МІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ
Кафедра комп'ютерних наук

04-05-11

МЕТОДИЧНІ ВКАЗІВКИ

до виконання
КУРСОВОЇ РОБОТИ З ДИСЦИПЛІНИ
„ ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ"

для здобувачів вищої освіти першого (бакалаврського) рівня
зі спеціальності 122 "Комп'ютерні науки"
денної та заочної форм навчання

Рекомендовано науково-методичною комісією
за спеціальністю 122 "Комп'ютерні науки"
Протокол № 4 від 3 травня 2018 р.

РІВНЕ–2018

Методичні вказівки до виконання курсової роботи з дисципліни „Проектування інформаційних систем” для здобувачів вищої освіти першого (бакалаврського) рівня зі спеціальності 122 “Комп’ютерні науки” денної та заочної форм навчання / Ю.Й. Тулашвілі. - Рівне: НУВГП, 2018. – 19 с.

Укладач: Ю.Й. Тулашвілі, доктор педагогічних наук, професор

Відповідальний за випуск: Ю.Й. Тулашвілі, доктор педагогічних наук, професор, завідувач кафедри комп’ютерних наук

Вступ

Тенденції розвитку сучасних інформаційних технологій призводять до постійного зростання складності інформаційних систем (ІС).

Для успішної реалізації проекту об'єкт проектування ІС повинен бути адекватно описаний та повинні бути побудовані повні і несуперечливі функціональні й інформаційні моделі ІС. Крім того, у процесі створення і функціонування ІС інформаційні потреби користувачів можуть змінюватися чи уточнюватися, що ускладнює розробку і супровід таких систем.

Приблизно чверть століття тому швидко зростаючий обсяг і складність систем вступили в явне протиріччя з відсутністю єдиного підходу до їх аналізу і проектування, недостатнім рівнем участі користувача в процесі розробки, неузгодженістю різних етапів розробки. Помилок було багато й обходилися вони дуже дорого. Модульне і структурне програмування, логічне моделювання структур баз даних, схеми потоків даних і проектування "зверху вниз" при всій початковій ейфорії залишилися внутрішньою справою розроблювачів. Проблема була глибша - потрібно було якось об'єднати замовників, розробників, програмістів, користувачів - причому в умовах мінливої ситуації. Для того, щоб про щось домовитися, потрібна якась спільна мова. Природна мова в силу малої наочності, неоднозначності, надмірності і багатослівності для цієї ролі не пасувала, і, зрештою, почалися спроби створення чіткої графічної мови.

Перераховані фактори сприяли появі програмно-технологічних засобів спеціального класу - CASE-засобів, що реалізують CASE-технологію створення і супроводу ІС. Термін CASE (Computer Aided Software Engineering) використовується в даний час у дуже широкому сенсі. Первісне значення терміну CASE, обмежене питаннями автоматизації розробки тільки програмного забезпечення (ПЗ), у даний час набуло нового сенсу, що охоплює процес розробки складних ІС у цілому. Нині термін CASE-засіб - це програмні засоби, що підтримують процеси створення і супроводу ІС, включаючи аналіз і формулювання вимог, проектування прикладного ПЗ (додатків) і баз даних, генерацію коду, тестування, документування, забезпечення якості, конфігураційне управління і управління проектом, а також інші процеси. CASE-засоби разом із системним ПЗ і технічними засобами утворять повне середовище розробки ІС.

CASE-технологія - методологія проектування ІС, а також набір інструментальних засобів, що дозволяють у наочній (візуальній) формі моделювати предметну область, аналізувати цю модель на всіх етапах розробки і супроводу ІС та розробляти додатки відповідно до інформаційних потреб

користувачів.

Проектування програмного забезпечення за допомогою CASE-систем має кілька етапів. Початковий етап - попереднє вивчення проблеми. Результат подається у вигляді вихідної діаграми потоків даних і погоджують із замовником. На наступному етапі виконують деталізацію обмежень і функцій програмної системи, отриману логічну модель знову погоджують із замовником. Далі розробляють фізичну модель, тобто визначають модульну структуру програми, виконують інфологічне проектування бази даних, деталізують схеми програмної системи і її модулів.

Виконання курсової роботи з навчальної дисципліни „Проектування інформаційних систем" формує у студентів уміння системно мислити, розробляти бізнес-логіку ІС, використовувати CASE-засоби.

1. Мета та завдання курсової роботи

Курсова робота - один з важливих етапів навчання здобувачів вищої освіти з дисципліни „Проектування інформаційних систем".

Мета виконання курсової роботи - закріплення студентами теоретичних навичок та розширення практичних навичок проектування та створення програмного продукту; розвиток творчих навичок при самостійному програмуванні та самостійній роботі з джерелами технічної інформації; підготовка до виконання складних курсових робіт і курсових проектів, а також підготовка до дипломного проектування.

Завдання курсової роботи - навчити студентів під час виконання курсової роботи:

- описувати прикладну галузь проектного завдання;
- будувати моделі ІС;
- описувати сутності та поля, що використовуються при проектування логічної та фізичної моделей;
- створювати проектну документацію та писати звіти.

Результатом виконання курсової роботи є створення функціональної моделі відповідної предметної області та створення діаграм ІС з використанням CASE-засобів, а також створення та друк за результатами роботи вихідних документів як мовою програмування, так і у форматі Word.

2. Методичні вказівки щодо виконання курсової роботи

На початку курсової роботи студент повинен:

- самостійно чи за допомогою викладача обрати тему роботи;

- проаналізувати предметну область застосування;
- визначити, які функції повинні реалізуватися в ІС, що розробляється;
- виділити вхідні та вихідні параметри задачі;
- ознайомитись з формами подання результатів проектної роботи, а також звітів і назв вихідних документів.

На основі аналізу здійснюється постановка та алгоритмізація задачі, розробка функціональної моделі у логічному вигляді, перетворення у фізичний вигляд та об’єднання даних моделей ІС.

За результатами курсової роботи студенти складають звіт, який містить опис всіх етапів роботи і додатки, що включають основні форми програмного продукту та згенеровані шаблони текстів програми, обраною мовою програмування.

Захист курсової роботи супроводжується демонстрацією на комп'ютері та поясненням усіх діаграм, розкриттям теоретичних завдань.

3. Тематика курсових робіт

Тематика курсової роботи відповідає програмі курсу „Проектування інформаційних систем” і пов'язується з спрямованістю цільової підготовки студентів. Тематика складається таким чином, щоб студент брав участь у створенні проекту ІС, яка за своїм обсягом і змістом відповідає реальним розробкам, що виконуються в межах ІТ студії кафедри. Перелік тем курсових робіт оновлюється. Тематика курсових робіт не обмежується будь-якими рамками.

Індивідуальне завдання складається з назви задачі. Студент повинен самостійно проаналізувати, які вхідні дані повинні використовуватися і які функції будуть реалізовані в програмному проекті з заданої предметної області, а також спроектувати діаграми ІС, скласти звіти та проектну документацію.

3.1 Орієнтовний перелік тем курсової роботи

1. ІС супроводу туристичного маршруту в місті.
2. ІС знаходження закладів громадського користування в місті.
3. ІС фотофіксування місцевості під час туристичного подорожі.
4. Автоматизована ІС аналізу цін та керування замовленнями фармацевтичної продукції.
5. Автоматизована система обліку залізничних вагонів ВАТ «Азот».
6. Автоматизована ІС розкладу руху поїздів.
7. ІС кадрового забезпечення
8. ІС нарахування заробітної плати

9. Автоматизована ІС контролю виконання робіт на СТО

10. Автоматизована ІС контролю за витратами сімейного бюджету.

11. Автоматизована ІС розрахунків кошторису на ремонт житлових приміщень

12. Автоматизована ІС складу побутових приладів

13. Автоматизована ІС реєстратури поліклініки

14. Автоматизована ІС ведення ресторанного бізнесу

15. Автоматизована ІС ведення готельного бізнесу

16. АІС розрахунку бізнес планів

17. АІС рієлторської контори

18. АІС адвокатської контори

19. АІС Нотаріальної контори.

20. Автоматизована ІС Бібліотечного каталогу

21. Автоматизована ІС відділу кадрів університету

22. Автоматизована ІС облік матеріальних цінностей (склад);

23. АІС довідкова служба поліклініки;

24. АІС довідкова служба аптек

25. АІС оплата послуг телефонної мережі

26. ІС клінічні лікарні міста

27. ІС результати екзаменаційних сесій студентів за весь період навчання

28. ІС довідкова служба підприємства (заводу)

29. АІС робота магазину (торгової мережі) міста

30. АІС облік платників податків

Питання до теоретичної частини курсової роботи вибираються за номером в списку групи N.

3.2 Питання для теоретичної частини курсової роботи

1. Поняття Case-засобів і їх призначення.
2. Суб’єкт моделювання в IDEF0. Принцип обмеження суб’єкта.
3. Об’єкти і функції в IDEF0.
4. Правила представлення функціональних блоків на IDEF0- діаграмі.
5. Призначення сторін функціональних блоків на IDEF0- діаграмі.
6. Принцип домінування і його уявлення на IDEF0- діаграмі.
7. Призначення дуг на IDEF0- діаграмі.
8. Опис дуг на IDEF0- діаграмі.
9. Види стосунків між об’єктами і дугами на IDEF0- діаграмі.
10. Типи взаємозв’язків між блоками на IDEF0- діаграмі.
11. Розгалуження дуг і правила їх позначення на IDEF0- діаграмі.
12. Сімейство методологій IDEF, їх призначення.

13. Злиття дуг і правила їх позначення на IDEF0- діаграмі.
14. Глосарії і Словник даних.
15. С-номери. Призначення і правила запису.
16. Бланк реєстру С-номерів IDEF0.
17. Напрями SADT-моделювання.
18. Етапи життєвого циклу програмних засобів, для яких найбільш ефективне використання методології SADT.
19. Переваги методології SADT.
20. Поняття системи і моделі в IDEF0.
21. Формальне визначення моделі в IDEF0.
22. Мета моделі в IDEF0.
23. "Точка зору" моделі в IDEF0.
24. Асоціативні сутності
25. Виділення сутностей
26. Визначення первинного ключа
27. Визначення типів сутностей
28. Відображення логічного та фізичного рівня моделі даних у ERwin
29. Зв’язки (relationships) у ERwin
30. Ідентифікація сутностей. Сутності в ERwin
31. Інструменти для створення моделі в ERwin
32. Інтерфейси до СУБД
33. Інформаційне моделювання
34. Кодові сутності
35. Місце ERwin в інформаційному моделюванні
36. Моделювання в ERwin
37. Підтримка засобів 4GL
38. Поняття діаграми-нащадка, батьківського блоку
39. Поняття сутностей
40. Правила та початкові значення
41. Синхронізація з базою даних
42. Стрижневі сутності
43. Структурна сутність
44. Сутності та атрибути в реляційній моделі
45. Характеристичні сутності

Питання до теоретичної частини курсової роботи студенти вибирають за номером в списку групи N. Перше питання відповідає N, друге N+20 Наприклад, якщо номер по списку 3, тоді студент обирає питання номер 3 та 23. Курсова робота виконується відповідно до графіка (2 сторінка технічного завдання).

4. Зміст курсової роботи

Курсова робота оформляється у формі «ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ», що містить:

Вступ

1 Аналіз предметної області.

2 Теоретичні завдання роботи.

3 Розробка специфікації СВ згідно зі стандартом RUP.

4 Побудова діаграм.

4.1 Вибір та опис CASE-засобу створення діаграм.

4. 2 Створення Use Case Diagram.

4. 3 Створення Activity Diagram чи Sequence Diagram.

4. 4 Створення Class Diagram.

4. 5 Створення Object Diagram чи Component diagram.

5 Генерація коду.

Висновок

Список використаної літератури

Додатки:

Додаток А Результати роботи з розробки Use Case Diagram у сервісі creately.com

Додаток Б Результати роботи з розробки Activity Diagram у сервісі creately.com

Додаток В Результати роботи з розробки Class Diagram у сервісі creately.com

5. Оформлення курсової роботи

Курсова робота оформляється кожним студентом у вигляді окремого звіту (пояснювальної записки).

Робота має бути оформлена в текстовому редакторі WORD, надрукована на аркушах формату А4.

Титульна аркуш звіту повинен бути підписаний студентом та керівником курсової роботи.

За титульним аркушем звіту розміщується аркуш технічного завдання, заповнений і завірений підписом викладача. Далі розміщується зміст (не менш ніж дворівневий), створений засобами редактора Word, а потім розміщують текстову частину курсової роботи.

Розміри полів на сторінці: верхнє - 20 мм, нижнє - 20 мм, ліве - 30 мм,

праве - 10 мм.

Для тексту звіту використовувати параметри: гарнітура - Dmes New Roman; кегль (розмір) 14; міжрядковий інтервал - 1,5. Текст вирівняти з обох боків.

Робота в цілому повинна мати колонтитули (з прізвищем студента та номером групи - верхній колонтитул) та нумерацію сторінок, створену в редакторі тексту (нижній колонтитул).

Нумерація сторінок у пояснювальній записці у нижньому правому кутку.

У висновку роботи зробити аналіз розробленої системи. Вказати на особливості розробки та застосовані підходи при розробці функціональної моделі.

Список інформаційних джерел (не менше 10 джерел) має містити підручники (посібники) з розробки функціональних моделей та сайти, з яких бралася інформація з назвою статті, розміщеної на сайті.

Рисунки, креслення, таблиці, графіки, розташовані на окремих аркушах записки, включають у загальну нумерацію сторінок. Ілюстрації позначаються словом "Рис." і нумеруються послідовно в межах розділу (див. Додатки оформлення роботи).

Кожен розділ звіту треба починати з нової сторінки. Найменування розділу записується у вигляді заголовка великими літерами. Крапку наприкінці заголовка не ставлять. Переноси слів у заголовках не допускаються. Якщо заголовок складається з кількох речень, то їх розділяють крапкою. Підкреслювати заголовки не треба. Пункти, підпункти починають з нового рядка і з прописної букви.

Зміст звіту включає розділи, їхні порядкові номери позначаються цифрами без крапки: 1, 2, 3, і т. д. (не нумерують розділи: зміст, список прийнятих скорочень, введення, висновок, список літератури, додатки). Перелік, що містяться в тексті пункту, позначають цифрами з дужкою, наприклад: 1), 2), 3) тощо.

Номер ілюстрації, формули, таблиці повинний складатися з номера розділу і відділеного крапкою номера в розділі, наприклад, друга формула (ілюстрація, таблиця) четвертого розділу має номер 4.2: Рис. 4.2, Таблиця 4.2.

Перелік нестандартних скорочень, позначень, символів, одиниць і термінів оформляється на окремій сторінці та міститься між змістом і вступом.

У тексті записки потрібно давати посилання на використану літературу згідно стандартам: вони містять номер літературного джерела і, у необхідних випадках, номер сторінки, укладені в квадратні дужки. Наприклад, "[8]", "[6, с. 5]". Якщо посилання відбувається на дані, що отримані раніше у своїй роботі, то вони включаються у круглі дужки і містять скорочене слово "дивися"

і посилальні сторінку, малюнок тощо, наприклад: "(див. с. 21)".

6. Приклад виконання розділу “4. Побудова діаграм”

4.1 Вибір та опис CASE-засобу створення діаграм

Потужний поштовх CASE-засоби отримали в пору впровадження об'єктно-орієнтованої технології розробки програмного забезпечення. Старі технології розробки програм «зверху вниз» вже не могли впоратися з трудомісткими програмними комплексами, що дедалі ускладнювались.

CASE-система, як система проектування програмного забезпечення, містить компоненти для розробки структурних схем алгоритмів і "екранів" для взаємодії з користувачем в інтерактивних процедурах, засоби для інфологічного проектування баз даних, налагодження програм, документування, збереження "історії" проектування тощо.

На ринку програмних продуктів є багато CASE-засобів для концептуального проектування ІС. Найчастіше в них підтримується методологія IDEF. Широко відомі такі програми: BPwin, ERwin фірми Platinum Technology, Design/IDEF фірми Meta Software та інші.

BPwin (Business Processing) призначена для розробки функціональних моделей за методикою IDEF0.

ERwin призначена для розробки інформаційних моделей за методикою IDEF1X. В ній є засоби, що забезпечують інтерфейс із серверами баз даних (від користувача приховане спілкування на SQL-мові), переклад графічних зображень ER-діаграм в SQL-форми або у формати інших популярних систем керування базами даних, передбачено інтерактивні процедури для зв'язування дуг IDEF0 із атрибутами IDEF1X, тобто для встановлення зв'язку між BPwin і ERwin.

Моделювання відіграє велику роль в успішності розробки ІС. Використання цих інструментів (BPwin і ERwin) спільно допомагає правильно оцінити завдання, що стоять, запропонувати адекватні рішення (аналіз бізнес-процесів, BPwin) і розробити центральну частку будь-якої ІС - бази даних - з використанням інформації, отриманої під час обстеження підприємства (моделювання бази даних, ERwin). Використання цих інструментів дає можливість отримати набір повністю документованих і узгоджених моделей, що в значній мірі полегшить підтримку створених систем в майбутньому, а також може бути повторно використано при розробці інших систем.

UML є відкритим засобом проектування ІС і володіє властивостями розширення базового ядра. На UML можна змістовно описувати класи, об'єкти і

компоненти в різних предметних областях, що часто сильно відрізняються одна від одної.

Пакет Rational Rose підтримує не тільки UML, але й інші нотації створення діаграм, такі як OMT або Booch. Сьогодні Rational Rose лідирує серед інших CASE-засобів. Цей пакет дозволяє створювати складні програмні системи на всіх стадіях проектних робіт: від задуму до створення вихідного коду, що приваблює не тільки проектувальників систем, але й програмістів.

Інструменти компанії Rational Software якнайкраще підходять для об'єктно-орієнтованої розробки програм. У поєднанні із засобами документування (Rational SoDA) Rational Rose може давати повне уявлення про проект. Повністю інтегруючись з Microsoft Visual Studio, цей пакет дає можливість отримувати вихідний код взаємодіючих класів і будувати візуальні моделі по вже написаному вихідному коду (реінжтніринг). Можливість інтеграції із засобами управління вимогами (Requisite Pro), із засобами тестування (SQA Suite, Performance Studio), із засобами конфігураційного управління (ClearCase, PVCS) піднімає процес ведення програмного проекту па абсолютно новий рівень. Відкрита архітектура Rational Rose дозволяє включати до нього підтримку мов програмування, які не передбачені стандартною поставкою, наприклад, мови Assembler, для чого достатньо написати лише власний модуль.

Формулювання задачі автоматизації. Для успішної реалізації проекту об'єкт проектування – автоматизована система (АС) агропромислового комплексу, повинна бути перш за все адекватно описана: повинні бути побудовані повні і несуперечливі функціональні моделі АС.

Проектування АС агропромислового комплексу – це логічно складна, трудомістка і тривала за часом робота, що вимагає високої кваліфікації що в ній фахівців. У процесі створення і функціонування АС потреби користувачів можуть змінюватися і або уточнюватися, що може ще більше ускладнювати процес проектування таких систем.

Для створення АС агропромислового комплексу будемо використовувати описаний нижче порядок роботи.

Насамперед, зробимо аналіз списку операцій, які виконуватиме система, і визначимо список об'єктів системи, які дані функції виконують. Таким чином, визначимо вимоги до системи і межі предметної області. Для цього будемо використовувати діаграми Use case.

Після визначення в системі необхідних класів визначимо поведінку конкретних класів за допомогою діаграм State diagram (діаграми станів) і Activity diagram (діаграми активності). Подальша деталізація взаємодії класів буде проводитися за допомогою Sequence diagram (діаграми послідовностей

дій).

Список класів, які повинні бути присутніми в системі, поки були без конкретної деталізації та докладного опису дій. Для деталізації класів використовувати діаграму класів (Class diagram). Тому, на підставі визначених класам дій створимо ієрархію класів АС за допомогою діаграми класів (Class diagram) і визначимо компоненти, в які ці класи необхідно включити за допомогою діаграми компонентів (Component diagram).

Необхідно розуміти, що розробка АС – це ітераційний процес. Не можливо за один раз створити повний проект АС. Потрібно декілька разів повертатися до вже створених діаграм і вносити до них зміни.

4.2 Побудова діаграми прецедентів

Починаємо роботу зі створення діаграми варіантів використання. На ній відображаються усі головні актори та їх взаємодія з головними елементами системи та їх функціями (відповідно до розділу 3 Розробка специфікації СВ згідно стандарту RUP). Прецедент (Варіант використання) – це опис на «високому рівні» фрагмента функціональності, яку забезпечує система. Інакше кажучи, варіант використання ілюструє, як можна використовувати систему. Узагальнення класу – це відношення, направлене від класу до його базового класу, тобто відношення, зворотнє успадкуванню. Узагальнення прецедентів (generalization) аналогічно узагальненню класів, тобто це відношення дочірнього прецеденту до його батьківського (базового) прецеденту. Позначення відношення узагальнення відображено на рис. 4. 1

Рис. 4. 1. Відношення узагальнення прецедентів

У нашій системі ми виокремили чотири головних актори:

Головний агроном – це фізична особа, та вона є одним з головних акторів адже взаємодіє з автоматизованою системою самостійно. Основними прецедентами даного актора є підтримання роботи системи та виявлення зауважень.

Плановий відділ – це колектив, що розраховує планову собівартість виробництва видів продукції рослинництва, а саме, прямі витрати безпосередньо на технологічні операції по місяцях. Основними прецедентами

даного актора є розрахунок планової собівартості.

Робітники – це фізичні особи, котрі відповідно до власної кваліфікації мають власні прецеденти: виконання робіт, взаємодія з головним агрономом.

Актори знаходяться поза сферою дії того, що ми розробляємо, і, отже, не підлягають контролю з нашого боку. Між актором та елементами роботи проводяться зв’язки. Якщо стрілка суцільна то даний актор взаємодіє з прецедентом напряму. А якщо ми маємо пунктирну стрілку то це означає, що актор взаємодіє з прецедентом не напряму. Він взаємодіє з прецедентом через іншого актора або через інший прецедент. Напрямок стрілки показує, хто ініціює комунікацію.

Поведінка системи, що розробляється описується за допомогою функціональної моделі, яка відображає системні прецеденти (use case), системне оточення (actors) і зв'язки між прецедентами та акторами. Основне завдання моделі прецедентів – представляти собою єдиний засіб, що дає можливість замовнику, кінцевому користувачеві і розробнику спільно обговорювати функціональність і поведінку системи. Отже продемонструємо створену діаграму:

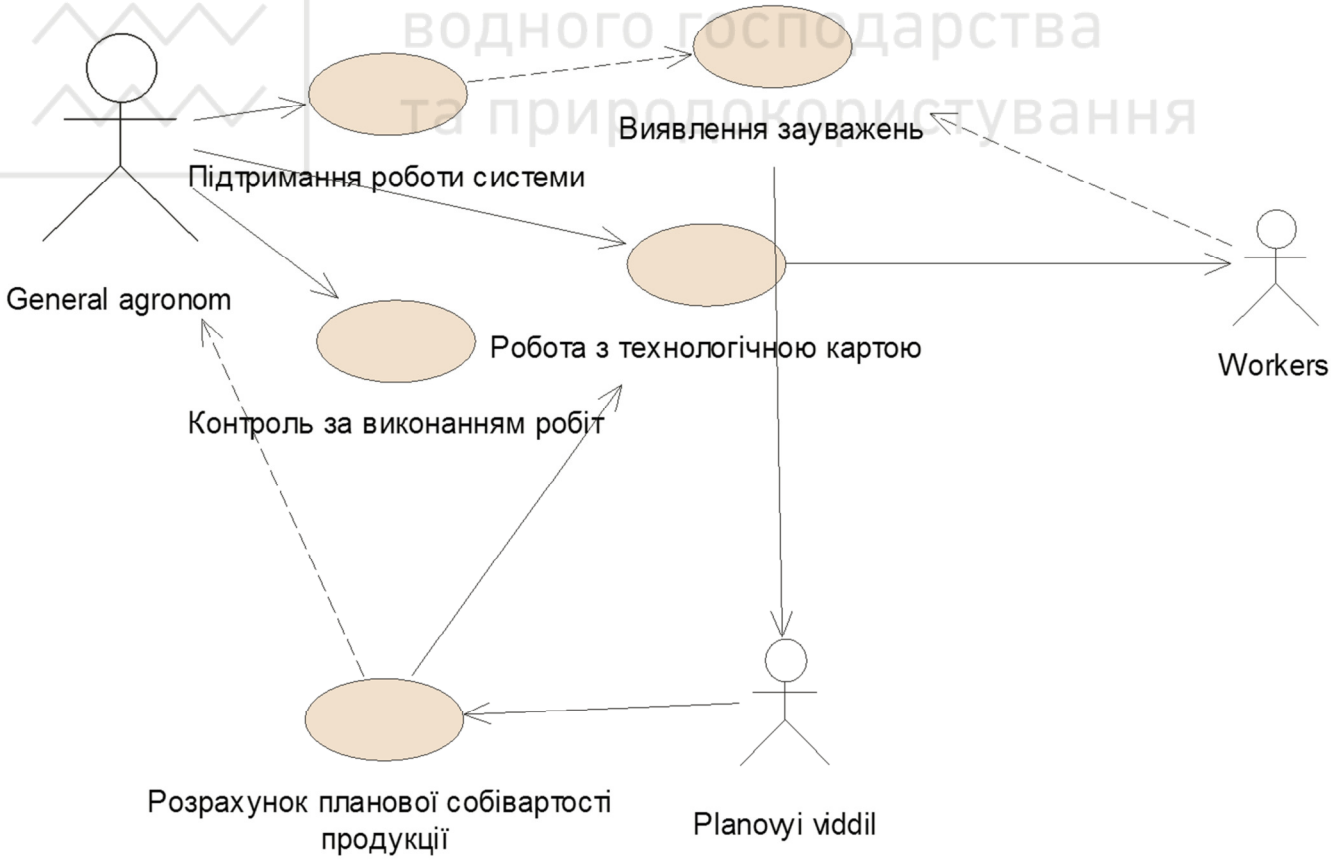

Рис. 4. 2. Діаграма прецедентів

4.3 Створення діаграми послідовності

Діаграма послідовності відображає потік подій, що відбуваються в рамках варіанту використання. Діаграма послідовності – це впорядкована за часом діаграма Взаємодії і читати її слід зверху вниз. Діаграма послідовності є іншою формою діаграми взаємодії і оперує об’єктами і повідомленнями.

Побудова діаграми послідовності зводиться до додавання і редагування властивостей окремих повідомлень і об’єктів. Доступ до вікон специфікації можна організувати також командою меню Browse, Specification. При додаванні повідомлення, вони одержують свій номер в загальній послідовності повідомлень.

Діаграми послідовності впорядковані за часом. Вони корисні для того, хто хоче зрозуміти логічну послідовність подій в сценарії. Хоча інформація про послідовність входить і в Кооперативні діаграми, вона краще сприймається на діаграмі Послідовності.

З діаграми послідовностей маємо послідовність дій, які має Головний агроном для планування посівної. Отже, спершу Головний агроном планує посівну, він вивчає технологічну карту і прейскурант оплати працівників, яких він залучатиме в ході посівної, пізніше при взаємодії з плановим відділом розраховується приблизна собівартість продукції, і якщо агронома влаштує прибуток то він відведе певну площу під посів і запустить процес посівної

Основний потік – яку саме роботу виконує варіант використання. В припущенні, що умови ідеальні і все що відбувається розвивається в позитивному напрямку.

Альтернативний потік. Уявіть всі можливі ситуації коли Основний потік може відхилятись з свого основного руслу. Почати опис альтернативних потоків добре спочатку описавши всі ситуації в які може попасти варіант використання після вдалого чи невдалого виконання, а потім описати в якому місці починається відділення від основного потоку та які дії повинні виконуватись.

Діаграма послідовності дій відображає взаємодію об'єктів, впорядкованих за часом. На ній показані об'єкти і класи, використовувані в сценарії, і послідовність повідомлень, якими обмінюються об'єкти, для виконання сценарію. Діаграми послідовності дій зазвичай відповідають реалізаціям прецедентів в логічному представленні системи.

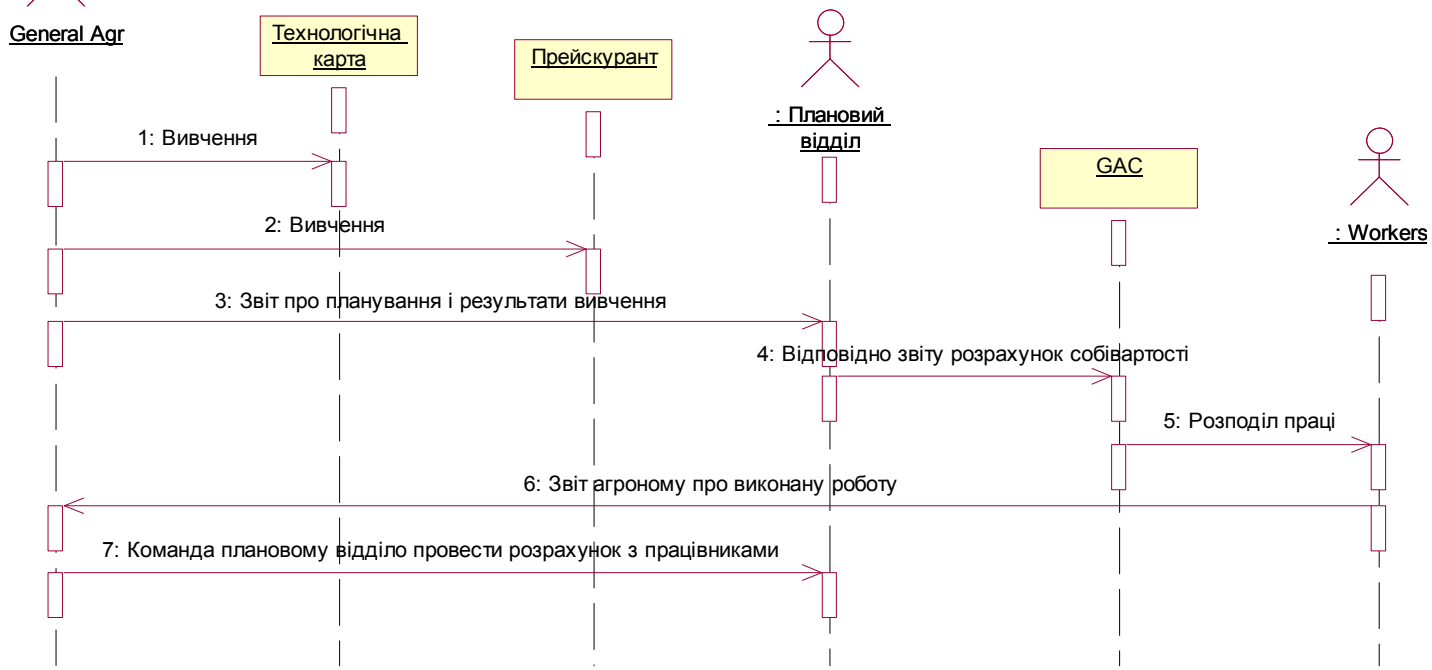

Рис. 4. 3. Діаграма послідовності

4.4 Побудова діаграми класів

Діаграма класів є основним логічним представленням моделі і містить детальну інформацію про внутрішню будову об'єктно-орієнтованої програмної системи або, використовуючи сучасну термінологію, про архітектуру програмної системи. Діаграми класів називають “статичними діаграмами”, оскільки на них показано класи разом з методами і атрибутами, а також статичний взаємозв’язок між ними: те, яким класам “відомо” про існування яких класів, і те, які класи “є частиною” інших класів, – але не показано методи, які при цьому викликаються.

В діаграмі класів повністю описуються класи, їх властивості – атрибути і операції. На підставі діаграми класів Rational Rose виконує такі функції: показати клас, описати його, показати опції, атрибути. Атрибути – це елементи з якими працює даний клас, чию інформацію він обробляє і над чим працює. У операціях класу вказуються дії для яких призначений даний клас, те що вини повинні виконувати.

У UML атрибути показуються щонайменше назвою, також може бути показано їх тип, початкове значення і інші властивості. Крім того, атрибути може бути показано з областю видимості атрибута:

+ відповідає публічним (public) атрибутам

відповідає захищеним (protected) атрибутам

- відповідає приватним (private) атрибутам

Операції (методи) також показуються принаймні назвою, крім того, може бути показано їх параметри і типи значень, які буде повернуто. Операції, як і атрибути, може бути показано з областю видимості:

+ відповідає публічним (public) операціям

відповідає захищеним (protected) операціям

- відповідає приватним (private) операціям

Для даної системи визначили три класи: Планування, Працівники та General Agronom Control.

Вони мають свої атрибути. Клас Технологічна карта працює з плановим відділом, Система GAC (General Agronom Control) – з технологічною картою та працівниками, а зі змінами по проекту системи. До операцій Планування було віднесено: перевірка зауважень, фіксація зауважень, виправлення зауважень, розрахунок планової собівартості продукції. До операцій класу Система GAC відносяться: Користування системою та виявлення зауважень, контроль працівників. До операцій класу Працівники відносяться: виконання робіт відповідно до технологічної карти. Отже продемонструємо створену діаграму класів.

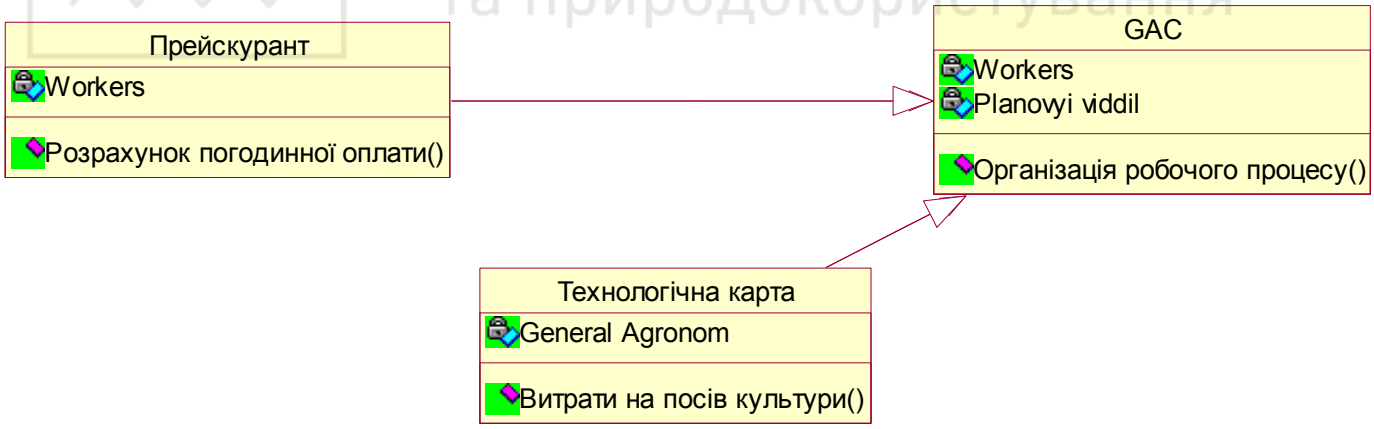

Рис. 4.4. Діаграма класів

4.5 Побудова кооперативної діаграми

Діаграма кооперації є різновидом діаграми взаємодії, і в контексті мови UML описує динамічний аспект взаємодії об'єктів при реалізації окремих варіантів використання. Подібно діаграм послідовності, кооперативні діаграми відображають потік подій через конкретний сценарій варіанту використання. Діаграми послідовності впорядковані за часом, а кооперативні діаграми

загострюють увагу на зв'язках між об'єктами. Також на даних діаграмах показані почергово варіанти використання необхідних прецедентів, що також є немало важливим у розумінні діаграми. Адже за допомогою вказаних зв’язків ми можемо простежити почергово лінію роботи системи. Кооперативна діаграма включає акторів та їх прецеденти які ми вже визначали за допомогою діаграми варіантів використання.

З діаграми кооперацій легше зрозуміти зв'язки між об'єктами, однак важче усвідомити послідовність подій. З цієї причини часто для будь-якого сценарію створюють діаграми обох типів. Хоча вони служать для одній і тієї ж мети і містять одну й ту ж інформацію, але представляють її з різних точок зору.

За допомогою пункту меню Browse\Create collaboration diagram створимо діаграму кооперації для нашого прецеденту. Отже продемонструємо створену діаграму:

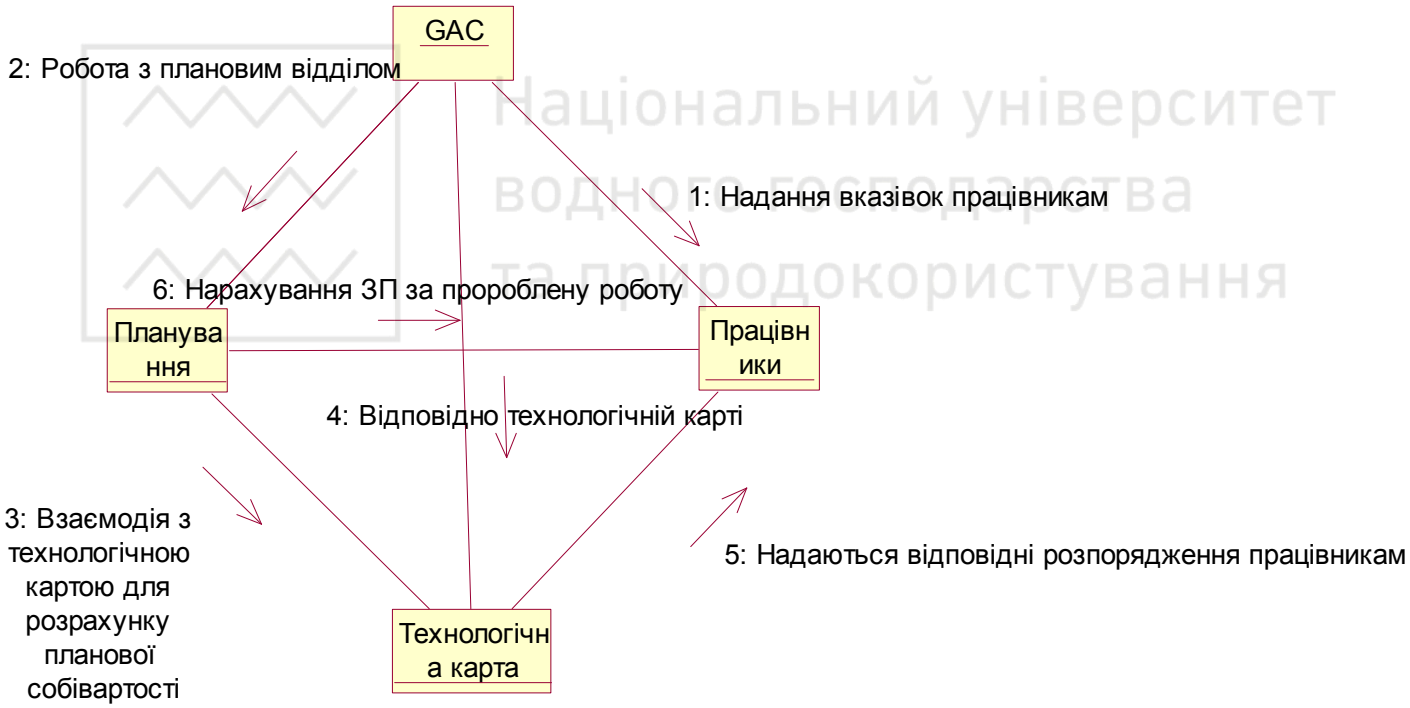

Рис. 4.5. Діаграма кооперації

7. Організація виконання та захисту роботи

Завдання на курсову роботу студенту видається на початку семестру. Одночасно студент отримує лист технічного завдання та заповнює календарний план виконання роботи, який він повинен чітко виконувати.

Курсова робота виконується відповідно до графіка. Контроль виконання студентом етапів курсової роботи здійснюється керівником і результати

виконання етапу фіксуються у календарному плані студента, де фіксуються такі параметри як дата, оцінка, без або з зауваженнями тощо.

Консультації проводяться згідно з розкладом, погодженим зі старостами групи. Керівник спрямовує зусилля студента, контролює обсяг виконаної роботи, коригує завдання.

Курсову роботу студенти здають в два етапи:

- демонстрація експлуатаційних можливостей програмного проекту (виконується розробником у присутності членів комісії). Виявлені недоліки фіксуються у протоколі;
- захист курсової роботи (у вигляді доповіді основних положень курсової роботи).

Завершену курсову роботу студент подає на перевірку керівникові не пізніше за 7 днів до завершення навчального графіку.

За один-два дні керівник допускає роботу до захисту або повертає її для доопрацювання. Після виправлення помилок або доопрацювання пояснювальна записка повторно перевіряється керівником і, якщо зауважень немає, курсова робота допускається до захисту.

Під час захисту студент робить доповідь (5-7 хвилин) про виконану роботу і відповідає на запитання керівника, членів комісії та присутніх. Комісія після обговорення приймає рішення про рівень роботи і оголошує оцінку. Робота здається комісії, після чого керівник записує оцінку в заліковій відомості, заліковій книжці (якщо оцінка задовільна) і на записці. Члени комісії ставлять підписи на записці і в заліковій відомості. Захист на цьому завершений.

Основні критерії оцінки курсової роботи:

- рівень якості поданої програмної розробки (повнота реалізованих функцій, рівень інтерфейсу, наявність можливостей настроювання, стійкість та надійність функціонування, наявність засобів допомоги тощо);

- практична цінність проектних рішень (відповідність реальним умовам об'єкта, універсальність та оригінальність прийнятих рішень):

- відповідність оформлення курсової роботи встановленим вимогам, дотримання встановлених стандартів;

- своєчасність виконання графіка робіт при проектуванні та поданні курсової роботи.

Список використаної літератури:

1. Орлов С. А. Технологии разработки программного обеспечения. Современный курс по программной инженерии: учебник для вузов / С. А. Орлов, Б. Я. Цилькер. − 4-е изд. − Санкт-Петербург : Питер, 2012. − 608 с.

2. Матвєєва Л.Є. Процес розробки програмного забезпечення. Від теорії до практики / Л.Є. Матвєєва, В.А. Волков. – К. : ТОВ «Інформаційні програмні системи», 2008. – 117 с.

3. Рудаков А.В. Технология разработки программных продуктов : учебное пособие / А.В. Рудаков. – М. : Издательский центр «Академия», 2006. – 208 с.

4. Табунщик Г.В. Інженерія якості програмного забезпечення: навчальний посібник / Г.В. Табунщик, Р.К. Кудерметов, Т.І. Каплієнко. − 2-ге видання. – Запоріжжя: Дике Поле, 2016. – 176 с.

5. ISO/IEC TR 15846:1998 Информационные технологии. Процессы жизненного цикла программного обеспечения. Управление конфигурацией [Електронний ресурс]. – Режим доступу : http://www.iso.org/iso/ru/home/store/catalogue_tc/catalogue_detail.htm?csnumber=30516.