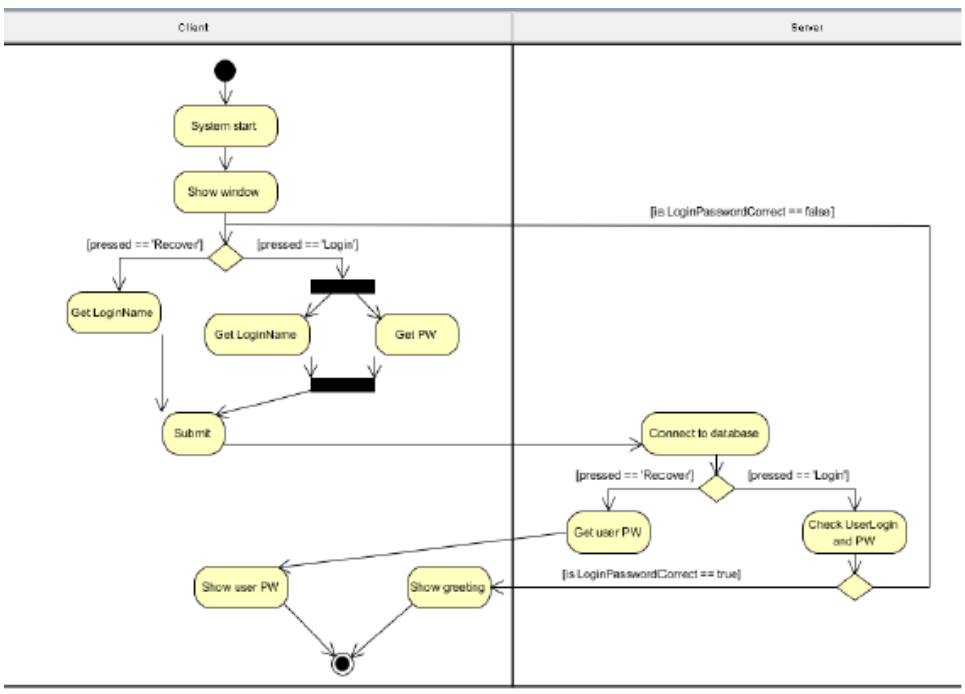

Рисунок 4.5 – Приклад побудови діаграми дії з використанням доріжок

1. Вхід у систему: виконується у випадку виконання умови pressed == «Login». Після цього будуть введені ім’я та пароль користувача Ger LoginName, Get PW – ці дії можуть виконуватися у будь-якому порядку, тобто паралельно, – а потім вони синхронізуються при виконанні дії Submit. Після цього на сервері виконується підключення до БД системи – дія Connect to database і виконується перевірка імені та пароля: Check Login and PW. У разі успішної перевірки («isLoginPasswordCorrect == true») буде показано привітання користувачеві («Show greeting») і система пе-рейде в кінцевий стан, у разі неуспішної перевірки («isLoginPasswordCorrect == false») виконується перехід до активності («Show window»).

2. Відновлення пароля: виконується у випадку виконання умови pressed == «Recover». Після цього система отримує ім’я користувача (LoginName), на сервері виконується підключення до бази даних та пошук пароля користувача. Після завершення пошуку пароля виконується активність «Show user PW» і система переходить до кінцевого стану (рис. 4.5).

Діаграми комунікації (communication diagram)

Для переходу до режиму побудови діаграми комунікації у середовищі Visual Paradigm потрібно на вкладці UML вибрати пункт Communication Diagram (рис. 4.6). Головна особливість діаграми комунікації полягає у можливості графічно представити не лише послідовність взаємодії, але і всі структурні стосунки між об’єктами ПС, що беруть участь у цій взаємодії. На цій діаграмі у вигляді прямокутників зображуються об’єкти, що беруть участь у взаємодії, кожен об’єкт містить назву, його клас і, можливо, значення атрибутів. Наприклад, елементарний такий об’єкт, назва якого – dw, а тип – DialogWindow (рис. 4.6).

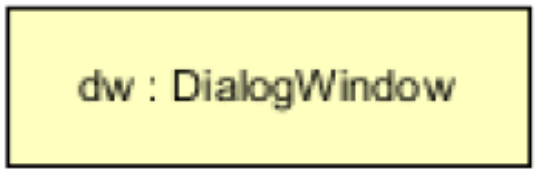

Рисунок 4.6 – Графічне зображення об’єкта на діаграмі комунікації

Далі, як і на діаграмі класів, указуються асоціації між об’єктами у вигляді різних сполучних ліній. При цьому можна явно вказати імена асоціації і ролей, які відіграють

об’єкти у цій асоціації. Додатково зображують динамічні зв’язки – потоки повідомлень. Вони подаються у вигляді сполучних ліній між об’єктами, над якими розташовується стрілка з вказівкою напряму, імені повідомлення і порядкового номера в загальній послідовності ініціалізації повідомлень.

Поведінка системи визначається на рівні окремих об’єктів, які обмінюються між собою повідомленнями, аби досягти потрібної мети або реалізувати деякий програмний сервіс. З точки зору аналітика або конструктора важливо представити у проекті системи структурні зв’язки окремих об’єктів між собою. Таке статичне представлення структури системи як сукупності взаємодіючих об’єктів і забезпечує діаграма кооперації.

На відміну від діаграми послідовності, на діаграмі кооперації зображуються лише відношення між об’єктами, що відіграють певні ролі у взаємодії. З іншого боку, на цій діаграмі не вказується час у вигляді окремого виміру. Таким чином, послідовність взаємодій і паралельних потоків може бути визначена на діаграмі кооперації за допомогою порядкових номерів для відповідних асоціацій. (Примітка: якщо необхідно явно специфікувати взаємозв'язки між об'єктами ПС в реальному часі, то кра-ще це робити на діаграмі послідовності – див. л/р 2).

На рис. 4.7. наведено варіант побудови діаграми кооперації для розглянутої вище процедури авторизації користувача ПС.

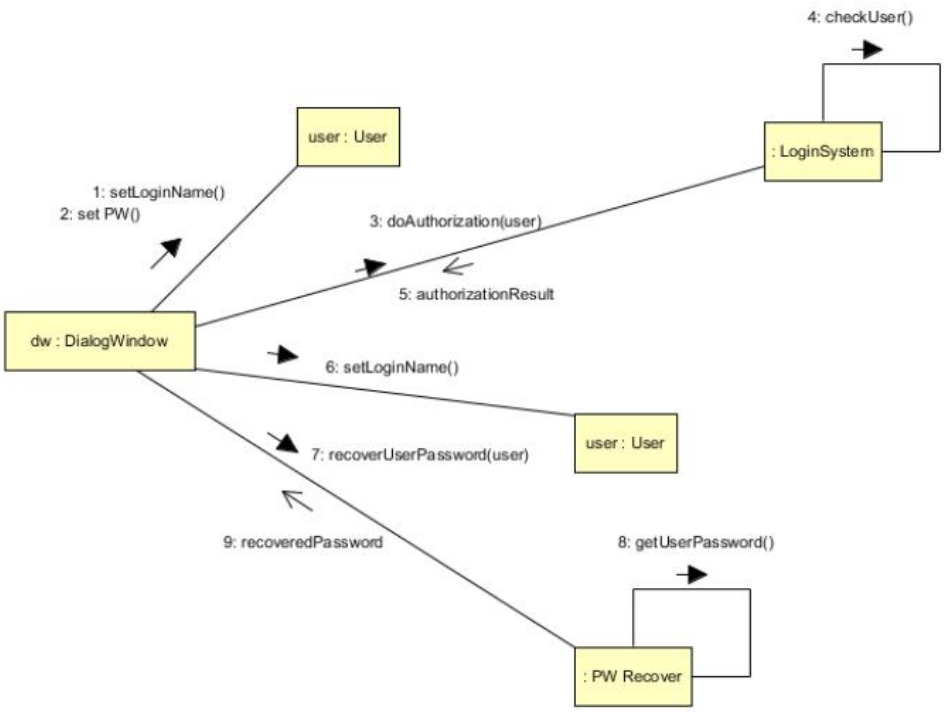

Рисунок 4.7 – Приклад побудови діаграми комунікації

Об’єкт dw класу DialogWindow посилає повідомлення setLoginName та setPW об’єкту user класу User.

Після чого об’єкт user передається системі для проведення перевірки (авторизації) – об’єкт dw посилає повідомлення doAuthorizaton з параметром user. Після чого об’єкт LoginSystem виконує метод checkUser, результат авторизації передається повідомленням AuthorizationResult.

Для відновлення пароля об’єкт dw класу DialogWindow посилає повідомлення setLoginName об’єкту user класу User. Після чого об’єкт user передається системі для проведення відновлення пароля – об’єкт dw посилає повідомлення recoverUserPassword з параметром user. Далі об’єкт PWRecover виконує метод getUserPassword, результат відновлення пароля передається повідомленням RecoveredPassword (рис. 4.7).

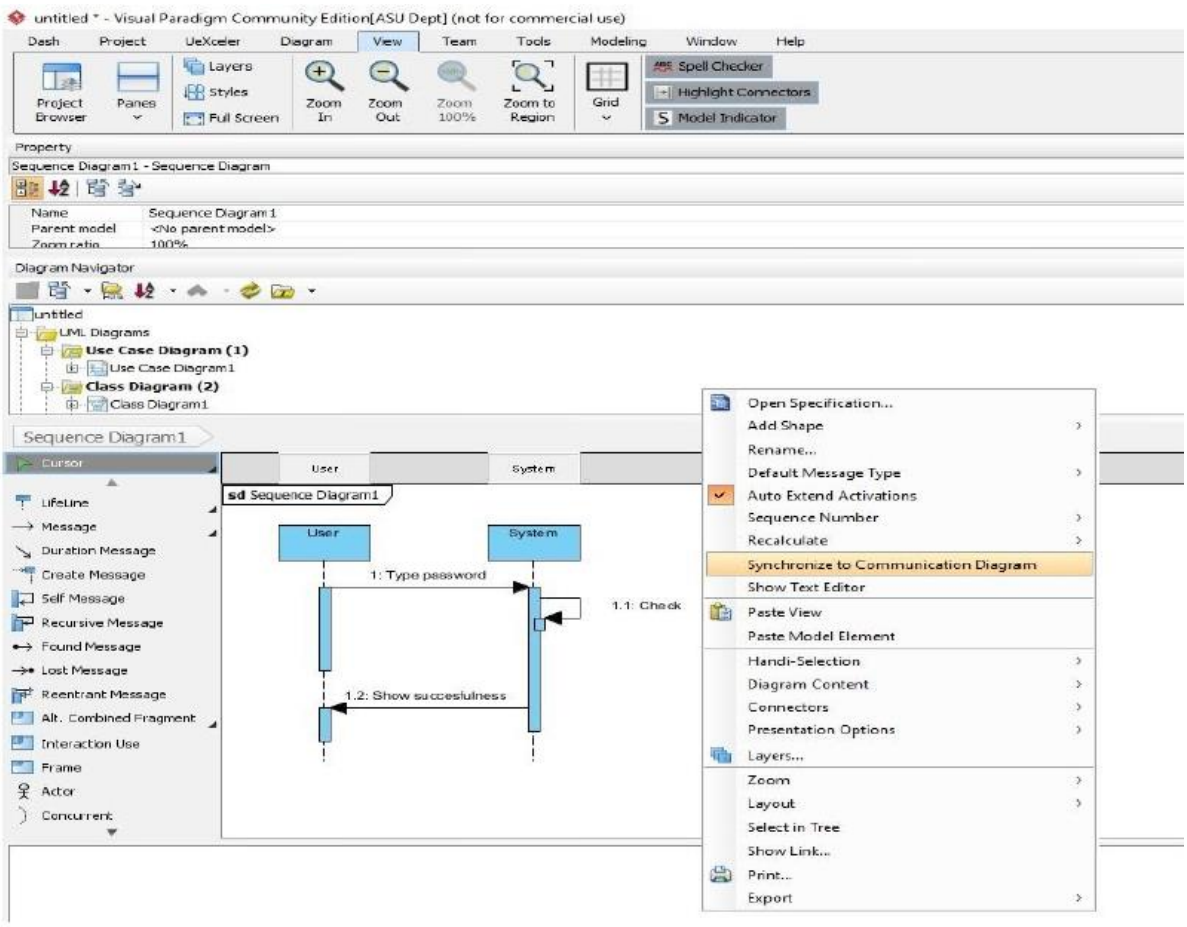

Рисунок 4.8 – Приклад інтерфейсу при автоматичній генерації діаграми комунікації

У VP також є можливість автоматичної генерації діаграм комунікації на основі діаграм послідовностей. Для цього необхідно у полі обраної діаграми послідовності натиснути правою кнопкою миші та вибрати пункт Synchronize to Communication Diagram (рис. 4.8).

Діаграми композитної структури (composite structure diagram)

Діаграма композитної (складової) структури – статична структурна діаграма, демонструє внутрішню структуру класів і, по можливості, взаємодію елементів (частин) внутрішньої структури класу. Такі діаграми використовуються спільно з діаграмами класів.

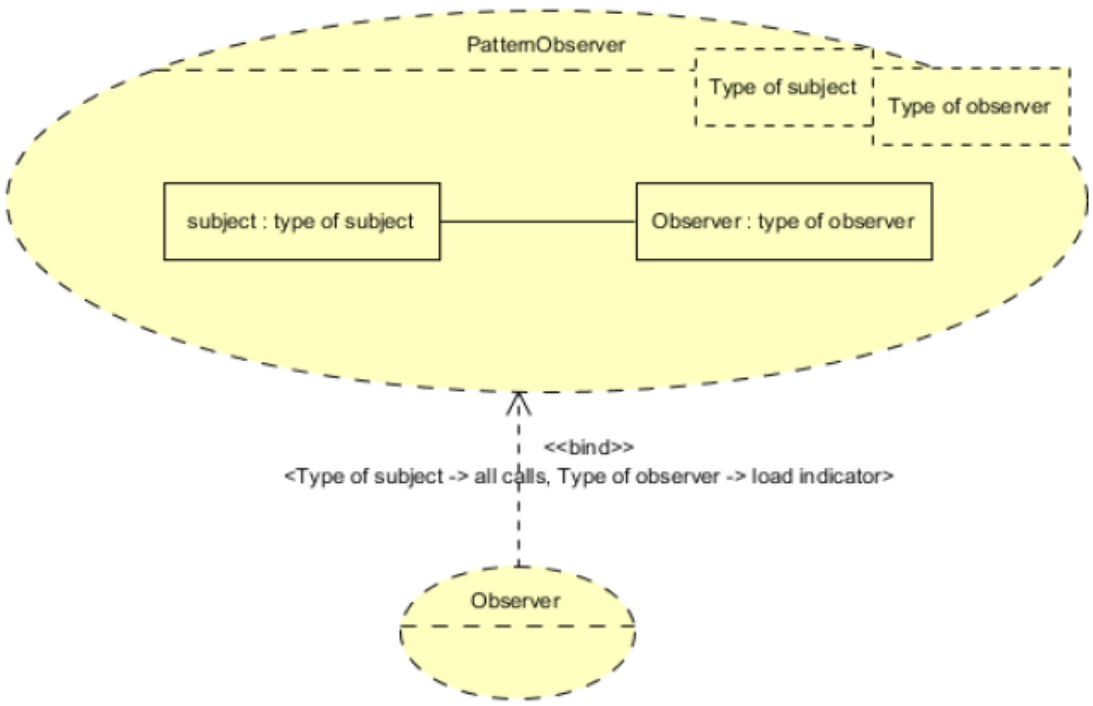

Рисунок 4.9 – Приклад побудови діаграми композитної структури (шаблон кооперації патерна Спостерігач та його зв’язування)

Як елементи опцій на цих діаграмах часто зображується класифікатор – кооперація, який призначається для опису деякої структури елементів або ролей, що виконують спеціалізовані функції та спільно забезпечують бажану функціональність. Ці елементи зображуються у вигляді різних шаблонів (патернів) проектування (рис. 4.9).

Діаграми обзору взаємодії (interaction overview diagram)

Діаграма обзору взаємодії – це комбінація діаграм діяльності та діаграм послідовності. Можна вважати діаграми обзору взаємодії діаграмами діяльності, у яких діяльності замінені невеликими діаграмами послідовності, або ж діаграмами послідовності, що розбиті за допомогою нотації діаграм діяльності для відображення потоку управління. У будь-якому випадку вони являють собою досить незвичайну суміш.

На рис. 4.10 поданий приклад побудови діаграми такого типу для процедури авторизації користувача ПС.

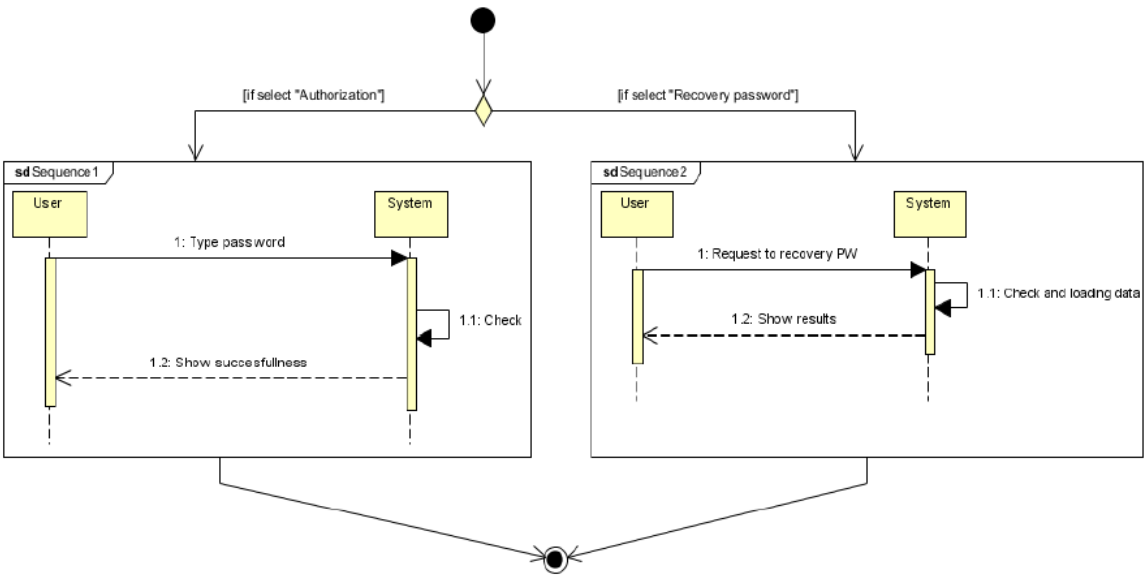

Рисунок 4.10 – Приклад побудови діаграми обзору взаємодії

4.2. Виконання лабораторної роботи

1. Розробити діаграму кінцевого автомату для обраної предметної області.
2. Розробити діаграму діяльності для обраної предметної області.
3. Розробити діаграму комунікації для обраної предметної області.
4. Розробити діаграму композитної структури для обраної предметної області.
5. Розробити діаграму обзору взаємодії для обраної предметної області.

4.3. Висновки до виконаної роботи та оформлення звіту

Зміст звіту:

1) титульний аркуш;

2) короткі теоретичні відомості;

3) результати практичного виконання, висновки та рекомендації.

Контрольні запитання

1. Які типи UML-діаграм застосовуються при проектуванні програмної системи на логічному рівні з метою моделювання динамічних аспектів ПЗ?

2. Для чого застосовуються діаграми кінцевого автомата (ДКА)? Назвіть основі графічні елементи, що мають місце для побудови ДКА, поясніть їх призначення.

3. Які види позначок дій є можливими для ДКА, що вони означають?

4. Що таке сторожова умова, навіщо вона застосовується в ДКА?

5. Для чого застосовуються діаграми діяльності (ДД)? У чому полягає різниця між застосуванням ДКА і ДД?

6. Назвіть основні графічні елементи, що мають місце для побудови ДД, поясніть їх призначення.

7. У чому полягає різниця між станом у ДКА та станом дії у ДД?

8. Що є спільного та в чому полягає різниця між такими елементами ДД як розгалуження (decision) та розділення (concurrent fork)?

9. Для моделювання яких особливостей виконання ПЗ використовується конструкція доріжки (swimlines)?

10. Для чого застосовуються діаграми комунікації (ДК)? Назвіть основні графічні елементи, що мають місце для побудови ДК, поясніть їх призначення.

11. У чому полягає різниця між ДК та діаграмами послідовностей?

12. Яким чином на ДК враховується фактор дії поточного часу?