ЛАБОРАТОРНЕ ЗАНЯТТЯ № 02

ТЕХНОЛОГІЯ СТВОРЕННЯ ПРОГРАМНИХ ПРОДУКТІВ

Тема: **ОБ'ЄКТНО-ОРІЄНТОВАНИЙ АНАЛІЗ ТА ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ. ДІАГРАМИ ВАРІАНТІВ ВИКОРИСТАННЯ.**

Мета: дослідження предметної області, вивчення процесу побудови діаграм прецедентів за допомогою інструмента об'єктного моделювання StarUML.

Програмне забезпечення: ОС Windows/Linux, StarUML (https://staruml.io/).

Теоретична частина

UML (Unified Modeling Language) — уніфікована мова моделювання, що використовується розробниками програмного забезпечення для візуалізації процесів та роботи систем.

Це не мова програмування, скоріше набір правил та стандартів для створення діаграм. Вони дозволяють розробникам програмного забезпечення та інженерам «говорити однією мовою», не заглиблюючись у фактичний код свого продукту. Складання діаграм за допомогою UML — це чудовий спосіб допомогти іншим швидко зрозуміти складну ідею чи структуру.

UML-діаграми поділяються на 14 типів:

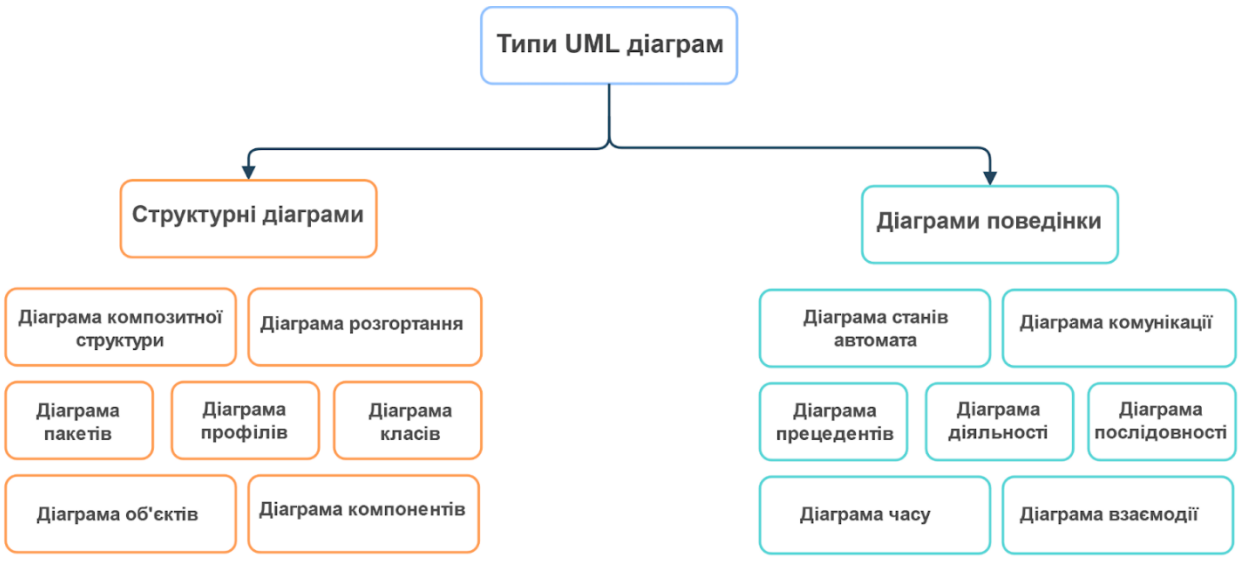


Рис.1 Основні типи UML діаграм

Діаграма варіантів використання

Діаграма варіантів використання — (діаграма прецедентів, сценарій використання, use case) — дозволяє уявити типи ролей та їх взаємодію із системою. Проте не показує порядок виконання кроків. Зображує функціональні вимоги (те, що система може зробити) з точки зору користувача. Може описуватись текстом або у вигляді діаграми.

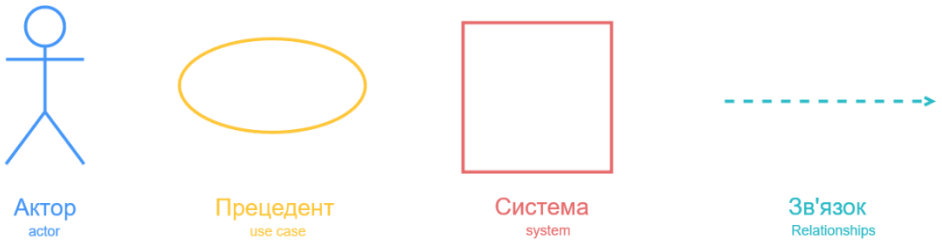


Рис.2 Основні позначення діаграми варіантів використання

Діаграми використання розробляються на ранній стадії проєктування системи та призначені для:
Простого пояснення роботи системи, створення та погодження ТЗ.
Формування функціональних (що має робити) вимог до системи
Створення основи для документації та тестування.

Складові діаграми варіантів використання

Діаграми варіантів використання складаються з 4 об’єктів: актор, прецедент (варіант використання), система, зв’язок.

Актор. Поняття когось, або чогось, що взаємодіє з системою, але не належить до неї (знаходиться за межами системи). Зазвичай позначається у вигляді стилізованого чоловічка. Рідше у вигляді прямокутника класу з написом <<actor>>.

Усіх акторів умовно можна розділити на первинних (ті, що ініціюють взаємодію з системою) та вторинних (ті що реагують на взаємодію). Первинні зазвичай зображуються зліва, а вторинні — справа від системи. Оскільки будь-яка система може взаємодіяти не лише з людьми, актор не завжди позначає користувача-людину. Він може позначати іншу систему або пристрій з яким взаємодіє.

Часто виникає плутанина між поняттями актор та користувач:

- *актор* — це поняття, що представляє клас користувачів (узагальнення групи користувачів), а не конкретного користувача, та може поєднувати в собі декілька ролей. Наприклад актор — працівник компанії може мати ролі інженер, менеджер, директор;
- *користувач* — це тип актора або його конкретна реалізація. Декілька користувачів можуть грати одну роль, тобто бути одним актором. Наприклад Іван та Роман — студенти. Також один користувач може належати до різних акторів, тобто виконувати різні ролі. Наприклад, в системі університету актор може бути викладачем в одному випадку, і науковим керівником в іншому.

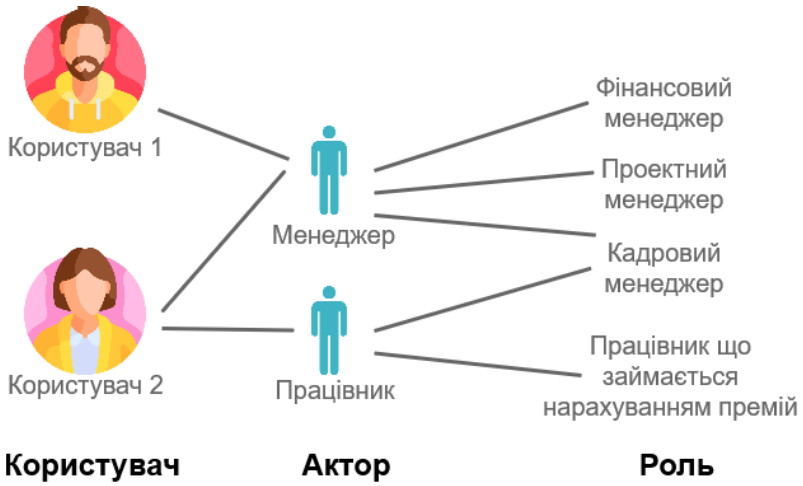


Рис. 3 Приклад користувача, актора та ролі в UML

Випадок використання (прецедент). Прецеденти визначають очікувану поведінку та відповідають на питання, що робить система. Представляють набір можливих функцій, дій або завдань. Зображується у вигляді еліпса з назвою дії (дієсловом) у ньому. Прецедент вказує, що має трапитись, проте не відповідає на запитання, як це має статись.

До прикладу: в системі банківського додатка прецедентами можуть бути: перевірити баланс, переказати кошти, оплатити рахунок та ін. Б

Система. Те, що моделюється. Це може бути сайт, мобільний додаток або навіть модуль програмного забезпечення. Зображується у вигляді прямокутника з назвою системи у верхній частині.

Коментар розширює функціональність, надає підказки та умови.

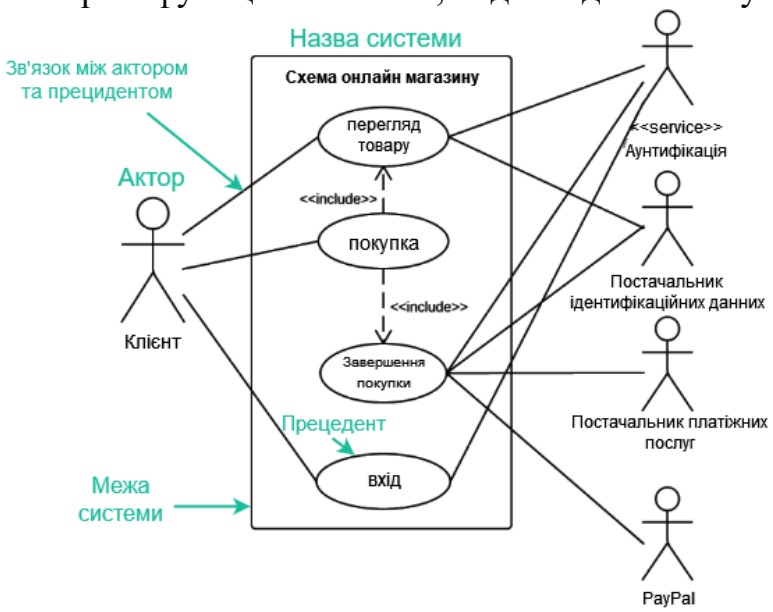


Рис.4 Складові діаграми послідовності

Зв'язки (відношення). Усі актори мають бути пов'язані з прецедентом. Проте не усі прецеденти повинні бути пов'язані з акторами. Частіше всього для зображення зв'язку використовується суцільна лінія.

У діаграмі варіантів використання існує 4 типи зв'язків:

1. Асоціація (Association). Звичайний зв'язок актора та прецеденту. Позначається суцільною лінією без напису (стереотипу). Незалежно від типу зв'язку, будь-який актор повинен бути пов'язаний принаймні з одним (можна з декількома) варіантом використання. Кілька акторів можуть бути пов'язані з одним варіантом використання.

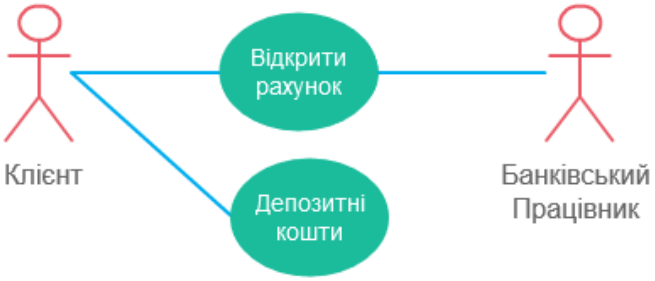


Рис.5 Асоціація між актором та варіантом використання

2. Розширення (Extend). Показує додаткову функціональність або можливий не обов'язковий варіант поведінки системи. Базовий прецедент має сенс сам по собі, не залежить від розширюючого і може існувати без нього. Відношення розширення позначається пунктирною лінією зі звичайним вказівником, що вказує на базовий прецедент та стереотипом (написом) <<extend>>. Розширюючий прецедент активується лише за виконання умови.

Наприклад: прецедент «хибний пароль» можливий лише при введенні невірного паролю. Відповідає extensions в текстовому варіанті.

Рис.6 Приклад зв'язку розширення з умовою

Точки розширення (extension points). При використанні відношення розширення базовий варіант надає точки розширення (те саме, що extensions в текстовому описі) для розширюючого прецеденту. За бажанням вони можуть бути описані в окремому розділі базового прецеденту.

Рис.7 Приклад зв'язку розширення з точками розширення

Якщо існує умова розширення (condition, те саме, що preconditions в текстовому варіанті) її можна описати в коментарі. Коментар з'єднується з пунктирною лінією з умовою розширення.

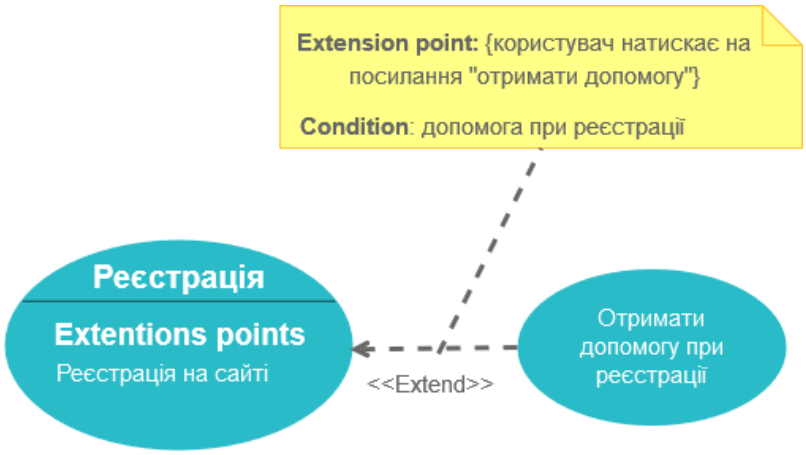


Рис.8 Приклад використання коментаря

3. Включення (Include). Показує, що поведінка одного прецеденту включається як складовий компонент у послідовність поведінки іншого прецеденту. Ілюструє, що саме використовує базовий варіант для виконання операції. На відміну від зв'язку розширення, дочірній варіант у зв'язку include має бути обов'язковим для базового. Відношення включення використовується для уникнення дублювання однакових прецедентів та додає функціональність, не вказану в базовому. Відношення позначається пунктирною лінією зі стрілкою та стереотипом <<include>>, що вказує на включений варіант.

Включення добре ілюструє сценарій відновлення працездатності комп'ютера (припустимо, що інших варіантів немає):

- ремонт або заміна апаратних компонентів;
- виявлення та видалення вірусу;
- перевстановлення системи.

Рис.9 Включення між варіантами використання

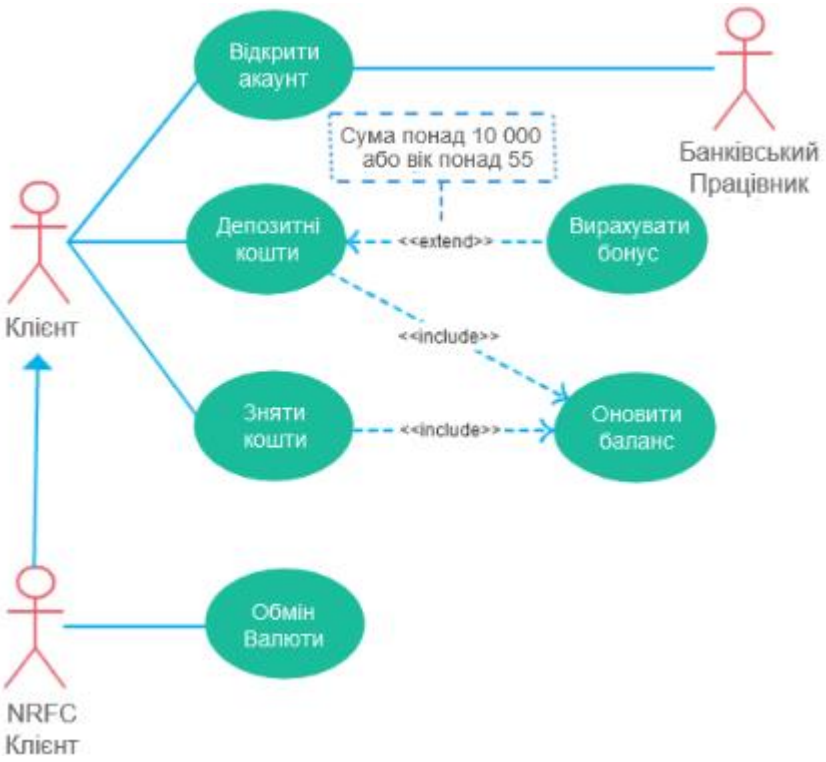


Рис.10 Включення дочірнього прецеденту «оновити баланс» до декількох прецедентів-предків

4.Генералізація (Generalization). Генералізація (успадкування, узагальнення) це батьківсько-дочірні відношення. Генералізація позначається суцільною лінією з трикутним не зафарбованим вказівником, що вказує на прецедент-предок.

Властивості відносин узагальнення:

- дочірні прецеденти мають всі властивості предків;
- для одного предка може існувати декілька дочірніх прецедентів;
- може бути кілька батьків (множинне успадкування).

Узагальнення актора. В узагальнені актора один актор може успадкувати роль іншого. Нащадок успадковує всі варіанти використання предка, проте може мати і власні унікальні варіанти.

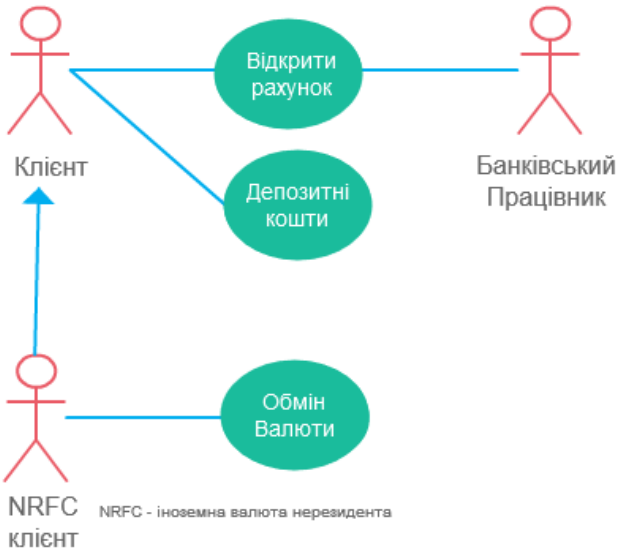


Рис.11 Узагальнення актора

Узагальнення варіанту використання. Схоже на узагальнення актора. В цьому випадку поведінка предка успадковується нащадком.

Використовується, коли існує спільна поведінка між двома варіантами використання. Наприклад, варіанти «Оплата банківською карткою» та «Оплата через Google Pay» можна узагальнити до «Оплата рахунку»

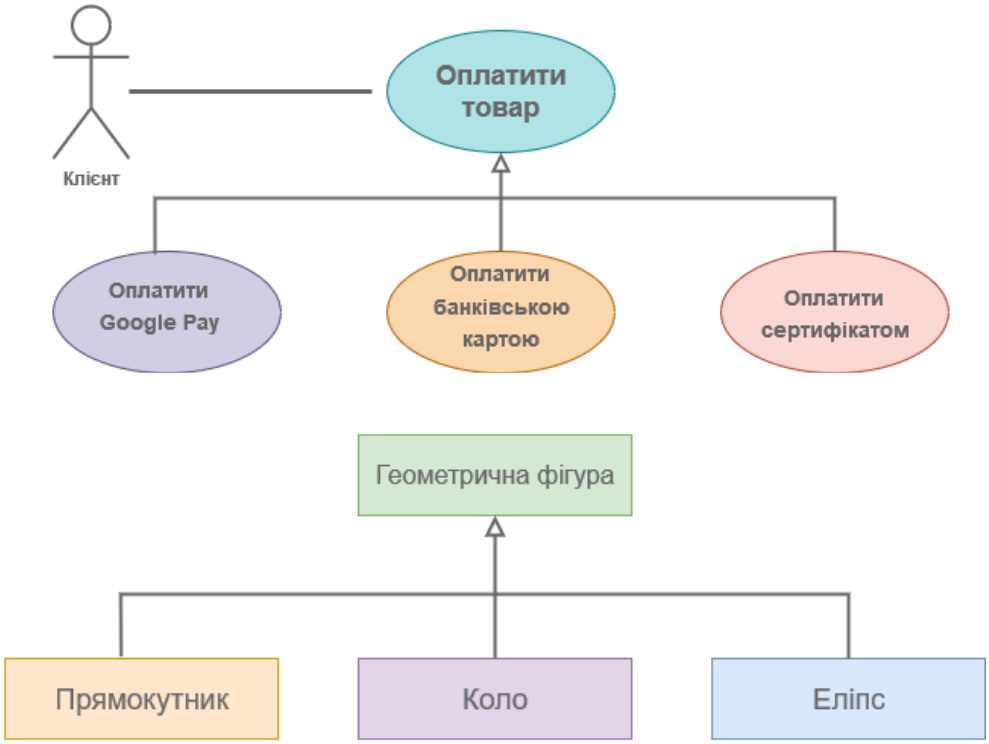


Рис.12 Узагальнення варіанту використання

Рекомендації зі створення діаграми варіантів використання

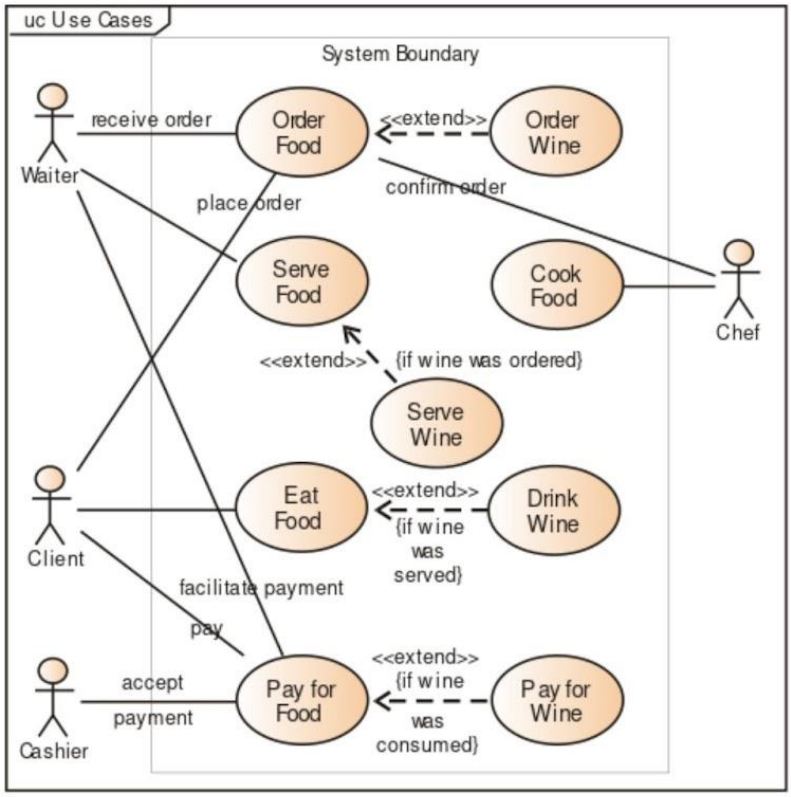
- Зручним способом моделювання є використання сервісу Draw.io. Також можливо створювати діаграму з телефону, через додаток Lucidchart. Після створення діаграми в мобільному додатку її можна виділити (ctrl +A) та вставити (ctrl +C) в Draw. io через комп'ютер.
- Якщо можливо, розміщуйте прецеденти в логічному послідовному порядку (наприклад, зверху донизу).
- За бажання, використовуйте кольори для розфарбування елементів діаграми.
- Лінії відношень не обов'язково мають бути прямими. Вони можуть утворювати гострі, або закруглені кути та повороти. Див. 1 2.
- Починайте моделювання з узагальненого варіанту і лише після цього додавайте нові деталі.

Поширені помилки

- Кожен актор очікує, що система поводитиметься строго певним, передбачуваним чином. Якщо очікується один результат, а отримується інший, таке моделювання є неправильним.

• Діаграма використання має розроблятись з точки зору користувача, а не програміста. У сценаріях повинні використовуватися назви елементів управління видимі користувачеві. Небажано зображати деталі реалізації, незрозумілі замовнику;

Приклади діаграм варіантів використання



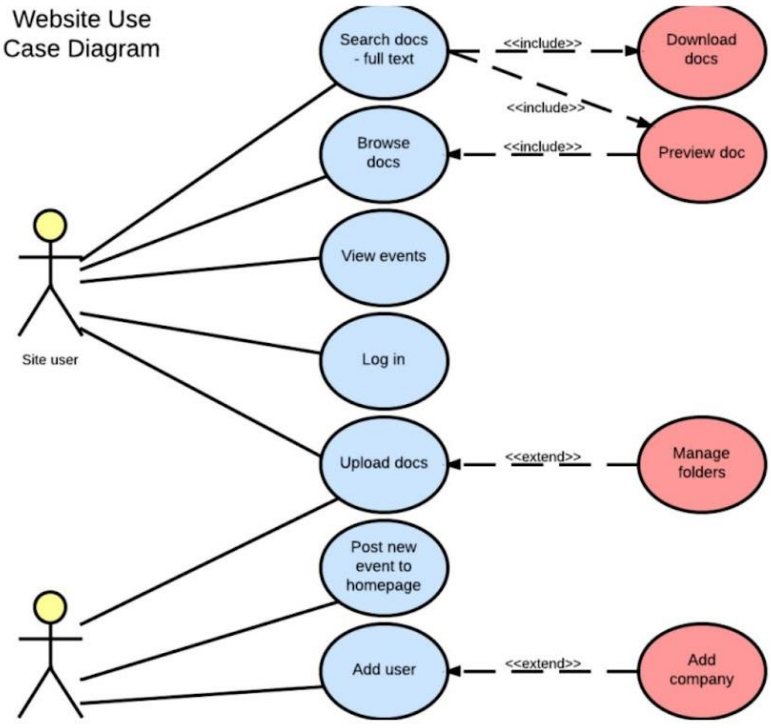


Рис.13 Приклади діаграм використання

В ідеальному світі юзкейси складає системний аналітик на основі правил, сформульованих бізнес-аналітиком після спілкування із замовником. У реальному світі таких позицій у команді може не бути через обмеження в часі та бюджеті, але проблема передачі знань усередині команди на проекті залишається. У таких випадках писати юзкейс може дизайнер, тестувальник, продакт та інші люди. Тому також можна зустріти і текстові формати опису.

Основний потік

Визначення: безумовний набір кроків, які описують, як можна досягти цілі сценарію використання та задовольнити всі відповідні інтереси зацікавлених сторін. Кожен крок є важливим для досягнення мети сценарію використання (жоден крок не можна пропустити), і кожен крок є успішним

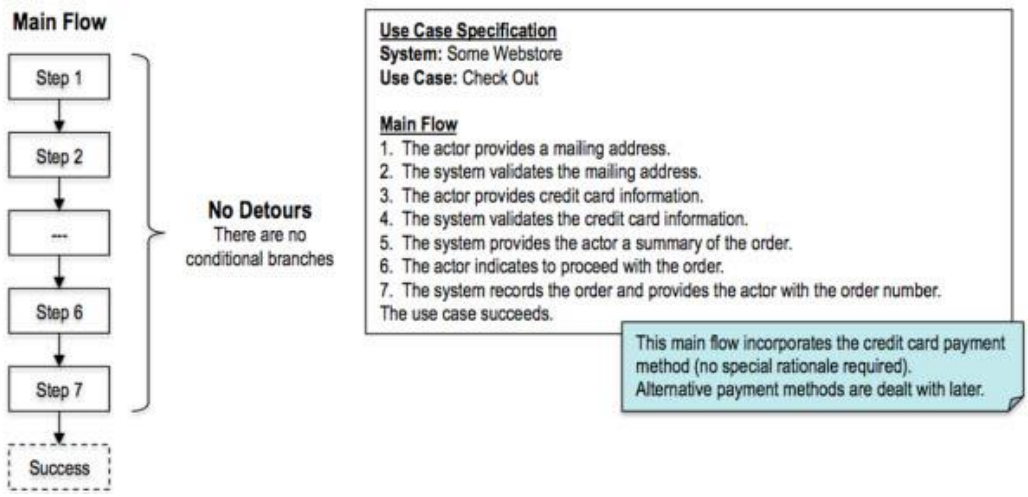


Рис.15. Приклади діаграм використання

Альтернативний потік

Визначення: умовний набір кроків, які є альтернативою одному або декільком крокам в іншому потоці (альтернативний потік виконується замість іншого кроку або кроків), після чого варіант використання продовжує досягати своєї мети.

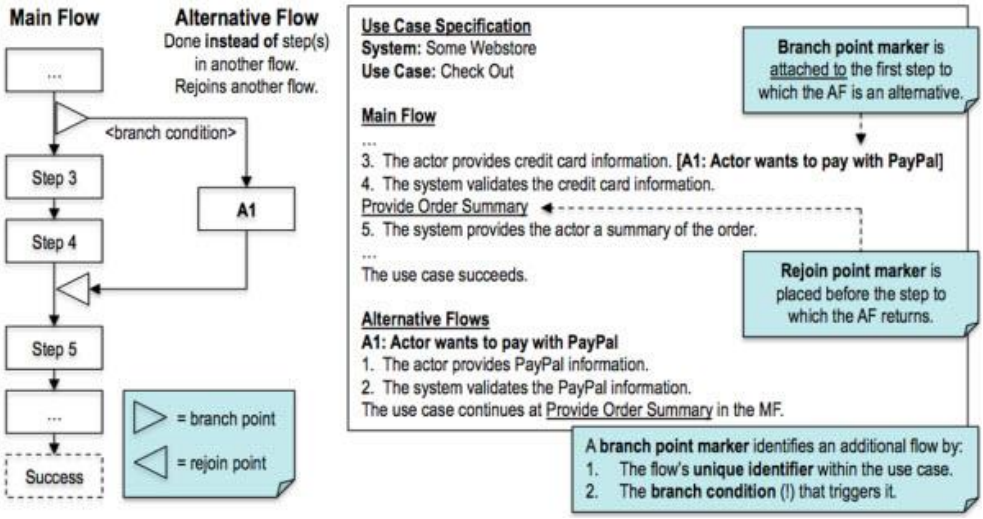


Рис.15. Приклади діаграм використання

Приклади документального офррмлення юзкейсів

Єдиного формату для написання юзкейсів не існує. Кожна компанія сама визначає собі формат і рівень деталізації. Тим не менш найбільш узагальнений варіанти може виглядати наступним чином:

Рис.15. Приклади оформлення текстового варіанту юзкейсу

В ЗАВДАННІ ДО ПРАКТИЧНОЇ РОБОТИ НАВЕДЕНІ ШАБЛОНИ ДЛЯ ОФОРМЛЕННЯ ТЕКСТОВОГО ПРЕДСТАВЛЕННЯ ЮЗКЕЙСУ

Порядок виконання роботи:

1. Опрацюйте теоретичний матеріал, засвойте концепції роботи із «Use Case» діаграмами (на основі лекційного матеріалу/на основі наданого теоретичного блоку до лабораторної).
2. Опрацюйте документацію програмного застосунку «StarUML» для побудови діаграм прецендентів за наступним посиланням: https://docs.staruml.io/working-with-uml-diagrams/use-case-diagram#extend

Група 344/344ск

Для попередньо імплементованого проекту в розрізі попередніх курсів/обчислювальної практики (у звіті надати посилання на репозитарій проекту):

3. Здійснити детальний аналіз функціональних вимог до проектуємого/розробляємого (для випадку розробляємого проекту вказати посилання на git-репозиторій проекту) програмного забезпечення та **провести побудову «use case» UML-діаграм**и, що концептуально відображала би функціональні вимоги через сутність «прецеденти» та ролі через сутність «актор».
   Врахувати можливість наявності різних типів залежностей на діаграмі.
4. Для довільно обраного прецеденту (1 шт.) **здійснити опис базового та альтернативних сценаріїв** у відповідності до вкладених прикладів шаблону.
5. Оформити звіт по результатам виконаної роботи.

Група 317

В розрізі обраної теми проекту для певної предметної області (із запропонованого стеку) або наданого проекту викладачем (https://github.com/ita-social-projects/EPlast; https://github.com/ita-social-projects/GreenCity):

3. Здійснити детальний аналіз функціональних вимог реалізованого програмного забезпечення та **провести побудову «use case» UML-**

діаграми, що концептуально відображала би реалізовані функціональні вимоги через сутність «прецеденти» та ролі через сутність «актор». Врахувати можливість наявності різних типів залежностей на діаграмі.

6. Для довільно обраного прецеденту (1 шт.) **здійснити опис базового та альтернативних сценаріїв** у відповідності до вкладених прикладів шаблону.
7. Оформити звіт по результатам виконаної роботи.

Контрольні питання:

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.

Література:

1. Grady Booch, James Rumbaugh, Ivar Jacobson. Unified Modeling Language User Guide. Addison-Wesley, 1999. 496 p. ISBN: 0-201-57168-4.
2. Martin Fowler. UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley, 2004. 208 p. ISBN: 0-321-19368-7.
3. Craig Larman. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Prentice Hall, 2004. 736 p. ISBN: 0-13-148906-2.
4. Grady Booch, James Rumbaugh, Ivar Jacobson. The Unified Modeling Language Reference Manual. Addison-Wesley, 1999. 720 p. ISBN: 0-201-30998-X.
5. Doug Rosenberg, Kendall Scott. Applying Use Case Driven Object Modeling with UML: An Annotated e-Commerce Example. Addison-Wesley, 2001. 512 p. ISBN: 0-201-60489-7.
6. Jean-Marc Nerson. Object-Oriented Software Construction. Prentice Hall, 2002. 704 p. ISBN: 0-13-629155-4.
7. Jim Arlow, Ila Neustadt. UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design. Addison-Wesley, 2005. 624 p. ISBN: 0-321-26779-8.
8. Michael Blaha, James Rumbaugh. Object-Oriented Modeling and Design with UML. Prentice Hall, 2004. 416 p. ISBN: 0-13-196845-0.
9. Alan Dennis, Barbara Haley Wixom, David Tegarden. Systems Analysis and Design: An Object-Oriented Approach with UML. Wiley, 2014. 544 p. ISBN: 978-1-118-56497-6.
10. Craig Larman. UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design. Prentice Hall, 2005. 736 p. ISBN: 0-13-148906-2.
11. Richard F. Schmidt. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Pearson, 2019. 480 p. ISBN: 978-0131489066.
12. Dan Pilone, Neil Pitman. UML 2.0 in a Nutshell. O'Reilly Media, 2005. 258 p. ISBN: 0-596-00795-7.
13. Doug Rosenberg, Matt Stephens. Use Case Driven Object Modeling with UML: A Practical Approach. Apress, 2007. 648 p. ISBN: 978-1-59059-774-3.
14. Grady Booch, James Rumbaugh, Ivar Jacobson. The Unified Modeling Language User Guide (2nd Edition). Addison-Wesley, 2005. 704 p. ISBN: 0-321-26777-1.