

ЛАБОРАТОРНА РОБОТА 2

АНАЛІЗ ВИМОГ ТА РОЗРОБКА UML–ДІАГРАМ КОНЦЕПТУАЛЬНОГО РІВНЯ ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

Мета роботи – ознайомитись з методикою побудови UML-діаграм концептуального рівня проектування програмного забезпечення.

2.1. Загальні відомості

На концептуальному рівні проектування ПС відповідно до раніше визначених та задокументованих вимог (наприклад, з використанням методології RUP або Scrum [3]) встановлюються найбільш загальні характеристики системи, що розробляються, до яких належать:

- основні варіанти використання функцій ПС;
- типи її користувачів;
- послідовність взаємодії окремих користувачів і системи або окремих її підсистем (так звані сценарії);
- основні групи програмних компонентів, що реалізують певні функції системи, з рознесенням їх по рівням логічного упорядкування та взаємодії, тобто архітекту-ра ПС.

При об'єктно-орієнтованому підході до проектування ПЗ з використання UML на концептуальному рівні моделювання використовуються такі діаграми [3–4]:

- діаграми варіантів використання (або прецедентів – use case diagram);
- діаграми послідовностей (sequence diagram);
- діаграми пакетів (package diagram).

Як додатковий засіб до діаграм прецедентів, який дозволяє розширити уявлення розробників щодо подальшої програмної реалізації системи, використовують діаграми стійкості (robustness diagram).

Діаграми прецедентів (use case diagram)

Під прецедентом (case) треба розуміти фіксовану послідовність дій між користувачами та ПС або між окремими підсистемами, яка забезпечує досягнення поставлених цілей. В обох випадках вони позначаються терміном «актор» (actor). Так, на-приклад, процедура авторизації користувача на деякому інтернет-ресурсі є певним прецедентом. Для нього учасниками будуть система авторизації та користувач, а ре-зультатом – перевірка наявності в системі реєстраційного запису (account) користувача.

Кожна модель відображає взаємозалежність прецедентів та акторів, яка визначається такими типами:

- асоціація (association);
- включення (include);

- розширення (extend);
- узагальнення (generalization).

Усі ці чотири типа залежностей надані на рис. 2.1.

Відношення *асоціації* є основним типом залежностей між акторами та прецеден-тами, що вказує на те, що деякий актор відіграє певну роль у взаємодії з системою. На діаграмі асоціацій позначається суцільною прямою лінією і позначається відпо-відною назвою. На рис. 1 це відношення «realize» між актором «User» та прецеден-том «Authorization».

Відношення *включення* дає можливість відобразити те, що деякі функції одного прецеденту А обов'язково мають бути задіяні при використанні іншого прецеденту В. На діаграмі включення позначається пунктирною прямою лінією зі стрілкою на кінці, яка направлена від того прецеденту, що включає в себе деякий інший і має на-зву «include». На рис. 2.1 це відношення між прецедентом «Authorization» та «Login PW».

Відношення *розширення* дає можливість відобразити те, що при опрацюванні си-стемою деякого прецеденту Х є можливим (але не обов'язковим) використання фун-кціональності іншого прецеденту Y. На діаграмі включення позначається пунктир-ною прямою лінією зі стрілкою на кінці, яка направлена від того прецеденту, що є розширенням та має назву «extend». На рис. 2.1 це відношення між прецедентом «Authorization» та «PW Recovery».

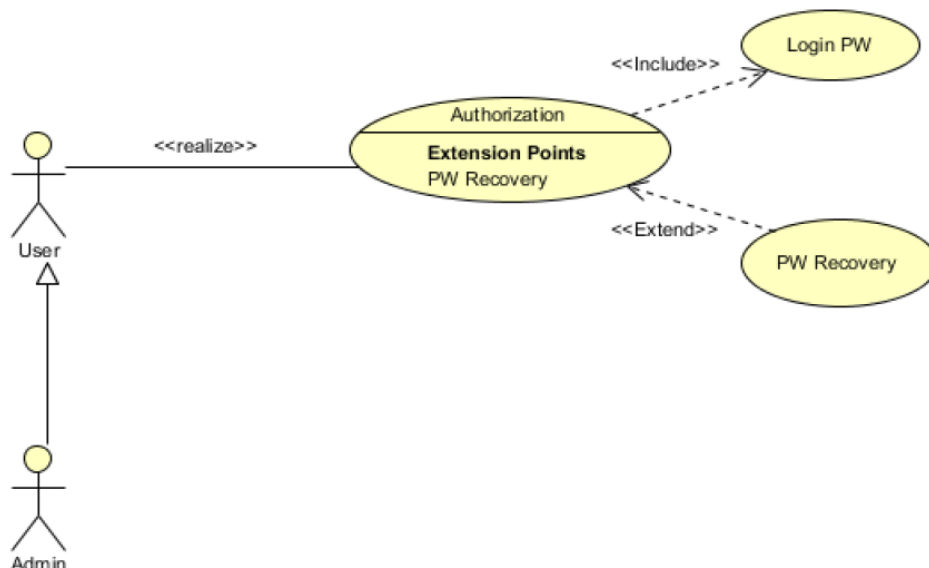


Рисунок 2.1 – Приклад побудови UML-діаграми прецедентів

Відношення *узагальнення* дозволяє показати певну ієрархію акторів та самих прецедентів, тобто позначити, який з них є супер-типом, а який – підтипом певного класу об'єктів. На діаграмі це позначається суцільною прямою лінією з трикутною стрілкою на кінці, яка направлена до прецеденту, що є супер-типом. На рис. 2.1 це відношення між акторами «Admin» та «User».




Діаграма стійкості (robustness diagram)

Прецеденти не можуть бути безпосередньо трансформовані у відповідні програ-мні об'єкти (класи). Для цього має бути використано певний проміжний рівень опи-су прецедентів, який дозволяє врахувати деякі особливості наступної програмної ре-алізації, наприклад, необхідність інтерфейсу користувача або наявність деяких фун-кцій обробки бізнес-логіки. Одним з таких засобів подальшої деталізації моделі прецедентів є *діаграма стійкості* (ці діаграми належать до процесу проектування ПЗ за стандартом ICONIX [3]).

Діаграма стійкості розробляється на базі шаблону (або патерну – pattern) проек-тування MVC (Model-View-Controller) [4]. Тобто в системі, що розробляється, мають бути наявні три типи програмних об'єктів:

- *Model* – це об'єкти, які моделюють дані предметної області;
- *View* – це об'єкти, які реалізують відображення даних із моделі;
- *Controller* – це об'єкти, які обробляють дані моделі для подальшого їх відо-браження.

Основна ідея шаблону MVC полягає у відокремленні даних від їх відображення. Таким чином, якщо у процесі розробки виникне потреба у зміні моделі даних, це ніяк не впливає на відображення (не треба буде нічого змінювати у компонентах View). У діаграмі стійкості є свої окремі позначки для відображення елементів мо-делі MVC:

-  – граничні об'єкти, які слугують для відображення даних (View)
-  – об'єкти обробки даних (Controller)
-  – об'єкти даних (Model)

На рис. 2.2 побудовано діаграму стійкості для прецеденту реєстрації нового ко-ристувача системи, де граничним об'єктом *Registration form* є певна діалогова форма (HTML-сторінка), яка з'являється у вікні браузера на комп'ютері-клієнті системи. Об'єктом-контролером *Add new* може бути, наприклад, програмний скрипт на мові PHP, який виконується на Web-сервері Apache і який за допомогою функцій оброб-ки SQL-запитів працює з об'єктом даних Account – це, у свою чергу, може бути таб-лиця БД, що знаходиться під керуванням СКБД MySQL.

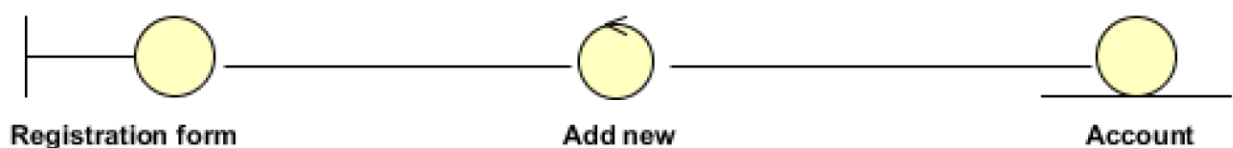


Рисунок 2.2 – Діаграма стійкості для прецеденту реєстрації нового користувача

Декілька прецедентів можуть бути пов'язані на одній діаграмі стійкості (рис. 2.3). Так, об'єкт даних Account використовується для прецеденту входу до системи. Можливим варіантом програмної реалізації є розширення прецеденту реєстрації, додавши до нього контролер *Find* для пошуку вже існуючого реєстраційного запису, тобто за його допомогою при реєстрації система буде перевіряти, чи вже існує реєстраційний запис, який хоче додати користувач.

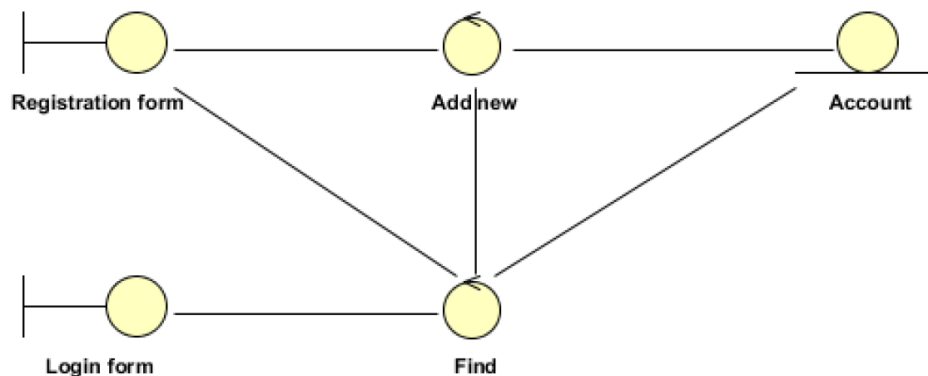


Рисунок 2.3 – Діаграма стійкості для двох прецедентів

Для відображення цих візуальних елементів у VP необхідно вибрати відповідні стереотипи для об'єктів: для View – це boundary, для Controller – це control, для Model – це entity.

Діаграми послідовності (sequence diagram)

Окрім розглянутих вище діаграм прецедентів та діаграм стійкості, для кожного прецеденту може бути побудована відповідна діаграма послідовності. Цей тип діаграм визначає певні сценарії обміну повідомленнями (або викликами відповідних функцій) між різними логічними (архітектурними) компонентами ПС.

Кожна діаграма послідовності розподіляється на декілька частин – за кількістю учасників прецеденту. Наприклад, у прецеденті Login беруть участь два компоненти: користувач (User) та система (System) (рис. 2.4). Ці компоненти позначаються прямокутниками, які знаходяться у верхній частині діаграми.

Кожен такий прямокутник має свою *лінію життя*: це вертикальна лінія, яка на-правлена донизу від цього компонента та відображає час його активності у системі. Передача повідомлень, що ініціюють певну функцію, позначається за допомогою горизонтальної стрілочки між лініями життя відповідних компонентів (1: Type Password), а результат виконання цієї функції показано пунктирною стрілочкою повернення управління по цьому сценарію (1.2. Show successfulness).

Відповідно до діаграми послідовності прецеденту Login маємо:

1. Користувач вводить логін та пароль.
2. Система перевіряє правильність вводу.
3. У разі розбіжності введених даних з існуючим реєстраційним записом вивести вікно помилки

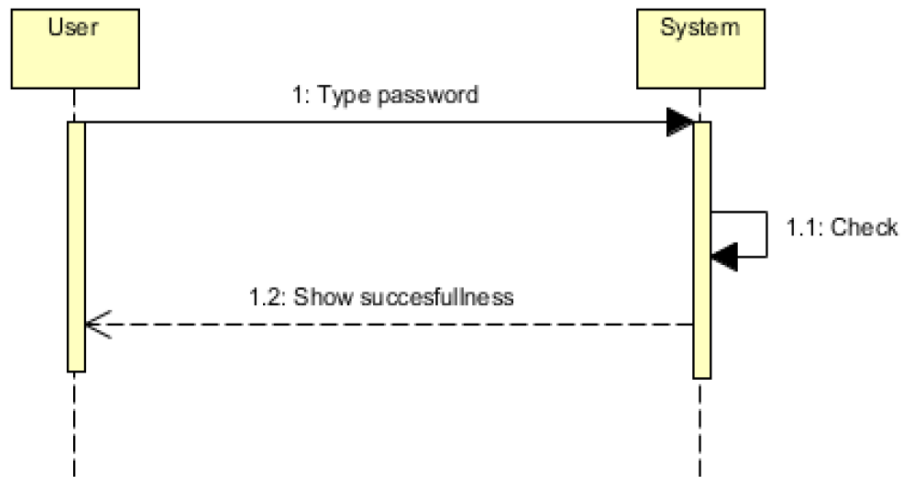


Рисунок 2.4 – Діаграма послідовності для прецеденту Login

Діаграма пакетів (package diagram)

У процесі об'єктно-орієнтованої розробки ПС проводиться розподіл програмного рішення на окремі підсистеми, що уможлиблює розподіл роботи між членами ко-манди розробників і забезпечує подальше багаторазове використання програмного коду. Засобом моделювання в UML, який дає змогу агрегувати окремі компоненти в окремі групи (підсистеми) залежно від їх логічного призначення в системі, є поняття пакета (package).

Кожен пакет володіє всіма елементами, що належать до нього, при цьому сам па-кет може входити до складу іншого пакета. Графічно пакет позначається відповід-ним прямокутником, який має унікальну назву та з'єднується з іншими пакетами за допомогою зв'язку Containment, що у VP позначає відношення вкладеності (рис. 2.5).

На цій діаграмі зображено пакет *Authorization*, до складу якого входять два інші пакети: *Login/PW* та *PW Recovery*.

Крім відношення вкладеності, деякі пакети можуть знаходитися у певній функці-ональній залежності один від одного, коли компоненти, що знаходяться в одному пакеті, використовуються в іншому. Така залежність позначається на діаграмі пунк-тирною лінією зі стрілкою на кінці. Для підвищення рівня інформативності діаграм пакетів на них (до речі, як і на інших UML-діаграмах) можливо використовувати коментарі.

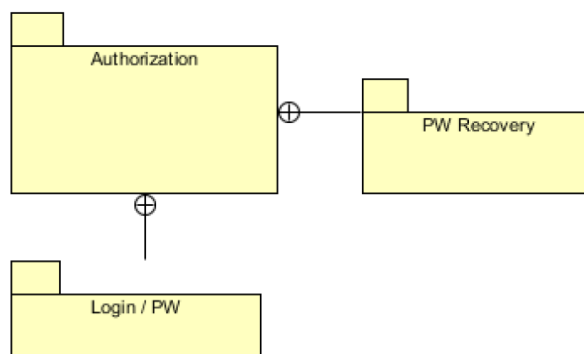


Рисунок 2.5 – UML- діаграма пакетів з відношенням вкладеності

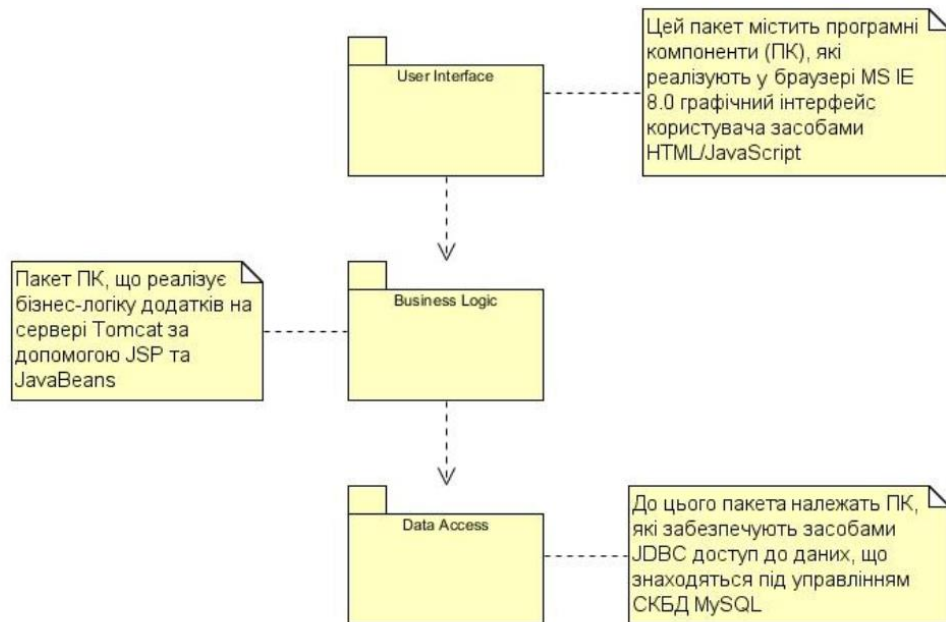


Рисунок 2.6 – Коментована UML-діаграма пакетів з відношенням залежності

Такий коментар позначається спеціальним графічним елементом, який містить відповідний текст. На рисунку 2.6 наведено приклад коментованої UML-діаграми пакетів з відношенням залежності, що відображає типову архітектуру Web-базової програмної системи для роботи з базою даних та сервером додатків.

2.2. Виконання роботи

На основі аналізу та текстової специфікації вимог (див. л/р 2) потрібно: побудувати модель прецедентів для наданої предметної області;

провести функціональний аналіз основних прецедентів та побудувати відповідні діаграми стійкості;

для основних прецедентів побудувати діаграми послідовності;

побудувати загальну діаграму пакетів для всієї ПС, що має бути спроектована.

2.3. 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. Для чого у процесі проектування ПЗ використовують моделі прецедентів? Що має відобразити така модель ?
2. На підставі якої інформації будується модель прецедентів?

3. Що таке «актор» та «прецедент»? Які типи відношень можуть бути визначені між ними?
4. Який тип відношень між акторами і прецедентами є найбільш поширеним? Навести приклади.
5. У чому полягає спільність і у чому є різниця між відношенням включення та розширення на діаграмі прецедентів? Навести приклади.
6. Для чого використовується відношення узагальнення? Навести приклади.
7. Який сенс у процесі моделювання ПЗ має застосування діаграм стійкості?
8. Назвіть основні графічні елементи, що використовуються при побудові діаграми стійкості. Який шаблон проектування ПЗ вони використовують?
9. Для чого при проектуванні ПЗ використовуються діаграми пакетів? Які типи відношень існують між окремими пакетами?