Лабораторна робота №1

Функціональне призначення проектованої системи

Теоретичні відомості

Як ви розумієте створити хороший додаток це не тільки зробити так, щоб цей додаток працював. Хороша програма - це та програма, яку можна змінювати, доповнювати тестувати, тобто, ця програма має:

1. Ефективну систему. В першу чергу програма, звичайно ж, повинна вирішувати поставлені завдання і добре виконувати свої функції, причому в різних умовах. Сюди можна віднести такі характеристики, як надійність, безпеку, продуктивність, здатність справлятися зі збільшенням навантаження (масштабованість) і тому подібне.

2. Гнучку систему. Будь-який додаток (застосунок) доводиться міняти з часом - змінюються вимоги, додаються нові. Чим швидше і зручніше можна внести зміни в існуючий функціонал, чим менше проблем і помилок це викличе - тим гнучкіша і конкурентоздатніша система. Тому в процесі розробки треба продумати все на рахунок, а раптом, щось доведеться міняти. Запитайте у себе: "А що буде, якщо поточне архітектурне рішення виявиться невірним"?, "Яка кількість коду піддасться при цьому змінам"?. Зміна одного фрагмента системи не повинна впливати на її інші фрагменти. Треба мінімізувати "ціну" помилок.

3. Систему яку можна розширювати. Можливість додавати в систему нові сутності і функції, не порушуючи її основної структури. На початковому етапі в систему має сенс закладати лише основний і найнеобхідніший функціонал. Але при цьому архітектура повинна дозволяти легко нарощувати додатковий функціонал в міру необхідності. Причому так, щоб внесення найбільш вірогідних змін вимагало найменших зусиль.

Вимога, щоб архітектура системи мала гнучкість і розширюваність (тобто була здатна до змін і еволюції) є настільки важливим, що вона навіть сформульована у вигляді окремого принципу - "Принципу відкритості/закритості" (Open - Closed Principle - другий з п'яти принципів SOLID) : Програмні сутності (класи, модулі, функції і тому подібне) мають бути відкритими для розширення, але закритими для модифікації.

Іншими словами: Має бути можливість розширити/змінити поведінку системи без зміни/переписування вже існуючих частин системи.

Це означає, що додаток (застосунок) слід проектувати так, щоб зміна його поведінки і додавання нової функціональності досягалася б за рахунок написання нового коду (розширення), і при цьому не доводилося б міняти вже існуючий код. У такому разі поява нових вимог не спричинить модифікацію існуючої логіки, а зможе бути реалізована передусім за рахунок її розширення.

Масштабованість процесу розробки. Можливість скоротити термін розробки за рахунок додавання до проекту нових людей. Архітектура повинна дозволяти розпаралелювати процес розробки, так щоб безліч людей могли працювати над програмою одночасно.

Тестування. Код, який легко тестувати, міститиме менше помилок і надійніше працюватиме. Але тести не лише покращують якість коду. Багато розробників вважають, «якщо код добре тестується» це автоматично веде до хорошого дизайну. Існує ціла методологія розробки програм на основі тестів, яка так і називається - Розробка через тестування (Test - Driven Development, TDD).

4. Можливість повторного використання. Систему бажано проектувати так, щоб її фрагменти можна було повторно використати в інших системах.

5. Добре читається, структурований і зрозумілий код. Супровід. Над програмою, як правило, працює безліч людей - одні йдуть, приходять нові. Після написання супроводжувати програму теж, як правило, доводиться людям, що не брали участь в її розробці. Тому хороша архітектура повинна давати можливість відносно легко і швидко розібратися в системі новим людям. Проект має бути добре структурований, не містити дублювання, мати добре оформлений код і бажано документацію. І по можливості в системі краще застосовувати стандартні, загальноприйняті рішення звичні для програмістів. Чим екзотичніша система, тим складніше її зрозуміти іншим.

Створення діаграми Варіантів використання

Діаграма Варіантів використання містить деякі варіанти використання системи, деяких дійових осіб і зв'язку між ними. Варіант використання (use case) - це опис функціональності системи на "високому рівні" (як можна використовувати систему). Дійова особа (actor) - це все, що взаємодіє з системою.

Варіанти використання і дійові особи визначають сферу застосування створюваної системи. При цьому варіанти використання описують все те, що відбувається всередині системи, а дійові особи - те, що відбувається з зовні.

Правила розробки діаграми Варіантів використання

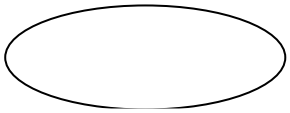
• Не моделюйте зв'язку між діючими особами. За визначенням діючі особи знаходяться поза системою.

• Не поєднуйте стрілкою два варіанти використання. Діаграми даного типу описують тільки, які варіанти використання (функції) доступні системі, а не порядок їх виконання.

• Кожен варіант використання повинен бути ініційований дійовою особою. Це означає, що завжди повинна бути стрілка, що починається з дійової особи і закінчується на варіанті використання.

• За допомогою одного варіанта можна вводити дані в базу, а отримувати їх за допомогою іншого. Для зображення потоку інформації не потрібно малювати стрілками від одного варіанта використання до іншого.

Варіанти використання зазвичай називають дієсловами або дієслівними фразами, описуючи при цьому, що користувач бачить як кінцевий результат процесу. На мові UML варіант використання представляють у вигляді:



Варіант використання

На початку роботи над проектом виникає природне питання: «Як виявити варіанти використання?» Краще всього уважно прочитати документацію замовника. Часто допомагає також розгляд області використання системи на високому рівні і документів концептульного характеру. Врахуйте думку кожної із зацікавлених осіб проекту. Подумайте, чого вони чекають від готового продукту. Кожній зацікавленій особі можна поставити наступні питання:

- Що він хоче робити з системою?
- Чи буде він з її допомогою працювати з інформацією (вводити, отримувати, оновлювати, видаляти)?
- Чи потрібно буде інформувати систему про які-небудь зовнішні події?
- Чи повинна система у свою чергу інформувати користувача про які-небудь зміни або події?

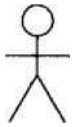
Що б переконатися що виявлені всі варіанти використання слід відповісти на питання:

- Чи присутня кожна функціональна вимога хоч би в одному варіанті використання?

Якщо вимога не знайшла віддзеркалення у варіанті використання, вона не буде реалізована.

- Чи враховано, як з системою працюватиме кожна зацікавлена особа?
- Яку інформацію передаватиме системі кожна зацікавлена особа?
- Яку інформацію отримуватиме від системи кожна зацікавлена особа?
- Чи враховані проблеми, пов'язані з експлуатацією? Хтось повинен буде запускати готову систему і вимикати її.
- Чи враховані всі зовнішні системи, з якими взаємодіятиме дана? Якою інформацією кожна зовнішня система обмінюватиметься з даною?

Дійова особа (actor) — це те, що взаємодіє із створюваною системою. На мові UML дійових осіб представляють у вигляді фігур



Дійова особа

Типи дійових осіб

Дійові особи діляться на три основні типи: користувачі системи, інші системи, що взаємодіють з даною, і час.

Перший тип дійових осіб — це фізичні особи. Вони найбільш типові і є практично в кожній системі. Називаючи дійових осіб, використовуйте їх ролеві імена, а не ті, що відповідають їх посаді. Використовуючи ролі для назви дійових осіб, вам не доведеться оновлювати модель кожного разу при появі нової посади або при зміні розподілу обов'язків між ними.

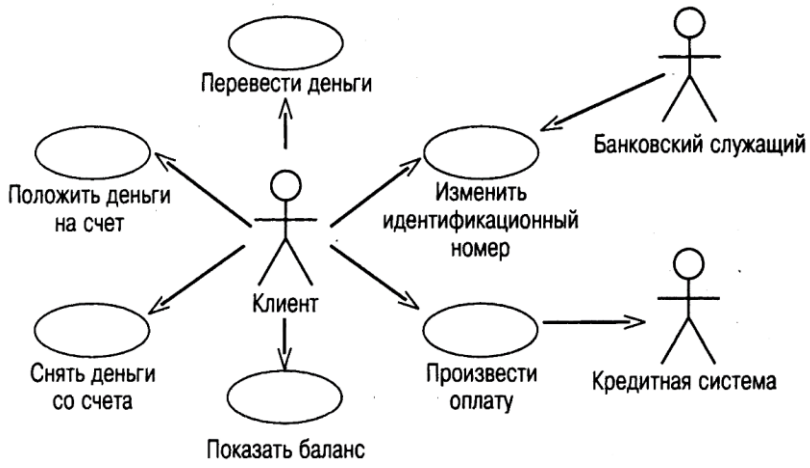
Другим типом дійових осіб є інша система. Роблячи систему дійовою особою, ми передбачаємо, що вона не буде змінюватися взагалі. Потрібно пам'ятати, що дійові особи знаходяться поза сферою дії того, що ми розробляємо, і, отже, не підлягають контролю з нашого боку.

Третій найбільш поширений тип дійової особи — час. Час стає дійовою особою, якщо від нього залежить запуск яких-небудь подій в системі. Оскільки час не підлягає нашому контролю, він є дійовою особою.

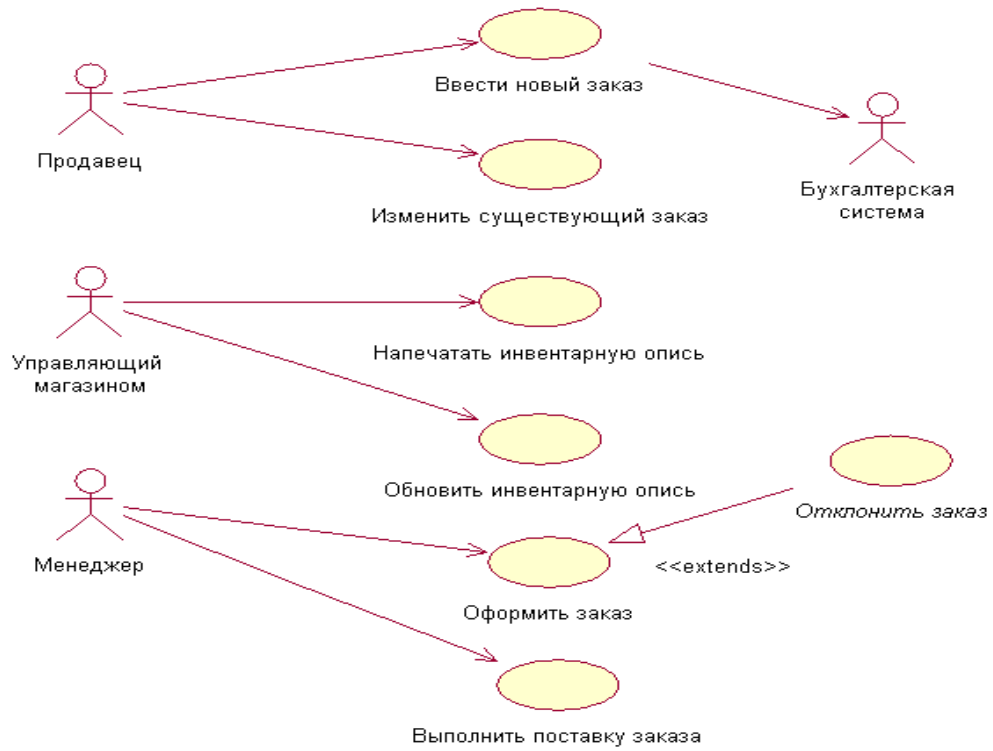
У мові UML для варіантів використання і дійових осіб підтримується декілька типів зв'язків. Це зв'язки комунікації (communication), використання (uses), розширення (extends) і узагальнення дійової особи (actor generalization).

Зв'язки комунікації описують зв'язки між дійовими особами і варіантами використання. Зв'язки використання і розширення відображають зв'язки між варіантами використання, а зв'язки узагальнення дійової особи — між дійовими особами.

Приклад 1.



Приклад 2.



Хто користується діаграмою Варіантів використання:

- за допомогою діаграм Варіантів використання споживачі і менеджери проекту отримають загальне уявлення про систему і зможуть прийняти рішення про сферу її використання.
- за допомогою діаграм Варіантів використання і документації менеджери проекту зможуть розділити проект на окремі керовані завдання.
- з документації по варіантах використання аналітики і споживачі зможуть зрозуміти, що робитиме готова система.
- вивчивши ту ж документацію, можна приступати до написання керівництва для користувачів.

Перелік орієнтовних тем
з дисципліни об'єктно-орієнтоване проектування

1. Додаток для приймальної комісії університету.
2. Додаток для управління БПЛА.
3. Додаток для готелю.
4. Додаток для супермаркету.
5. Додаток для логістики вантажних авіаперевезень.
6. Прикладний додаток для соціальної служби міста (району).
7. Додаток для приватної клініки.
8. Додаток для кіноконцертного комплексу.
9. Додаток для ОСББ.
10. Додаток для автовокзалу.
11. Додаток для інтернет-магазину.
12. Додаток «Розумний дім».
13. Гра.
14. Додаток для ДАІ.
15. Додаток для туристичної агенції.
16. Додаток для агенства нерухомості.
17. Записна книга.
18. Додаток продаж квитків в цирк.
19. Додаток «Розважальні центри великого міста».
20. Додаток «Робота деканату».
21. Додаток «Склад».
22. Додаток «Логістика промислового підприємства».
23. Додаток «Аптека».
24. Додаток «Комунальні платежі».
25. Додаток «Таксі».

Звіт до лабораторної роботи має містити:

1. Опис середовища в яке буде впроваджено ваш додаток (опис предметної області).
2. Глосарій до предметної області.
3. Мета створення додатку.
4. Функції вашого додатку.
5. Словесно-графічний опис роботи вашого додатку.
6. Порівняльний аналіз інструментальних засобів для побудови UML діаграм.
7. Діаграму «Варіантів використання».

Питання для самоконтролю:

1. Яке функціональне призначення вашого додатку.
2. Що можна дізнатися з діаграми Варіантів використання.
3. Які графічні елементи використовуються на діаграмі Варіантів використання.
4. Правила розробки діаграми.
5. Типи діючих осіб.
6. Як визначитись з Варіантами використання.
7. Для кого корисна діаграма Варіантів використання.