Об'єктно-орієнтовне моделювання та проектування складних систем

Лекція 2 – Що таке ООП

Ніколаєнко Дмитро Володимирович

Зміст

- Способи використання UML
- Нотація та метамодель UML
- Різні погляди на UML
- UML описовий чи розпорядчий?
- UML недостатньо?
- Типи діаграм UML
- Ще раз про підходи до використання UML

Вступ

Основна причина виникнення графічних мов моделювання полягає в тому, що мови програмування не забезпечують належний рівень абстракції, здатний полегшити процес моделювання.

Способи використання UML

Три способи використання UML

1

Режим ескізу

2

Режим проектування

3

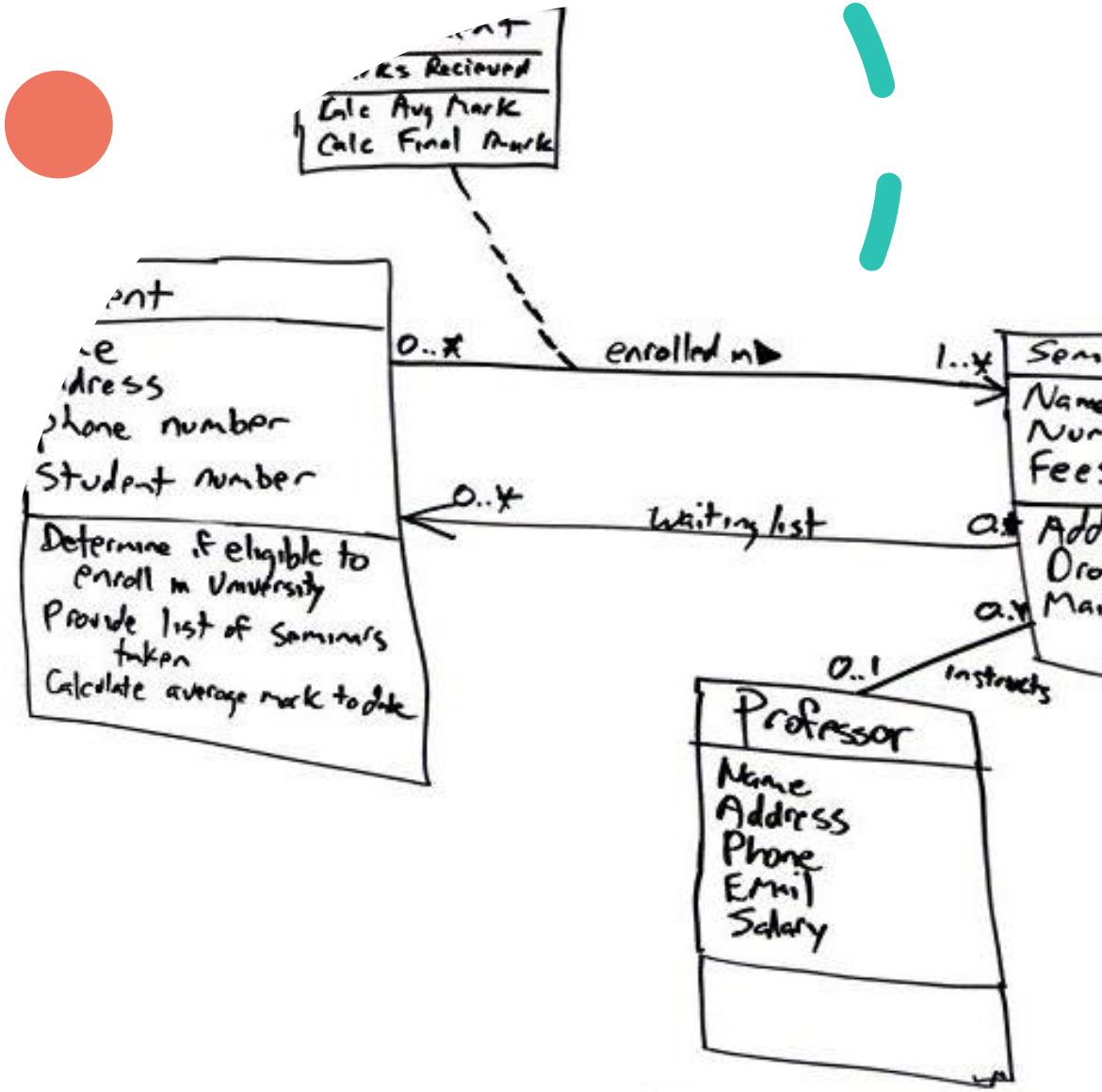
Режим мови програмування

Режим ескізу

Використовується як під час прямої розробки (forward - engineering) коли діаграми будуються до написання коду, так і під час зворотної розробки (reverse - engineering).

Режим ескізу

- Ви обговорюєте та моделюєте не всю систему, а тільки найважливіші її частини, які хочете донести до колег
- Головну роль відіграє саме процес передачі інформації, її прозорість та доступність, не повнота
- Інструментами в такому режимі виступають полегшені засоби малювання
- Часто розробники не дуже дотримуються всіх правил UML

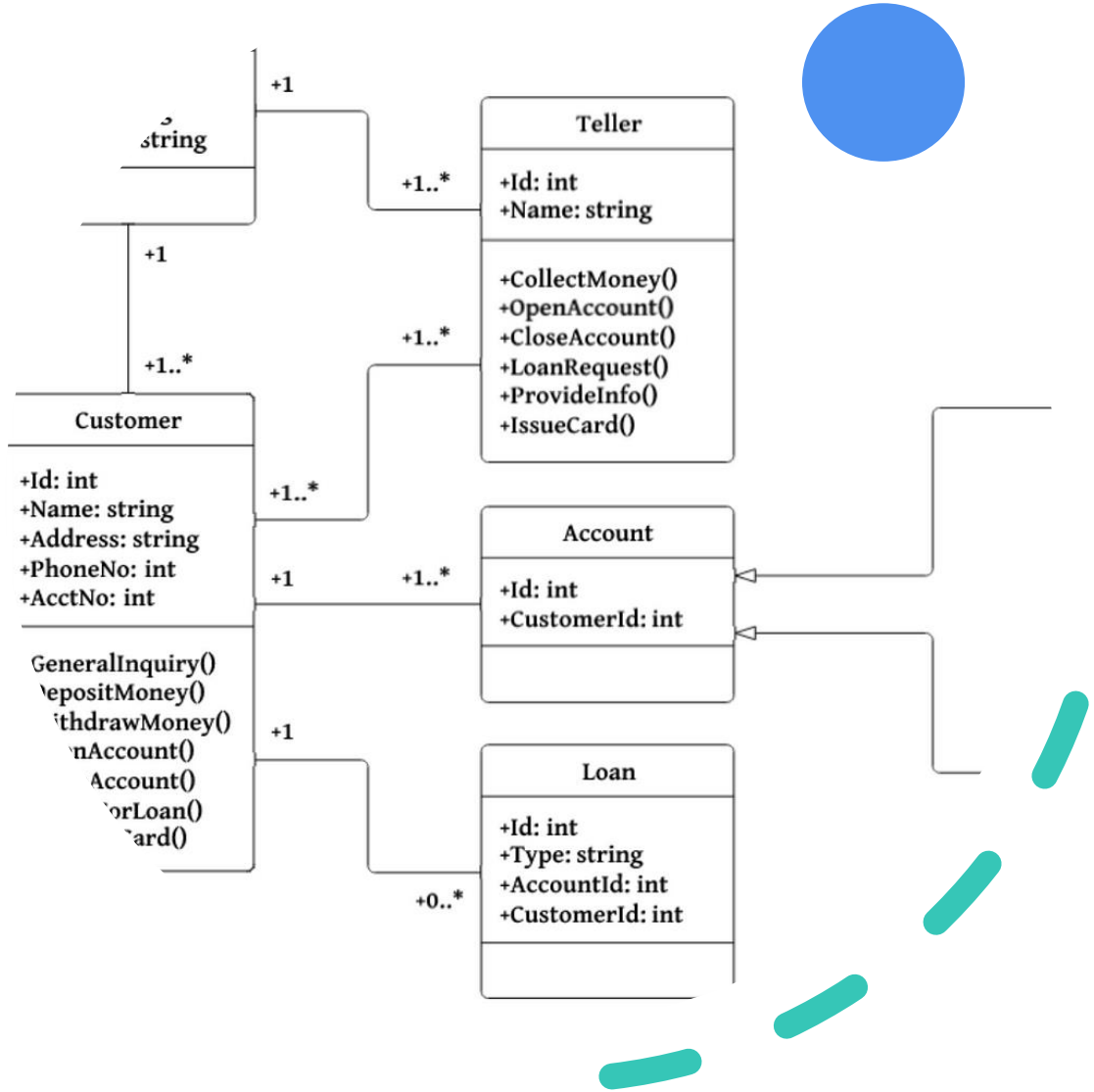


Режим проектування

Використовується як під час прямої розробки (forward - engineering) коли діаграми будуються до написання коду, так і під час зворотної розробки (reverse - engineering).

Режим проектування

- Націлений на повноту інформації
- Модель має бути достатньо детальною, аби розробник мав можливість слідувати їй особливо не задумуючись та не витрачаючи час на створення архітектури.
- Як правило виконується старшим розробником, який створює моделі для команди розробників.



Режим проектування

Під час розробки моделей необхідно використовувати складний інструментарій, адже потрібно підтримувати детальність та відповідати вимогам.

Межа між ескізом та проектуванням є доволі розмитою. Проте ескізи виконуються неповними аби підкреслити саме важливі моменти.

Режим програмування

Чим довше дизайнер працює з UML тим точніші та детальніші моделі він створює і поступово процес програмування перетворюється на більш механічну роботу.

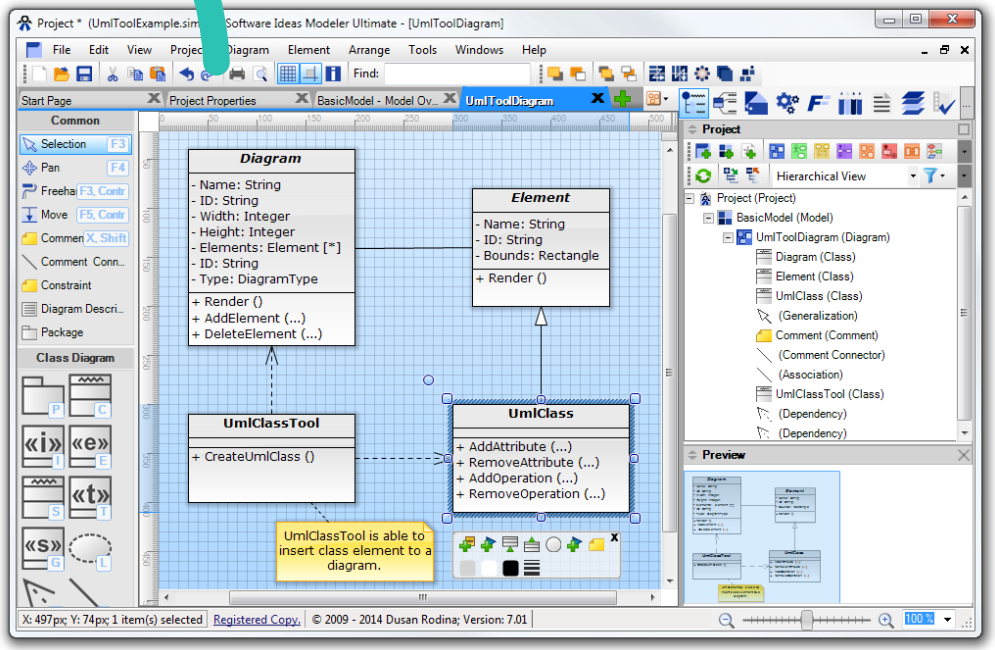
В решті решт дизайн стає настільки повним та описує всю систему, таким чином UML перетворюється на мову програмування.

Режим програмування

CASE-засоби дозволяють створювати програмний код використовуючи дизайн у вигляді діаграм UML

CASE – computer-aided software engineering

В таких засобах діаграми компілюються в програмний код, а UML стає мовою програмування.



Нотація та метамодель UML

Нотація та метамодель UML

- **Нотація** – сукупність графічних елементів, які використовуються під час створення моделей

- **Метамодель** – закони та концепції, що приховуються за кожною діаграмою

Різні погляди на UML

Різниця в поглядах на UML

- UML це точна мова, яка має передавати всі аспекти системи, аби не було непорозумінь
- UML дизайн завжди приймає до уваги вимоги мови програмування, за допомогою якої буде створюватись система

Різниця в поглядах на UML

- UML потрібно застосовувати без прив'язки до мови програмування
- Діаграми вторинні, первинна метамодель UML
- UML в якості мови програмування навряд стане широко використовуватись

Різниця в поглядах на UML

- Під час читання UML моделей дуже важливо розуміти саме точку зору автора та контекст в якому ці моделі створені
- Детальні моделі під час прямого проектування складно впроваджувати та вони сповільнюють процес розробки

UML описовий чи розпорядчий?

UML описовий чи розпорядчий?

Мова з **розпорядчими** (предписуючими) **правилами** - керується офіційними основами, які чітко встановлюють що є, а що не є допустимим.

Наприклад: мова програмування C++

Prescriptive rules

Мова з **описовими правилами** – це мова, правила якої розпізнаються по тому, як люди його використовують на практиці.

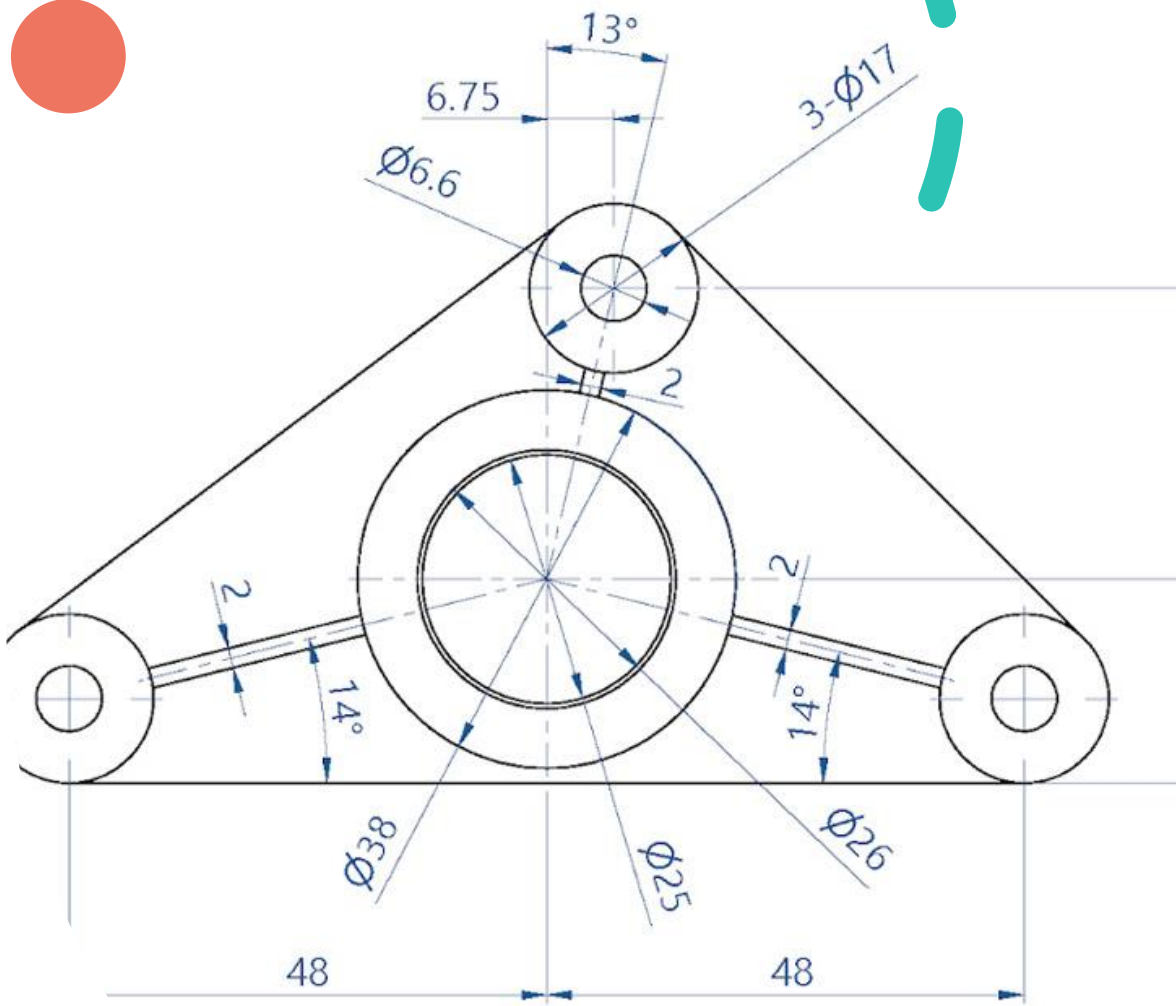
Наприклад: українська мова.

Descriptive rules

UML описовий чи розпорядчий?

UML – це точна мова, тому цілком доцільно очікувати, що вона базується на розпорядчих правилах.

Проте UML часто розглядають як програмний еквівалент креслень, а креслення не базуються на розпорядчих правилах.



Подавлення інформації на діаграмах

Розглядаючи діаграму UML дуже важливо пам’ятати, що головний принцип UML полягає в тому, що вона передає лише ту інформацію, яка потрібна в цей саме час в цих саме умовах.

Отже будь-яка інформація на конкретній діаграмі може бути подавлена.

Подавлення інформації на діаграмах

Подавлення інформації на діаграмі може носити:

- **Локальний характер** – коли, наприклад, не відображаються якісь окремі класи на діаграмі класів
- **Глобальний характер** – коли, наприклад, на тій самій діаграмі класів приховані всі атрибути та методи.

Тому *по конкретній діаграмі ніколи не можна робити висновки про щось, спираючись на його відсутність*.

Навіть якщо метамодель UML має якусь поведінку за замовченням.
Наприклад, для атрибутів класу відсутність квантору видимості за замовченням має на увазі тип Public.


UML
недостатньо?

UML недостатньо?

UML надає широкий спектр діаграм, проте іноді їх буває недостатньо і для задач конкретного проекту використовуються інші діаграми.

Наприклад:

- діаграма потоку екранів, яка надає інформацію про різні екрани інтерфейсу взаємодії користувача з системою та способи переміщення між ними.
- таблиця розв’язків
- mock-up інтерфейсів користувача

UML недостатньо?

Якщо діаграма або якийсь прийом є доцільним для даного проекту – має сенс його використовувати.

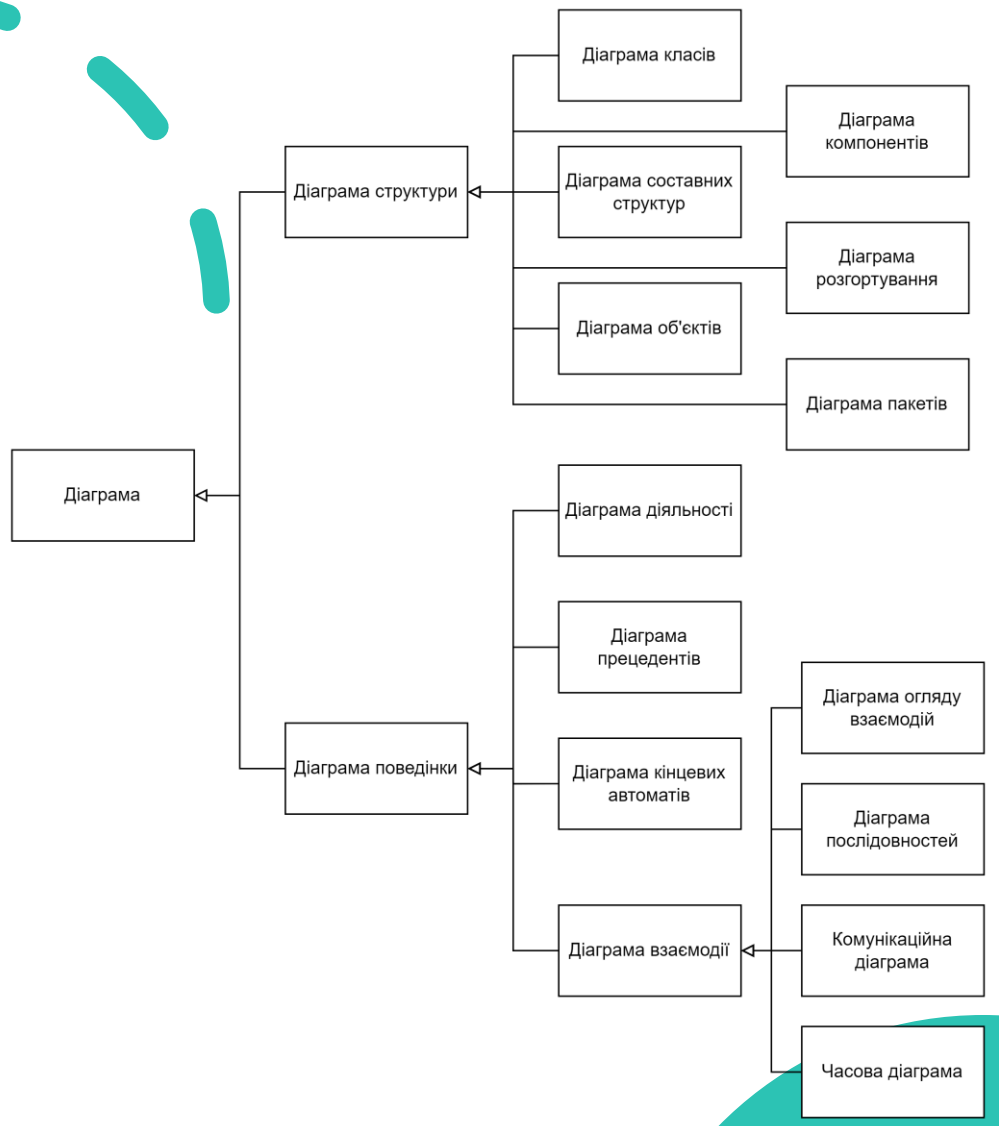
Те саме стосується і самих діаграм UML. Якщо діаграма є недоцільною – потрібно просто забути про неї.

Ніхто,
навіть засновники мови UML
не використовують
всі його можливості.

Типи діаграм UML

Типи діаграм

- **Діаграми структури** - передають систему в статиці
  - Класів
  - Об’єктів
  - Компонентів
  - …
- **Діаграми поведінки** – передають систему в динаміці
  - Прецедентів
  - Діяльності
  - Послідовностей
  - Взаємодії
  - …



Ще раз про підходи до використання UML

Аналіз вимог

Під час **аналізу вимог** необхідно зрозуміти та **дати зрозуміти замовнику**, що вони очікують від системи.

Тому доцільно використовувати наступні прийоми:

- **Діаграми прецедентів**, що описують способи взаємодії з системою
- **Діаграми класів**, що будуються з точки зору концептуальної архітектури системи
- **Діаграми діяльності**, що показують бізнес-процес, способи взаємодії користувачів з системою.
- **Діаграми станів**, що доцільна якщо система має складний життєвий цикл.

Аналіз вимог

Використання діаграм UML на етапі **аналізу вимог** та в режимі ескізу – це **живий рухомий процес**.

Діаграми створюються як під час написання функціонального дизайну та узгодження вимог, так і перед початком процесу розробки з метою чіткого донесення ідеї до розробників системи.

Аналіз вимог

Треба бути готовим в будь-який час відійти від правил UML, якщо це допоможе покращити взаєморозуміння та донести думку до замовника.

Проблема діаграм UML побудованих за всіма правилами полягає в тому, що кінцевий дизайн є незрозумілим спеціалісту в конкретній галузі (стейкхолдеру). Така діаграма гірша за її відсутність.

Проектування

Під час **проектування** можна широко використовувати увесь спектр діаграм, нотації та метамоделі UML.

Тому доцільно використовувати наступні прийоми:

- **Діаграми класів**, що відображають взаємозв’язки між класами, перелік атрибутів та методів класів, тощо
- **Діаграми послідовності** для загальних сценаріїв з метою надати розуміння що саме відбувається в системі
- **Діаграми пакетів**, що демонструють високорівневу організацію продукту
- **Діаграми станів** для класів зі складним життєвим циклом
- **Діаграми розгортання**, що показують фізичну конфігурацію програмного забезпечення.

Проектування

В режимі проектування програмна реалізація будується у відповідності до створених діаграм, тому будь яка зміна в моделі буде вважатись відхиленням, яке призведе до необхідності переробляти вже створений продукт

Проектування

Навіть для досвідченого проектувальника дуже складно з самого початку побудувати модель системи, яка буде повністю вірною та відображати всі необхідні її аспекти

Існує висока вірогідність того, що побудована модель зазнає змін під час проектування

Дякую

Ніколаєнко Дмитро
Володимирович