

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ**

**КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**



## **ПРОФЕСІЙНА ПРАКТИКА ПРОГРАМНОЇ ІНЖЕНЕРІЇ**

Лабораторний практикум для студентів спеціальності  
121 «Інженерія програмного забезпечення»



Київ 2020

УДК 004.412 (076.5)

ББК В 192.145 я 7

Е 605

Укладач: С. В. Поперешняк

П ПРОФЕСІЙНА ПРАКТИКА ПРОГРАМНОЇ ІНЖЕНЕРІЇ.  
Лабораторний практикум / уклад. С. В. Поперешняк – К.: Вид-во  
«Друк», 2020. – 57 с.

Містить завдання, методичні вказівки та варіанти для виконання лабораторних робіт. Складено відповідно до програми дисципліни «Професійна практика програмної інженерії».

Для студентів спеціальності 121 «Інженерія програмного забезпечення».

## ЗАГАЛЬНІ МЕТОДИЧНІ ВКАЗІВКИ

Дисципліна «Професійна практика програмної інженерії» входить до плану підготовки студентів спеціальності 121 «Інженерія програмного забезпечення» та є ключовою у підготовці фахівців даного напрямку. Ця дисципліна в аспекті сучасної галузі інженерії програмного забезпечення узагальнює та систематизує знання щодо процесів створення та супроводження програмного забезпечення.

Метою викладання дисципліни є навчання студентів сучасному інженерному підходу щодо розробки і супроводження програмного забезпечення шляхом розглядання усіх фаз його життєвого циклу; оволодіння кількісними методами та засобами їх реалізації; ознайомлення з практичними навиками та прийомами, які накопичено у цій галузі. У результаті вивчення дисципліни студент повинен знати сучасний стан розробки програмного забезпечення, включаючи моделі життєвого циклу, методи управління персоналом, економіку проектування, методи розробки тестування і супроводження програмного забезпечення та вміти застосовувати ці методи та відповідні засоби при аналізі, проектуванні, розробці, та супроводженні програмного забезпечення.

Кожна лабораторна робота виконується студентом під керівництвом викладача та включає наступні етапи:

- вивчення відповідного теоретичного матеріалу лекцій, лабораторних робіт та завдання;
- виконання завдання на комп'ютері з використанням відповідних програмних засобів та аналіз отриманих результатів.

Оцінювання знань та навиків, які здобув студент при виконання лабораторної роботи, проводиться викладачем під час захисту роботи.

## Зміст

Лабораторна робота 1. Оцінка розміру та вартості проекту за COCOMO .....	5
Лабораторна робота 2. Планування операцій та ресурсів проекту зі створення програмного забезпечення .....	9
Лабораторна робота 3. Специфікація вимог до програмного забезпечення та організація проекту для його створення .....	14
Лабораторна робота 4. Встановлення вимог до функціональності програмного забезпечення.....	17
Лабораторна робота 5. Специфікування предметної галузі проекту засобами мови UML.....	22
Лабораторна робота 6. Кількісна оцінка якості діаграм UML .....	26
Лабораторна робота 7. Моделювання поведінки ПЗ. Діаграма діяльності.....	28
Лабораторна робота 8. Проектування архітектури ПЗ .....	31
Лабораторна робота 9. Детальне проектування. Проектування інтерфейсу користувача.....	33
Лабораторна робота 10. Тестування та управління змінами програмного забезпечення .....	36
Лабораторна робота 11. Генерація вхідних кодів ПЗ. Використання зворотної інженерії для відновлення специфікації ПЗ .....	38
Домашнє завдання.....	42
Додатки.....	44
Список літератури .....	50

## Лабораторна робота 1

### ОЦІНКА РОЗМІРУ ТА ВАРТОСТІ ПРОЕКТУ ЗА СОСОМО

**Мета** – набути навичок обчислення оцінки розміру та вартості проекту за СОСОМО.

#### Теоретичні відомості

**СОСОМО** (COConstructive COst MOdel) – це множина моделей, яка дозволяє обчислити вартість проекту ПЗ на основі одиниці виміру, яка представляє собою кількість рядків коду (LOC). СОСОМО дає усереднене значення оцінок. СОСОМО включає наступні моделі:

- базова СОСОМО (застосовується у фазі специфікування вимог);
- проміжна СОСОМО (застосовується у фазах розробки множин вхідних умов проекту, наприклад - досвід персоналу, апаратні обмеження, обмеження у інструментах розробки);
- удосконалена СОСОМО (застосовується після розробки ПЗ).

Основними виразами базового СОСОМО є:

$$E = ab \times (KLOC)^{bb},$$

$$D = cb \times E^{db},$$

де  $E$  – людино-місяці проекту;  $KLOC$  – кількість тисяч рядків коду;  $ab$ ,  $bb$ ,  $cb$  та  $db$  – коефіцієнти, які дані у табл. 1;  $D$  – час розробки у календарних місяцях.

Таблиця 1

Тип проекту	ab	bb	cb	db
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Тип „Organic” представляє відносно невеликий та простий проект, який виношується невеликою командою з добрим досвідом.

Тип „Semi-detached” передбачає середній по розміру та складності проект, в якому команда має змішаний рівень досвіду і відносно жорсткі вимоги.

Тип „Embedded” представляє проект, який виконується в умовах жорстких технічних, програмних та експлуатаційних обмежень.

Для автоматизації розрахунків використовують різні інструменти, наприклад, **СОСОМО Calculator** (рис.1). Такий вигляд калькулятора відповідає базовій СОСОМО.

СОСОМО Calculator включає:

- поля, які заповнює користувач:
  - a) Code Size in KLOCs – кількість тисяч рядків коду;
  - b) Costs Per Month – витрати на виконання проекту в місяць;
  - c) Project’s notes – додаткові пояснення до проекту;
- поля, які розраховує Calculator:
  - d) Man Month – кількість людино-місяців;
  - e) Number of Men – необхідна кількість чоловік для виконання;
  - f) Calendar Time (ММ) – час виконання проекту в місяцях;
  - g) Project Casts – загальна вартість проекту.

Рис 1. Вікно Cocomo Calculator

Для проведення розрахунків за проміжною СОСОМО потрібно розширити калькулятор (рис. 2). Це дає змогу ввести додаткові проектні характеристики та провести більш точний розрахунок.

Рис 2. Розширення калькулятора

Проектні характеристики, які формують вартість проекту наведені в табл.2. Приклад: Обчислимо значення  $E$  та  $D$  при вхідних даних проекту в 50000 строк коду:

- використаємо формули:

$$E = 3 \times (50)^{1,12} = 239,8654 ;$$

$$D = 2,5 \times (239,8654)^{0,35} = 17,0187 ;$$

- застосуємо Cocomo Calculator, заповнивши відповідні поля у вікні значеннями, які задані (див. рис.3).

Таблиця 2

Характеристики проекту (Project Attributes)	
MODP	Використання інструментів ПЗ або сучасних засобів розробки
TOOL	Використання різних методів та CASE- засобів
SCED	Уплотнение графіка робіт
Характеристики програмного продукту (Product attributes)	
RELY	Требуемая надійність системи
DATA	Розмір використовуваної бази даних
CPLX	Складність системних модулів
Характеристики персоналу (Personal attributes)	
ACAP	Способности лиц, які виконують аналіз проекту
AEXP	Досвід аналітика проекту в області ПЗ
PCAP	Способности програмістів
VEXP	Досвід роботи з віртуальною машиною або знання середовища розробки
LEXP	Досвід використання даної мови програмування та засобів розробки
Характеристики апаратних засобів (Hardware attributes)	
TIME	Показники, які обмежують час виконання
STOR	Обмеження об'єму пам'яті
VIRT	Використання віртуальної машини або довершеність середовища розробки
TURN	Час обмеження розробки

The screenshot shows the Cocomo Calculator window with the following data:

Category	Organic Project	Semi-detached Project	Embedded Project
Man Months	194,57	239,87	306,14
Number of Men	10,50	14,09	19,61
Calendar Time (MM)	18,53	17,02	15,61
Project Costs	0	0	0

Input fields on the left include Code Size in KLOCs (Pessimistic: 0, Most likely: 50, Most optimistic: 0) and Costs Per Month (Avg. salary: 0, Avg. overhead: 0, Other costs: 0, One time costs: 0).

Рис 3. Результати обчислень калькулятором Cocomo

Для визначення витрат праці на розробку використовуються аналітичні і експертні методи оцінок. Вибір методу здійснюється в залежності від ступеню обліку факторів, які впливають на трудомісткість розробки. Для більш точного визначення трудомісткості розробки програмного продукту по окремим елементам і операціям процесу може бути використаний метод, в якому складаючи витрати праці визначаються з врахуванням особливості організації, яка веде розробку, і основних параметрів програмного продукту: ступеню

новизни задачі, складності алгоритму, кількості різновидів вхідної і вихідної інформації, складності організації контролю вхідної і вихідної інформації, мови програмування, використання стандартних модулів і типових задач. Надати механізм розрахунку трудомісткості і вартості робіт проекту створення інформаційної системи державних органів на стадії розробки техніко - економічного обґрунтування проекту (до початку проектування інформаційної системи). Задачі: Забезпечити єдиний підхід до оцінки трудомісткості і вартості всіх проектів створення інформаційних систем. Визначити єдині нормативи на створення, розвиток і супровід інформаційних систем. Методика включає оцінку трудомісткості тільки на розробку прикладного програмного забезпечення інформаційних систем і виключає компоненти, які вже були створені або є умова ми їхнього функціонування:

- Апаратне забезпечення (обчислювальне й телекомунікаційне устаткування);
- Готові програмні продукти (ОС, СУБД, сервера додатків, галузеві додатки й ін.) від ІТ - вендорів (Microsoft, SAP, Oracle, IBM, Fujitsu ін.);
- Готові платформ и розробки (мова програмування, СУБД, бібліотеки компонентів);
- Інженерна інфраструктура (серверні приміщення);
- Послуги зв'язку (Інтернет, виділені канали та ін.).

Робоче завдання Функціональний розмір (ФР) прикладного програмного забезпечення інформаційних систем визначається по формулі оцінки складності майбутнього проекту в балах функціональності, запропонований А.Альбрехтом:

$$\text{ФР} = (K_1 + K_2 + K_3)^{2.35}. \quad (1)$$

У формулі (1) було використано емпіричне правило — зростання розміру ПЗ втричі збільшує трудомісткість розробки і виготовлення в сім раз. Показник степені зростання трудомісткості з ростом об'єму коду дорівнює 2,35. Класифікатори проекту створення інформаційної системи:  $K_1$  - масштаб об'єкту автоматизації;  $K_2$  - тип замовника;  $K_3$  - тип програмного забезпечення, визначаються за таблицею 2 додатку 1. Розмір коду прикладного програмного забезпечення інформаційної системи в тисячах логічних рядків вихідного коду (KLOC) визначається за формулою:

$$\text{KLOC} = \text{ФР} \cdot \text{КП} / 1000,$$

де КП - коефіцієнт переводу балу функціональності в кількість логічних рядків коду, значення якого визначається за таблицею 3 додатку 1. Розрахунок трудомісткості розробки прикладного програмного забезпечення інформаційної системи в людино-місяцях (далі – Т) на основі раніше визначених даних KLOC і Е визначається за формулою:

$$T = 2.94 \cdot (\text{KLOC})^E \prod_{i=1}^7 Z_i,$$



де  $E$  – показник масштабу трудомісткості створення (розробки) програмного забезпечення інформаційної системи, який обчислюється за формулою  $E = 0.91 + 0.01 \cdot \sum_{j=1}^5 R_j$ . Формули для обчислення використані з СОСОМО II

(Методика СОСОМО дозволяє оцінити трудомісткість і час розробки програмного продукту. В моделі використовується формула регресії з параметрами, визначеними на основі галузевих даних і характеристик конкретного проекту.)

Значення кожного показника розробки –  $R_j$  ( $j = 1, \dots, 5$ ), множника затрат –  $Z_i$  ( $i = 1, \dots, 7$ ) визначаються за таблицями 4 та 5 додатку 1. Таблиця 4 додатку 1 містить нормативи кожного показника розробки  $R_j$  залежно від його рівня. Характеристики всіх рівнів по кожному показнику розробки  $R_j$  наведені в таблиці 4 додатку 1.

### Завдання

1. Обрати вхідні дані проекту для обчислення його характеристик за базовою СОСОМО та обчислити необхідні значення  $E$  та  $D$  за наведеними у лабораторній роботі формулами згідно варіанту (додаток 1).

2. За допомогою спеціалізованого засобу СОСОМО Calculator обчислити значення оцінок проекту для трьох типів проекту „Organic”, „Semi-detached” та „Embedded” за базовою та проміжною СОСОМО.

3. Виконати завдання, які сформульовані у додатку 1.

4. Порівняти розрахунки та провести аналіз.

*Примітка:* Модель СОСОМО поділяється на рівні: базовий, проміжний, деталізований. Значення драйверів витрат (додаток 1, табл. 1).

**ЛІТЕРАТУРА:** [1–2, 7-8].

## Лабораторна робота 2

### ПЛАНУВАННЯ ОПЕРАЦІЙ ТА РЕСУРСІВ ПРОЕКТУ ЗІ СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

**Мета** – набути навичок у створенні графіку робіт проекту, плануванні його ресурсів, аналізі ризиків та створенні і конструюванні звітів.

### Теоретичні відомості

Розглянемо наступні аспекти планування проектів ПЗ:

1. *Створення графіку робіт проекту.* Для планування проекту ПЗ застосовуються календарні графіки, що відображають операції, які передбачені проектом. Для автоматизації задачі створення та ведення календарного графіку проекту використовуються спеціальні програмні засоби, наприклад, Microsoft Project.

Створення календарного графіку робіт в Microsoft Project передбачає наступні етапи:

- вибір або побудова календаря. MS Project дозволяє обрати календар для окремих працівників або ресурсів із стандартних або створити специфічні

календарі на основі вмонтованого календаря шляхом визначення тривалості робочого часу, святкових та вихідних днів (рис. 4);

- визначення операцій проекту, назви яких перелічуються у вигляді списку;
- призначення початку та закінчення проекту та тривалості кожної операції;
- формування логічної послідовності операцій.

MS Project будує календарний графік у вигляді діаграми Ганта, яку можливо відкрити у панелі меню **Вид→Діаграма Ганта** (рис. 5). У колонці **Назва задачі** вписуються етапи (задачі) розробки системи. **Задача** – дія, яка має початок та завершення. **Сумарна задача** – задача, яка вміщує під задачею і формується з результатів під задачею. **Перервана задача** – розподіл виконання задачі у календарному плані (наприклад, якщо для виконання задачі потрібно два дні, то можливо розподілити частину роботи на понеділок, а частину на четвер).

Колонка **Тривалість** показує тривалість кожної задачі у днях. Надалі у колонках **Початок** та **Закінчення** вказується початок та закінчення етапів розробки. Подальші колонки користувач може налаштувати за потребами розроблюємого проекту.

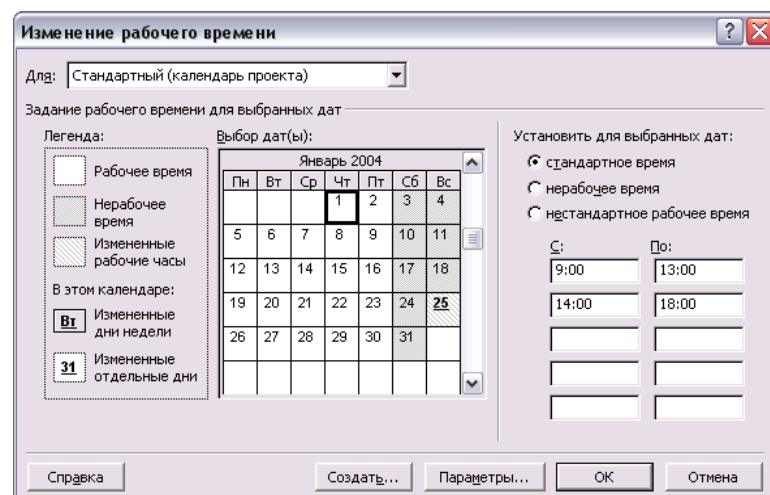


Рис 4. Формування календаря

Ид.	Название задачи	Длительность	Начало	Окончание	07	29 Окт '07
1	Анализ предметной области	3 дней	Чт 25.10.07	Пн 29.10.07	Ч	П
2	Определение целей и задач	1 день	Чт 25.10.07	Чт 25.10.07	П	С
3	Предварительный анализ затрат	1 день	Пт 26.10.07	Пт 26.10.07	В	П
4	Определение основных вех	1 день	Пн 29.10.07	Пн 29.10.07	П	С
5	Анализ ПО завершён	0 дней	Пн 29.10.07	Пн 29.10.07	П	С
6	Проектирование	20 дней	Вт 30.10.07	Пн 26.11.07	В	П
7	Разработка	60 дней	Вт 27.11.07	Пн 18.02.08	П	С
8	Тестирование	15 дней	Вт 19.02.08	Пн 10.03.08	П	С

Рис 5. Календарний графік у вигляді діаграми Ганта

На діаграмі Ганта потрібно зв'язати задачі одним з чотирьох типів зв'язків:

- 1) Finish-to-Start (FS) (завершення до початку) – наступна операція починається тільки після закінчення попередньої;

2) Start-to-Start (SS) (початок до початку) – операції розпочинаються одночасно;

3) Finish-to-Finish (FF) (завершення до завершення) – операції повинні закінчитися одночасно;

4) Start-to-Finish (SF) (початок до завершення) – одна операція не може завершитися доки інша не розпочнеться.

На діаграмі можна призначити операції, які є **контрольними точками проекту** (віхи) – точки, що визначають важливу подію та використовується для контролю за ходом проекту. Такі операції мають нульову тривалість та відображаються ромбами (рис. 5).

**Критичними** називають операції, затримка яких може впливати на дату закінчення проекту. Всі критичні операції проекту створюють **критичний шлях**.

MS Project дозволяє також представляти календарний план робіт проекту у вигляді сіткового графіку, який виводиться з меню **Вид→Сітковий графік** (рис. 6).

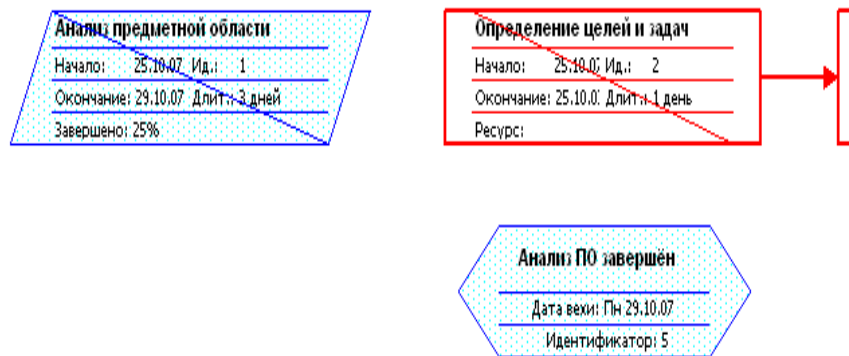


Рис.6. Календарний план робіт проекту у вигляді сіткового графіку


Блоки сіткового графіка можуть розрізнятися кольором та формою у залежності від типу задачі (сумарна задача, задача або контрольна точка проекту) та її стану (виконується, не виконується, завершена). На блоці може бути вказана додаткова інформація, наприклад назва задачі, дати початку та завершення, ресурси, процент завершення і т.ін. На сітковому графіку паралелепіпедами визначені сумарні задачі, прямокутниками — задачі, а багатокутниками — контрольні точки. Розпочаті задачі перекреслені однією лінією, а завершені — двома.

2. *Планування ресурсів проекту.* Будь-який проект ПЗ потребує певних ресурсів для його реалізації. MS Project підтримує засоби для автоматизованого призначення та обліку ресурсів проекту. Спочатку рекомендовано створити таблицю ресурсів проекту. До неї вносяться назви ресурсів, їх вартість, кількість, календар використання і т. ін. (табл.3). Для призначення ресурсів потрібно в меню **Вид→Лист ресурсів** ввести потрібні ресурси та усі їх атрибути. Ресурси бувають двох типів: матеріальний (обладнання та різні додаткові ресурси – папір, CD і т.ін.) та трудовий (співробітники, які приймають участь у проекті).

Таблиця 3

№	Назва ресурсу	Тип	Одиниці виміру	Стандартна ставка	Базовий календар
1	комп'ютер	матеріальний	шт	0 грн	
2	менеджер	трудоий		40,00 грн/г	стандартний
3	дизайнер	трудоий		15,00 грн/г	стандартний

Надалі, окремо для кожної операції відбувається призначення ресурсів (рис.7).

MS Project автоматично обчислює завантаження кожного ресурсу. Якщо ресурс перевантажений, то він позначається на діаграмі символом . У цьому випадку потрібне розвантаження – передача частини роботи іншому ресурсу, зменшення роботи взагалі чи призначення додаткового ресурсу і т. ін.

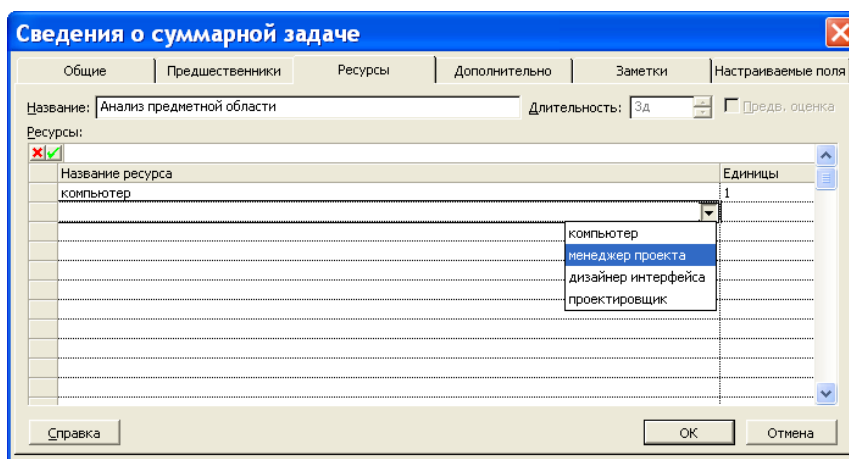


Рис.7. Призначення ресурсів для кожної з операцій

При створення призначення трудовитрати розподіляються рівномірно. Це називається **плаский профіль загрузки**. Але інколи потрібно розподіл проводити інакше. Створення профілю проводиться в меню **Вид→Використання задач**, у полі **Назва задачі** вибрати ресурс і на вкладці **Загальні** вибрати профіль. Профілі мають наступні типи:

- плаский;
- загрузка в кінці (розподіл трудозатрат так, що основна загрузка проводиться в останні дні задачі);
- загрузка на початку (основна загрузка проводиться в перші дні задачі);
- двійний пік (вміщує два піка в середині задачі);
- ранній пік (планує пікову загрузку ближче до початку задачі);
- пізній пік (планує пікову загрузку ближче до кінця задачі);
- колокол (пік роботи приходить на середину виконання задачі);
- черепаха (основна загрузка всередині, а на початку та в кінці зменшується).

3. *Аналіз ризиків проекту.* При проведення планування один із самих складних етапів аналіз ризиків проекту. Від його проведення залежить наскільки успішно буде виконаний проект. Аналіз ризиків вміщує наступні етапи: визначення можливих ризиків, для кожного з них визначення стратегії пом'якшення впливу ризику на проект (дії, що виконуються для запобігання ризику або, якщо він здійснився, для успішного завершення проекту).

Аналіз ресурсних ризиків – визначення ресурсів та призначень, які можуть збільшити ймовірність зриву проекту. Наприклад, ризиковано ставити на відповідальні роботи співробітники з невеликим досвідом роботи. Або ризик — використання одного співробітника в багатьох задачах, оскільки проект буде залежним від одного співробітника і якщо він стане недоступним, то проект може провалитися.

Для позначення ризиків в меню **Лист ресурсів** вставляється новий стовпчик з ім'ям поля **Флаг1** (текст заголовка – **Досвід**). Далі в меню **Сервіс**→**Настройка**→**Поля** можливо визначити графічне відображення, наприклад зеленого індикатора коли значення поля буде **Так (Yes)** для позначення досвіду співробітників і червоного індикатора, коли значення поля буде **Ні (No)** (рис. 8).

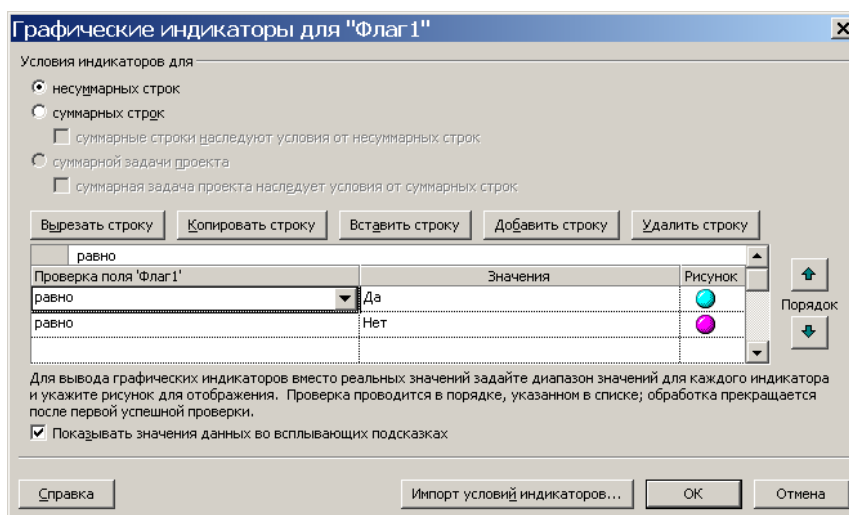


Рис. 8. Представлення ризиків

MS Project підтримує проведення аналіз тривалості виконання проекту за методом PERT (Program Evaluation and Review Technique), який передбачає процес оцінки можливих результатів на основі сценаріїв: оптимістичного, очікуваного та песимістичного. Для проведення аналізу потрібно призначити оптимістичну, песимістичну та очікувану тривалість операцій. Microsoft Project автоматично розраховує звантажені значення. За замовчуванням, оптимістична і песимістична тривалості мають ваговий коефіцієнт 1, а очікувана - 4, при загальній мірі 6. Зважені значення можна використовувати для отримання більш надійної оцінки тривалості задач (рис.9). Провести аналіз можливо у меню **Вид**→**Панелі інструментів**→**Аналіз за методом PERT**.

Аналіз бюджетних ризиків проводиться визначенням кількісних характеристик відхилень (процент відхилення) залежить від прийнятих в

організації стандартів. Наприклад: до 20% - допущення; до 35% - виникнення ризику перевищення бюджету; 50% - неприйнятно, потрібно приймати рішення про термінові дії або закриття проекту. Налаштування формул розрахунків проводиться в меню **Сервіс→Налаштування→Поля**.

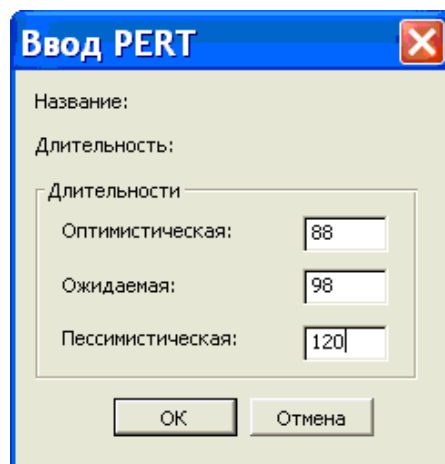


Рис. 9. Введення значень для отримання більш надійної оцінки тривалості задач

### **Завдання**

1. Створити новий проект у Microsoft Project (варіанти завдань у додатку 2).
2. Побудувати календарний графік робіт відповідно до етапів життєвого циклу розробки ПЗ.
3. Визначити критичний шлях проекту.
4. Представити план робіт у вигляді сіткового графіку.
5. Створити та заповнити таблицю ресурсів проекту.
6. Призначити ресурси, які необхідні для реалізації кожної операції проекту.
7. Усунути перевантаження ресурсів проекту (якщо воно є).
8. Призначити профілі завантаження.

**ЛІТЕРАТУРА:** [1–2, 7-8].

### **Лабораторна робота 3**

#### **СПЕЦИФІКАЦІЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ОРГАНІЗАЦІЯ ПРОЕКТУ ДЛЯ ЙОГО СТВОРЕННЯ**

**Мета** – набути навичок у створенні графіку робіт проекту, плануванні його ресурсів, аналізі ризиків та створенні і конструюванні звітів.

#### **Теоретичні відомості**

1. *Документ опису вимог (технічне завдання).* Документ опису вимог є основним документом, який визначає вимоги та порядок створення програмного забезпечення та її прийняття при вводі в експлуатацію. В Україні в якості такого документа використовується технічне завдання, зміст якого визначається діючим стандартом ГОСТ 34.602-89 на створення автоматизованих систем. Нижче наведено спрощений зміст технічного завдання:

1. Загальні відомості (повне найменування роботи, умовне позначення, шифр (номер) договору, найменування та реквізити замовника та розробника системи, планові терміни початку та закінчення робіт, відомості про джерела та порядок фінансування робіт).

2. Призначення та цілі створення ПЗ (від діяльності, що автоматизується, перелік об'єктів автоматизації, основні показники, що повинні бути досягнуті в результаті впровадження ПЗ).

3. Характеристика об'єктів автоматизації (короткі відомості щодо об'єкта автоматизації або посилання на документи, що містять такі відомості; відомості щодо експлуатації об'єкта автоматизації та характеристики оточуючого середовища).

4. Вимоги до ПЗ:

- вимоги до ПЗ в цілому (структура, основні компоненти або підсистеми, чисельність та кваліфікація персоналу, режим роботи; вимоги до: надійності, продуктивності, ергономічності, технічної естетики, експлуатації, технічному забезпеченні та зберіганні, захисту інформації від несанкціонованого доступу, зберіганню інформації при аваріях, захисту від зовнішніх чинників, патентної чистоти, стандартизації та уніфікації);

- функціональні вимоги, які виконуються ПЗ (перелік підсистем, їх призначення, взаємодія між собою; вимоги для роботи з іншими системами, режиму функціонування, діагностування системи, перспективи розвитку та модернізації);

- вимоги до видів забезпечення (технічного, системного програмного, інформаційного).

5. Склад та зміст робіт зі створення системи (перелік етапів, терміни, результати, фінансування).

6. Вимоги до документування (перелік документів, які підлягають розробці).

7. Джерела розробки (перелік літератури та посилань).

2. *Організація планування управління проектом створення ПЗ.* Для досягнення ефективного, стабільного проектування та розробки ПЗ потрібно мати чітку організацію відповідних робіт, яка забезпечується в межах спеціального проекту. Опис проекту забезпечується певними структурами, а саме:

- робоча структура (Work Breakdown Structure – WBS);
- організаційна структура (Organization Breakdown Structure – OBS).

**WBS** — ієрархічна структура, необхідна для логічного детального розподілу усіх робіт на окремі операції, які плануються відповідно до обраної моделі життєвого циклу (рис. 10). **OBS** — це внутрішня організація команди проекту (рис. 11).

При поєднанні цих структур можна отримати **двонаправлену модель управління проектом**, яка показує відповідальність учасників проекту за певні етапи розробки (рис 12).



Рис. 10. Приклад ієрархічної структури розподілу робіт



Рис. 11. Приклад внутрішньої організації команди

При потребі можливо додавання до двонаправленої моделі третього напрямку – затратної структури проекту (Cost Breakdown Structure — CBS), яка визначає вартість етапів проекту. Але це загалом перевантажує структуру, тому більш додільне використання двонаправленої моделі.

### Завдання

1. Скласти документ опису вимог до системи, що проектується.
2. Обрати модель життєвого циклу системи розробки ПЗ.
3. Специфікувати проект шляхом побудови WBS, OBS та двонаправленої моделі управління проектом.

**ЛІТЕРАТУРА:** [1–2, 7-8].



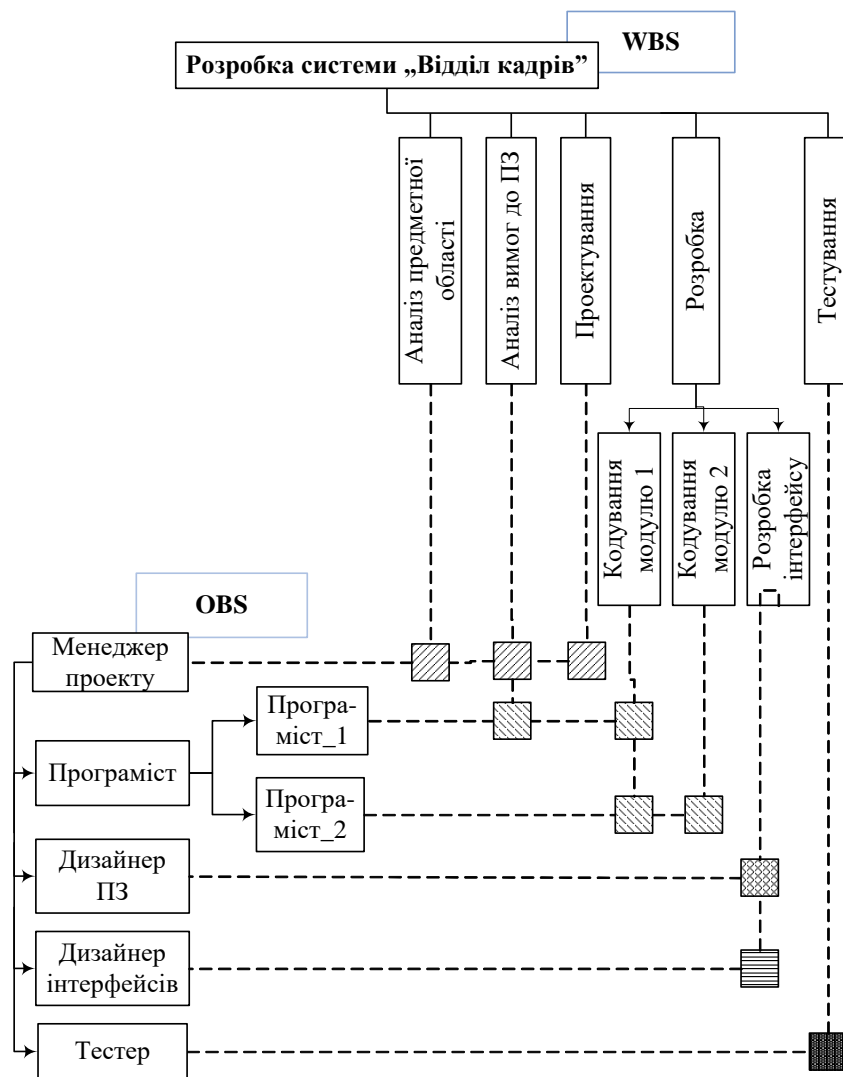


Рис. 12. Приклад двонаправленої моделі управління проектом

## Лабораторна робота 4

### ВСТАНОВЛЕННЯ ВИМОГ ДО ФУНКЦІОНАЛЬНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

**Мета** – набути навичок у встановленні вимог до функціональності програмного забезпечення.

#### Теоретичні відомості

Встановлення вимог – це перший етап життєвого циклу розробки системи. Мета встановлення вимог у тому, щоб надати розгорнуте визначення вимог, які учасники проекту мають отримати у системі, яка буде реалізовуватися. Специфікації загальних вимог до функцій програмного забезпечення (ПЗ) можуть бути представлені у двох формах:

- графічній, яка має високу наочність;
- вербальній, яка дозволяє деталізувати опис вимоги.

1. *Графічна форма специфікації. Діаграми прецедентів UML.* Діаграми прецедентів (Use Case Diagram) є графічним засобом специфікування вимог, які використовуються для визначення наступного:

–загальних меж та контексту предметної області, в якій повинне функціонувати ПЗ;

–загальних вимог до функціональної поведінки ПЗ, що проектується;

–взаємодії ПЗ, що проектується, із зовнішнім середовищем.

Діаграма будується за допомогою графічних елементів, які наведені у табл.4.

Таблиця 4

Класифікатор	Функція класифікатору	Нотація
Суб'єкт (Actor)	Абстрактний опис сутності, яка знаходиться поза межами системи та на пряму взаємодіє з системою.	
Варіант використання (Use case)	Специфікація поведінки системи або її частини при взаємодії з суб'єктами	
Асоціація	Комунікативний зв'язок. Позначається взаємодія суб'єкта з системою	
Включення «include»	Відношення між базовим та тим варіант використання, який включається в нього. За його допомогою визначається яким чином поведінку у варіанті, який включають можливо вставити у базовий варіант.	
Розширення «extend»	Відношення між базовим та тим варіант використання, який розширює його. За його допомогою визначається яким чином поведінку у базовому варіанті можливо розширити. Виконується тільки після певної умови у базовому варіанті, в протилежному випадку не виконується зовсім.	
Нотатки	Для довільної текстової інформації, яка має відношення до проекту. З'єднується пунктирною лінією.	
Успадкування	Суб'єкт чи варіант використання успадковує структуру та поведінку загального предка об'єкту, визначаючи при цьому власні властивості та поведінку в системі.	

Побудова діаграми прецедентів відбувається у такій послідовності:

- визначення діючих осіб (actors), які будуть мати відношення до систем ;
- визначення варіантів використання системи (use case), виходячи з потреб діючих осіб;

- встановлення зв'язків між суб'єктами та варіантами використання.

Зв'язки включення та розширення обов'язково підписуються позначеннями «include» та «extend» (рис.13).

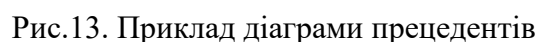
При описі прецедентів розширення посилання на точки розширення розміщуються у вигляді списку в розділі extension points (точки розширення) та беруться у круглі дужки. Зв'язок «extend» підписується наступним чином: у

2. *Вербальні специфікації прецедентів.* Кожен основний прецедент повинен бути описаний за допомогою документально зафіксованого потоку подій. Вербальні специфікації всіх прецедентів одного проекту повинні мати однакову структуру та вмішувати наступні розділи:

- Прецеденти включення та розширення описуються в основних, від яких вони походять.

## 1.Короткий опис

**2.Суб'єкт – Інспектор, Адміністратор.**



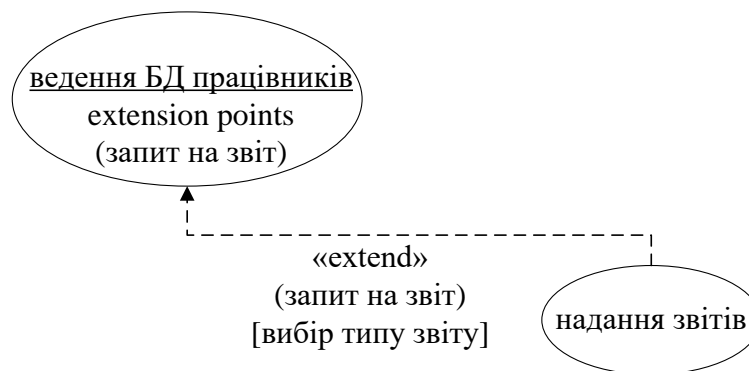


Рис. 14. Приклад оформлення зв'язка «extend»

### 3.Передумова

Користувач авторизується для входу в БД.

### 4.Основний потік

4.1. Система відображує форму ведення БД. Форма вміщує наступне:

4.1.1. Форма має заголовок **Дані Працівників**.

4.1.2. Поля для заповнення або змінювання користувачем на вкладці **Особисті дані Працівників**:

4.1.2.1. Прізвище, ім'я, по-батькові.

4.1.2.2. Домашня адреса

4.1.2.3. Телефон ...

4.1.3. Поля для заповнення або змінювання користувачем на вкладці **Службова інформація**:

4.1.3.1. Дата прийняття на роботу.

4.1.3.2. Посада...

4.1.4. Поля, які користувач помічає маркером:

4.1.4.1. Спосіб доставки: кур'єр...

4.1.4.2. Спосіб оплати: каса; перечислення на банківський рахунок...

4.2. Якщо користувач натискає кнопку **Збереження**, дані відправляються на сервер та заносяться до БД.

4.2.1. Якщо форма з неповною інформацією, виконується А1.

4.2.2. Якщо сервер не в змозі внести дані до БД, виконується А2.

4.3. Якщо проводиться запит на надання звітів, користувач помічає маркером поля для виводу запиту на екран:

4.3.1. Загальний звіт.

4.3.2. Звіт за поточний місяць.

4.4. Якщо проводиться запит на архівування звітів, користувач натискає кнопку **Архівування**, дані відправляються на сервер та створюється архівна копія БД, в протилежному випадку виконується А3.

### 5.Альтернативні потоки

**А1. Неповна інформація.** Якщо інспектор не заповнив усі необхідні поля, система дає можливість ввести інформацію. Прецедент продовжується.

*A2. Помилка серверу.* Якщо сервер не заносить замовлення до БД, то система видає повідомлення «Помилка при зберіганні!» та надає можливість внести дані ще раз.

*A3. Помилка архівації.* Якщо сервер не в змозі провести архівування БД, то система видає повідомлення «Помилка архівації, спробуйте ще раз!».

## **6.Постумови**

Якщо внесення, змінювання та архівування даних, надання звітів пройшло вдало, дані записуються до БД або видаються на екран. У протилежному випадку стан системи залишається незмінним.

Після проведення аналізу ризиків складається план реагування на ризики (табл.5). Поля **Опис ризику проекту** та **План реагування на ризики** встановлюються як текстові, а поле **Ймовірність** – як **Флаг**.

*4. Створення та конструювання звітів проекту.* MS Project надає засоби для створення звітів, які дозволяють отримати документацію проекту.

Таблиця 5

	<b>Назва ресурсу</b>	<b>Опис ризику проекту</b>	<b>Ймовірність</b>	<b>План реагування на ризики</b>
1	менеджер проекту	зрив виконання через недоступність ресурсу	висока	залучення додаткових ресурсів...
2	дизайнер		невисока	
3	проектувальник	зрив виконання через брак досвіду	висока	щоденний контроль менеджером проекту

MS Project має набір вбудованих форм звітів, а також дозволяє користувачу конструювати власні форми звітів.

MS Project має шість категорій стандартних звітів:

- оглядові;
- поточної діяльності;
- витрати;
- призначення;
- завантаження;
- такі, що налагоджуються.

Виклик конструктора відбувається через меню **Вид → Звіти**. Для конструювання власного звіту треба використовувати категорію „Такі, що налагоджуються”, у якій можливо обирати параметри звіту, якщо не задовольняють стандартні звіти.

MS Project дозволяє публікувати дані про проект у вигляді Web-сторінок, що дає можливість розповсюджувати дані проекту між територіально розгалуженими виконавцями та зберігати їх у наступних форматах: HTML; Microsoft Access; Microsoft Excel; текст (.txt); XML.

## **Завдання**

1. За узгодженням з викладачем обрати варіант завдання (предметну галузь) для виконання лабораторних робіт.

2. Провести попередній аналіз предметної галузі, визначити функції ПЗ, що проектується.

3. Побудувати діаграму прецедентів на основі проведеного попереднього аналізу.

4. Для прецедентів, визначених у діаграмі прецедентів побудувати вербальні специфікації.

5. Провести аналізи ресурсних, бюджетних ризиків та ризиків щодо тривалості виконання проекту.

6. Скласти план реагування на усі виявлені ризики.

7. Створити наступні звіти проекту:

- загальний звіт проекту (Оглядові → зведений по проекту);
- календар проекту;
- призначення операцій за виконавцями;
- контрольні точки проекту;
- лист ресурсів та їх завантаження;
- звіт про бюджет проекту.

8. Провести зберігання проекту у наступних форматах: HTML; Microsoft Access; Microsoft Excel ; текст; XML.

9. Створити засобами MS Project власну Web-сторінку проекту. Використати дані які найбільш точно для замовника будуть відображати виконання проекту.

**ЛІТЕРАТУРА: [1–2, 7-8].**

## **Лабораторна робота 5**

### **СПЕЦИФІКУВАННЯ ПРЕДМЕТНОЇ ГАЛУЗІ ПРОЕКТУ ЗАСОБАМИ МОВИ UML.**

**Мета** – набути навичок у специфікуванні предметної галузі проекту засобами мови UML.

### **Теоретичні відомості**

У цій роботі розглядаються такі аспекти специфікування предметної галузі засобами UML: опис класів, побудова діаграми класів, використання стереотипів, використання пакетів.

#### *1. Опис класів. Побудова діаграми класів*

Діаграма класів (Class diagram) призначена для відображення статичної структури ПЗ проекту, що проектується. Діаграма містить класи і взаємозв'язки між ними та дозволяє описати їх структуру та типи відношень.

Клас - це група об'єктів із спільними властивостями (атрибутами), функціями та відношеннями з іншими об'єктами. Графічна нотація класу розділяється на три сектори: ім'я класу, атрибути та операції (рис. 15).

Об'єкт (екземпляр класу) – це представлення реальної або абстрактної сутності. Екземплярів класу може бути будь-яка кількість. Але в деяких випадках їх потрібно обмежити. Такі випадки виникають коли потрібно задати клас, у якого:

- немає жодного екземпляра (тоді клас становиться службовим (Utility));

- є один екземпляр - синглетний клас (Singleton).

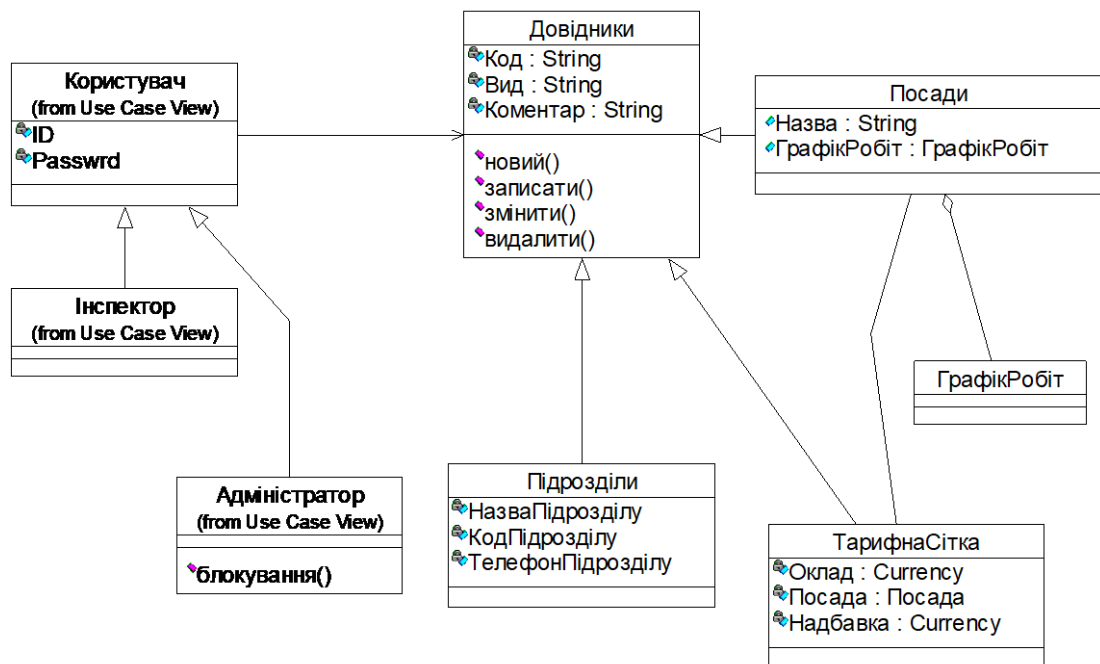


Рис. 15. Приклад діаграми класів

**Ім'я класу** – унікальне ім'я в рамках пакету, що записується з великої букви без пропусків. При необхідності до імені класу додається ім'я пакету, до якого відноситься клас, згідно з таким синтаксисом:

<Ім'я\_пакету>::<Ім'я\_класу>

**Ім'я об'єкту** обов'язково підкреслюється та записується згідно з наступним синтаксисом:

<Ім'я\_об'єкту> : <Ім'я\_класу>

або без запису імені (тобто об'єкт анонімний):

: <Ім'я\_класу>

**Атрибут** є властивістю класу, якому надається ім'я, тип та інші характеристики згідно з таким синтаксисом:

<ознака видимості> <ім'я атрибуту>[кратність] : <тип атрибуту> =  
 <значення за умовчанням> {строка- властивість}

**Операція** - це опис способу виконання дій з об'єктом, наприклад: читання або змінення значень атрибутів, обчислення нових значень за інформацією, яка знаходиться в об'єкті і т. ін. Синтаксис опису операцій:

<ознака видимості> <ім'я операції> (<список-параметрів>):  
 <тип виразу, який повертає значення> {строка- властивість}

Ознака видимості певного класу показує чи має інший клас доступ до атрибутів та операцій цього класу. Основні ознаки видимості наведені у табл. 6.

Усі додаткові види ознак видимості визначаються розробниками, інструментами моделювання та генератором коду.

Таблиця 6

Значення	Опис
public (загальний)	доступний для усіх клієнтів класу
protected (захищений)	доступний тільки для підкласів і друзів класу
private (секретний)	доступний тільки для друзів класу
implementation (реалізація)	доступний тільки всередині пакету, що містить даний клас

Строка-властивість використовується для значень атрибутів, які не можливо змінити в програмі при роботі з даним типом об'єктів. Фігурні дужки означають фіксоване значення відповідного атрибута для класу в цілому, яке повинні приймати усі екземпляри класу без виключення.

Наприклад, атрибут «заробітна\_плата: Currency={\$100}» означає фіксовану заробітну плату у 100 \$ для кожного об'єкту класу «Працівник».

Між класами можна встановлювати зв'язки таких видів:

1. асоціація – зв'язок між об'єктами;
2. агрегація – відношення виду „частина-ціле” між класом, який є збіркою (агрегатом), що включає в себе інші класи, та цими класами;
3. композиція – агрегація, в якій об'єкт, що включається, не може існувати без класу-агрегату (рис.16);
4. успадкування – клас-потомок успадковує структуру та поведінку предка, визначаючи при цьому власні атрибути та операції.

Кратність зв'язків вказує на кількість класів або об'єктів, що можуть взаємодіяти за даним зв'язком, та позначається так:

- 0..1 - жодного або один;
- 0..\* - від нуля до безлічі;
- 1..1 - обов'язкова приналежність;
- 1..\* - від одного до безлічі (рис.16).

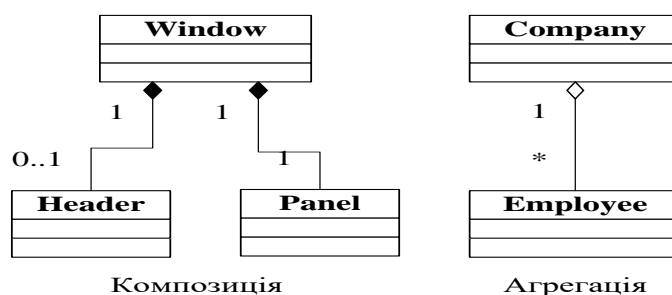


Рис. 16. Приклад успадкування клас-потомок

## 2. Стереотипи. Пакети

Стереотип – це механізм розширення UML, якій дозволяє створювати нові елементи мови на основі існуючих. Якщо є необхідність створити елемент, якого немає у UML, але він схожий на будь-який існуючий елемент, то можна сконструювати його як стереотип. Наприклад, елемент „база даних” може бути введений у UML як стереотип класу і буде розглядатися як специфічний клас. Назва стереотипу обмежується символами « ».



Стандартні розширення мови UML включають у себе наступні стереотипи (рис. 17):

- сутність («entity»). Описує пасивні об'єкти. Вони не можуть ініціювати взаємодії;
- граничний («boundary»). Описує об'єкти, які є посередниками між системою та діючими особами, що знаходяться за її межами;
- керування («control»). Описує об'єкти, які керують взаємодіями (наприклад, контролер зовнішнього пристрою). Його поведінка відповідає поведінці варіанта використання;
- інтерфейс («Interface»). Це множина абстрактних операцій, які використовуються для визначення послуг класу або компонента. У нього немає атрибутів і обов'язково відноситься до якогось класу з яким з'єднується прямою лінією. Ім'я інтерфейсу обов'язково повинно починатися з великої букви „I”;
- стандартний.

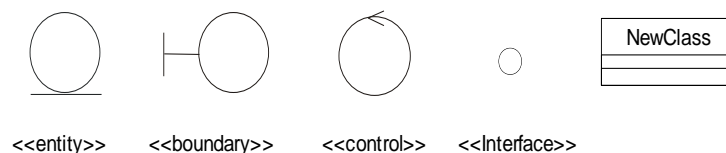


Рис. 17. Стандартні розширення мови UML

Пакети відображають структуру ПЗ проекту - розподіл на підсистеми та залежності між ними. Пакети не є засобом декомпозиції самої системи, а призначені для структуризації моделі, яка створюється засобами UML, тобто, дозволяє „розташувати” діаграми моделі у „папки” (рис. 18). Є декілька способів організації пакетів: за видом моделі, за функціональністю або за будь-якою ознакою, яку обере розробник.

Між пакетами можна встановлювати відношення вкладеності, які відображаються на діаграмі пакетів у вигляді зв'язків.

Крім того, для пакетів можна встановлювати властивості видимості. Видимість пакетів **public** (загальний) дозволяє іншим пакетам доступ до всіх елементів пакету. Елементи **protected** (захищений) дають можливість доступу тільки потомкам того пакету, у якому вони знаходяться. Видимість **private** (секретний) закриває зовні доступ до елементів пакету або пакетів, що вкладені в нього.

Верхній рівень діаграми пакетів вміщує повний опис системи, що проектується.



Рис. 18. Приклад структуризації моделі

### Завдання

1. Виявити класи, які відносяться до системи, що проектується. Описати атрибути і операції кожного класу.

2. Виявити зв'язки між класами.
3. Задokumentувати кожен клас.
4. Побудувати діаграму класів з використанням різних стереотипів класів.
5. Побудувати діаграму пакетів.
6. Задokumentувати вкладеність кожного пакету у формі: назва пакету – опис вмісту.

**ЛІТЕРАТУРА:** [3, 5, 7-8].

## **Лабораторна робота 6**

### **КІЛЬКІСНА ОЦІНКА ЯКОСТІ ДІАГРАМ UML**

**Мета** – набути навичок у формуванні кількісної оцінки якості діаграм UML.

#### **Теоретичні відомості**

При побудові діаграм виникає питання їх якості, для оцінки якої можна використовувати кількісні показники. Одна з можливих методик оцінки діаграм включає в себе формули, наведені нижче. При занадто низьких значеннях цих показників діаграма вважається недостатньо інформативною, а при занадто високих - важкою для сприйняття. Формула для діаграми:

$$S = \frac{\sum S_{Obj} + \sum S_{Lnk}}{1 + Obj + \sqrt{T_{Obj} + T_{Lnk}}},$$

де  $S$  – показник якості діаграми;  $S_{Obj}$  – оцінка складності елементів діаграми;  $S_{Lnk}$  – оцінка складності зв'язків;  $Obj$  – число об'єктів діаграми;  $T_{Obj}$  – число типів об'єктів;  $T_{Lnk}$  – число типів зв'язків.

Формула для обрахунку класу:

$$S_{cls} = \frac{\sqrt{Op} + \sqrt{Art}}{0,3 \times (Op + Art)},$$

де  $S_{cls}$  – показник якості класу;  $Op$  – число операцій класу;  $Art$  - число атрибутів класу.

У табл. 7 наведено емпіричні оцінки складності елементів та зв'язків діаграми.

У табл. 8 наведено діапазони значень показників якості діаграм, які рекомендуються.

Таблиця 7

Тип елементу	Оцінка $S_{Obj}$	Тип зв'язку	Оцінка $S_{Lnk}$
Вузол (node)	3	Асоціація (association)	1
Інтерфейс (interface)	4		
Клас (class)	5	Агрегування (aggregation)	2
Компонент (component)	4		
Пакет (package)	4	Залежність	2

Прецедент (use case)	2	(dependency)	
Примітка (node)	2	Композиція (composition)	3
Процесор (processor)	2		
Стан (state)	4	Успадкування (generalization)	3

Таблиця 8

Тип діаграми	Діапазон оцінок
Прецедентів (use case)	2,5 – 3
Класів (class)– без атрибутів та операцій	3 – 3,5
Класів (class) – з атрибутами та операціями	5 – 9,5
Пакетів (package)	3,5 – 4
Стану (state)	2,5 – 3
Послідовності (sequences)	3 – 3,5
Кооперації (collaboration)	3,5 – 4
Компонентів (component)	3,5 – 4
Розгортання (deployment)	2 – 2,5

Приклад визначення показників якості для діаграми класів, що наведена на рис.15, розрахований нижче.

Спочатку проводяться оцінки для окремих класів. Розрахунок показника для класу „Довідники” здійснюється за формулою:

$$S_{cls} = \frac{\sqrt{Op} + \sqrt{Art}}{0,3 \times (Op + Art)} = \frac{\sqrt{4} + \sqrt{3}}{0,3 \times (4 + 3)} = 1,24$$

Аналогічно проводиться оцінка для інших класів: „Підрозділи” – 1,9; „ТарифнаСітка” – 1,9; „Посади” – 2,4; „Користувач” – 2,4; „Адміністратор” – 3,33; „Працівники” – 1,9. Класи „ГрафікРобіт” та „Інспектор” без атрибутів та операцій, тому вони важать по 5 балів кожен. Обчислення показника якості всієї діаграми:

$$S = \frac{8 \times 5 + 1,24 + 1,9 + 1,9 + 2,4 + 2,4 + 3,33 + 1,9 + 23}{1 + 9 + \sqrt{1 + 3}} \approx 6,5$$

Висновок: діаграма входить до діапазону значень показника якості діаграм класів з атрибутами та операціями, який становить 5 – 9,5 од.

### Завдання

1. Провести кількісну оцінку якості діаграми класів, яка була розроблена у попередній роботі.
2. Провести кількісну оцінку якості діаграми пакетів, яка була розроблена попередній роботі..
3. Провести кількісну оцінку якості діаграми прецедентів, яка була розроблена у лабораторній роботі 1.

**ЛІТЕРАТУРА:** [3, 5, 7-8].

## Лабораторна робота 7

### МОДЕЛЮВАННЯ ПОВЕДІНКИ ПЗ

**Мета** – набути навичок у створенні діаграм діяльності, взаємодії та стану специфікування динамічної поведінки програмного забезпечення.

#### Теоретичні відомості

Для специфікування динамічної поведінки програмного забезпечення UML надає такі види діаграм: діяльності, взаємодії, стану.

1. *Діаграма діяльності.* Основне призначення діаграми діяльності (Activity diagram) полягає у відображенні потоків робіт та операцій, за допомогою елементів, які наведені у табл. 9. Приклад фрагменту діаграми діяльності з послідовними, так паралельними потоками робіт представлений на рис. 19. Діаграма діяльності супроводжується додатковими специфікаціями, які містять детальний аналіз кожної дії (табл. 10).


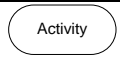


2. *Діаграми взаємодії.* Діаграми взаємодії (interaction diagrams) використовують для моделювання взаємодії між об'єктами ПЗ. Вони розділяються на діаграми послідовності (Sequence diagram) та діаграми кооперації (Collaboration diagram).

Діаграма послідовності – моделює взаємодію об'єктів у часі. Графічно об'єкт зображується прямокутником, у якому записують ім'я об'єкту та класу через двокрапку.

Кожен об'єкт має лінію життя (object lifeline), яка зображується пунктирною вертикальною лінією та означає період часу, у якому об'єкт присутній у взаємодії (рис. 20). Непотрібні об'єкти знищуються за допомогою символу «X», після цього об'єкт припиняє існування у поточному процесі.

Об'єкт має змогу виконувати дію (ініціювати повідомлення), тільки отримавши на себе фокус керування (focus of control), зображений у формі витягнутого вузького прямокутника.

Таблиця 9

Класифікатор	Функція	Нотація
Початковий стан (start)	Початковий стан дій. Буває тільки один	
Дія (activity)	Фрагмент дії	
Перехід (transition)	Напрямок дії	
Точка прийняття рішень (decision point)	Умова для можливих альтернативних шляхів. Задається у вигляді питання. Відповіді на переходах обов'язково беруться у квадратні дужки	
Лінії синхронізації (synchronization bars)	Між лініями знаходяться дії, які виконуються паралельно	
Кінцевий стан (end)	Кінець дій. Може бути декілька, якщо є альтернативні варіанти закінчення дії.	

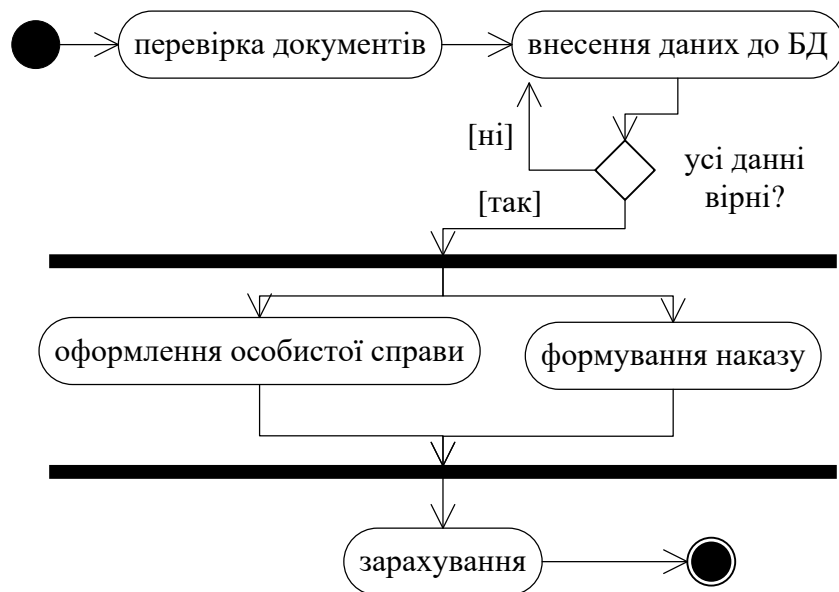


Рис. 19. Приклад фрагменту діаграми діяльності з послідовними та паралельними потоками робіт

Таблиця 10

Формулювання прецеденту	Стан виду діяльності
1.Перевірка документів	Перевірка наявних документів: паспорт, трудова книжка, фотографії, особистий листок ...
2.Внесення даних до БД	Передбачає внесення даних про працівника до БД підприємства та присвоєння йому табельного номеру, розпорядку робочого дня, тарифної ставки оплати праці ...
3. ...	...

Часові обмеження на діаграмі позначаються так:

- {час\_прийому\_повідомлення < 1 с};
- {час\_простою\_системи = 5 с}.

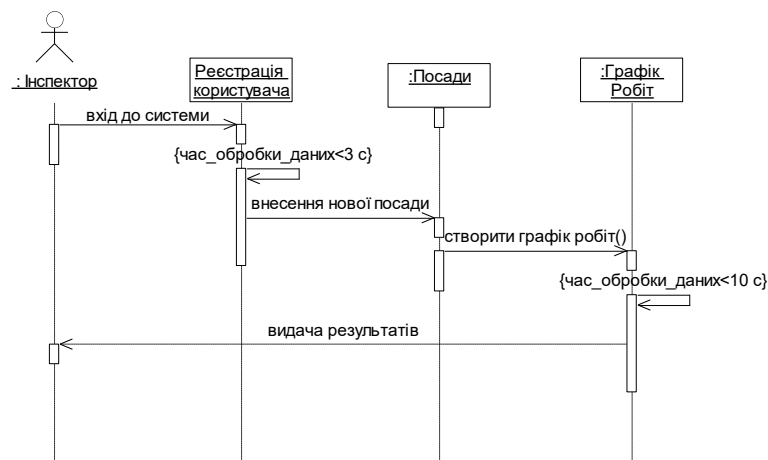


Рис. 20 Приклад діаграми послідовності з лініями життя об'єктів

Повідомлення – специфікація передачі інформації між об'єктами у розрахунку на те, що за цим послідує деяка діяльність.

Повідомлення можуть мати власне позначення операції. У цьому разі після імені операції записуються круглі дужки, в яких вказуються параметри операції. Якщо параметрів немає – записуються пусті дужки (наприклад: „встановити з’єднання між абонентами (a, b)”); „створити графік робіт ()”).

Діаграма кооперації є інформаційно тотожною діаграмі послідовності, але вона акцентує увагу на відношеннях між об’єктами та відображає порядок операцій за допомогою порядкових номерів (рис.21).

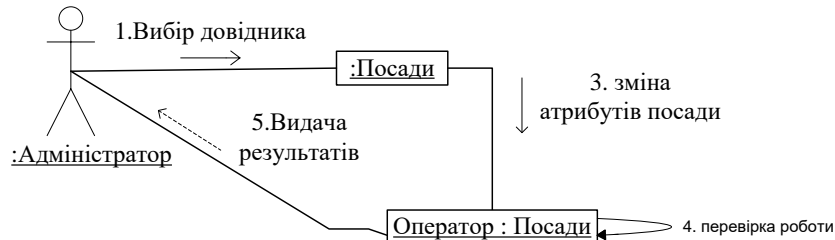


Рис. 21. Приклад діаграми кооперації

3. *Діаграми стану.* Діаграма стану (State diagram) показує стани ПЗ, що проектується, чи окремого об’єкта. Переходи між станами позначаються направленими дугами з назвами подій, що викликали зміни стану системи та обов’язково підписується у вигляді формули за таким синтаксисом:

Подія [Умова] / Дія ^Ім’яКласу.Ім’яПодії

У станах можна показати операції системи. Операції, які виконуються одразу після переходу до певного стану, називають вхідними (entry/). Внутрішні (do/) операції, які закінчуються до переходу в інший стан. Операції, які виконуються перед виходом із стану, називаються вихідними (exit/). Дія, яка активізується за певною умовою, позначається, як „event/” (рис. 22).

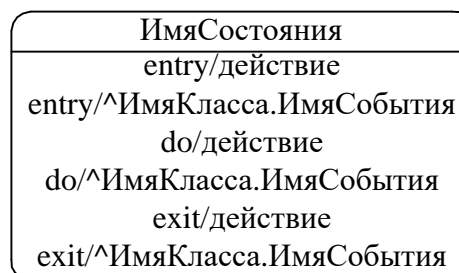


Рис. 22. Приклад дії, яка активується за певної умови

Приклад діаграми станів, який зображує стани системи при внесенні до неї співробітника, наведений на рис.23.

Деякі стани ПЗ дають можливість виходу з нього та повернення назад. Для повернення у той же стан, з якого здійснювався вихід, існує поняття історичного стану (history state). Він буває недавнім та давнім.

Недавній історичний стан використовується для запам’ятовування підстану, який є поточним на момент виходу системи із стану. При наступному вході системи в цей стан, вона повертається до підстану, який був поточним. Недавній історичний стан позначається як  $\textcircled{H}$ .

Давній історичний стан використовується для запам'ятовування усіх підстанів різних рівнів вкладеності. Давній історичний стан позначається як  $\odot$ .

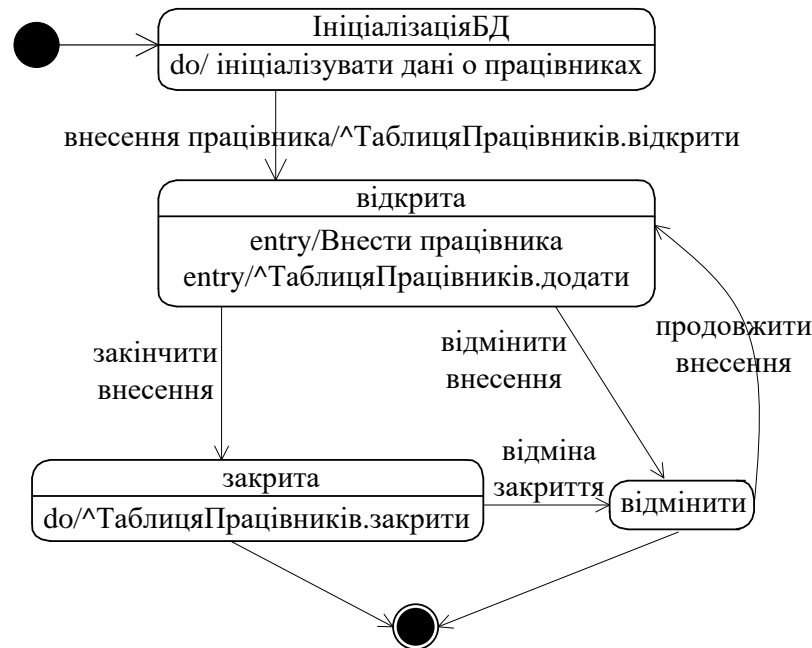


Рис. 23. Приклад діаграми станів

### Завдання

1. Побудувати діаграму діяльності для свого варіанта.
2. Специфікувати діаграму діяльності (див.табл.10).
3. Побудувати діаграми послідовності для свого варіанту.
4. Побудувати діаграми кооперації для свого варіанту.
5. Провести кількісну оцінку якості діаграм послідовності та кооперації.
6. Побудувати діаграму стану.
7. Побудувати діаграму стану з використанням історичного стану.
8. Провести кількісну оцінку якості діаграми стану.

**ЛІТЕРАТУРА:** [3, 5, 7-8].

## Лабораторна робота 8

### ПРОЕКТУВАННЯ АРХІТЕКТУРИ ПЗ

**Мета** – набути навичок у проектуванні архітектури програмного забезпечення за допомогою компонентів, з'єднувачів та даних.

### Теоретичні відомості

#### 1. Компоненти системи.

Архітектура ПЗ – це представлення ПЗ, за допомогою базових елементів трьох типів: компонентів, з'єднувачів та даних. Проектування архітектури включає в себе визначення її базових елементів та зв'язків між ними.

Діаграма компонентів (Component diagram) описує фізичне представлення системи та забезпечує перехід від логічного представлення до реалізації

проекту в формі програмного коду. Компонент є частиною фізичної реалізації системи (наприклад програмний модуль або інтерфейс користувача), який інкапсулює певний набір функціональних можливостей. З'єднувачі здійснюють взаємодію між компонентами. Компоненту дається ім'я, яке може складатися з будь-якої кількості букв та цифр (рис. 24).

Стереотипи компонентів такі: база даних (DB); модуль, що виконується (.exe); динамічна бібліотека (.dll); Web-сторінка (.html); файл (.h, .cpp, .java, .asp та ін.); документ (.txt, .doc, .hlp).

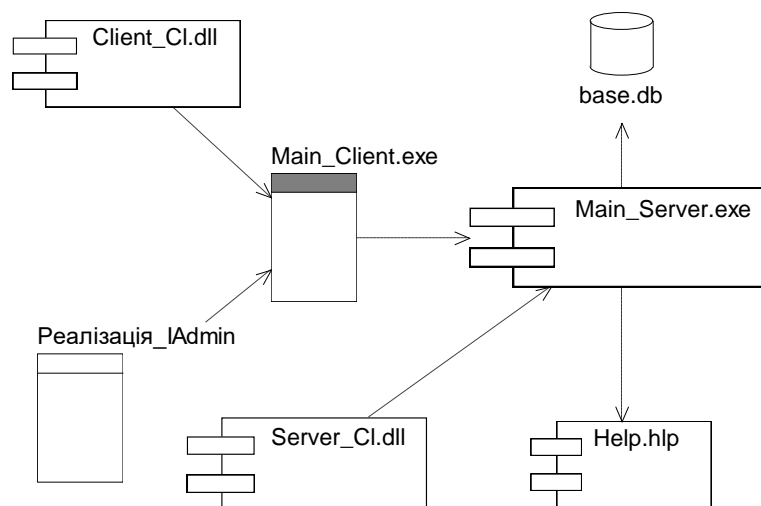


Рис. 24. Приклад діаграми компонентів

## 2. Розміщення системи.

Діаграма розгортання (Deployment diagram) відображає загальну конфігурацію і топологію системи, фізичний взаємозв'язок між програмними та апаратними компонентами (рис. 25).

Вузол – це фізичний об'єкт. Він може бути обчислювальним ресурсом (який має пам'ять, процесор і т. ін.), ресурсом механічної обробки даних, або людським ресурсом. Вузли можуть мати ємність, потужність, надійність, пропускну здатність.

Зв'язки між вузлами показують канали, за допомогою яких відбуваються комунікаційні з'єднання.

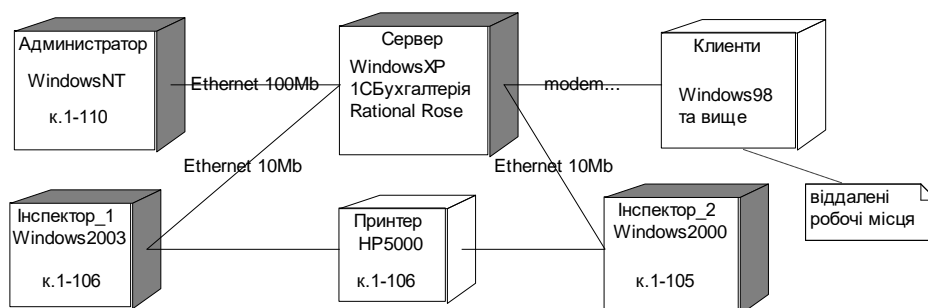


Рис. 25. Приклад діаграми розгортання

## Завдання

1. Побудувати діаграму компонентів для свого варіанта.
2. Провести кількісну оцінку якості діаграми компонентів.



3. Виявити вузли ПЗ, що проектується, та описати їх. Виявити зв'язки між вузлами. Побудувати діаграму розгортання.

4. Провести кількісну оцінку діаграми розгортання.

**ЛІТЕРАТУРА: [1-8].**

## **Лабораторна робота 9**

### **ДЕТАЛЬНЕ ПРОЕКТУВАННЯ**

**Мета** – набути навичок у проектуванні інтерфейсу користувача та баз даних.

### **Теоретичні відомості**

1. *Проектування інтерфейсу користувача.* Інтерфейс користувача забезпечує взаємодію з ПЗ. До сучасного інтерфейсу користувача висуваються такі вимоги:

- відповідність прийнятим стандартам у сфері програмування;
- терпимість до помилок (виявлення помилкових дій користувача та надання можливості їх виправлення);
- інформативність (наявність інтуїтивно зрозумілих візуальних елементів, допомоги, контекстних підказок);
- естетичність та зручність у користуванні.

Віконний інтерфейс може складатися з головного та дочірніх вікон. Головне вікно ініціюється на початку роботи, а дочірні вікна відкриваються через елементи головного.

Залежність між вікнами, як правило, має три форми:

- **документ/представлення** - містить тільки текстові дані;
- **sdi - інтерфейс** - складається з єдиного головного вікна;
- **mdi - інтерфейс** – складається з батьківського вікна та дочірніх вікон, які відкриваються з нього.

На рис. 26 наведено приклад інтерфейсу користувача та схема його взаємодії з користувачем та об'єктами ПЗ. Ця схема відображає процеси запиту до об'єкта **Замовлення**, виконання необхідних обчислень і передачі інформації до **ЖурналуПродажу**.

На основі виявлення усіх цих зв'язків надалі проектується повний інтерфейс користувача. Один з можливих підходів до специфікування інтерфейсу користувача – це його представлення у вигляді станів, які відображають візуальні елементи, та дій, що відображають функції, пов'язані з цими елементами – діаграми навігації по вікнам (рис. 27).

Перелік стереотипів для GUI-інтерфейсу Microsoft Windows:

- Стани (вікна):
  - головне вікно (панель у головному вікні, вікно перегляду строк, вікно перегляду дерев, Web- сторінка);
  - дочірнє вікно (діалогове вікно, вікно повідомлень, папка зі вкладками);
  - типи даних вікон (текстове поле, комбіноване вікно, колонка, строчка, група полів).

➤ Види діяльності (елемент управління вікном) - пункт випадаючого меню; пункт впливаючого меню; кнопка панелі інструментів; командна кнопка; двійний клік мишею; вибір зі списку; клавіша клавіатури; функціональна клавіша клавіатури; комбінація клавіш клавіатури; бігунок полоси прокрутки; кнопка закриття вікна.

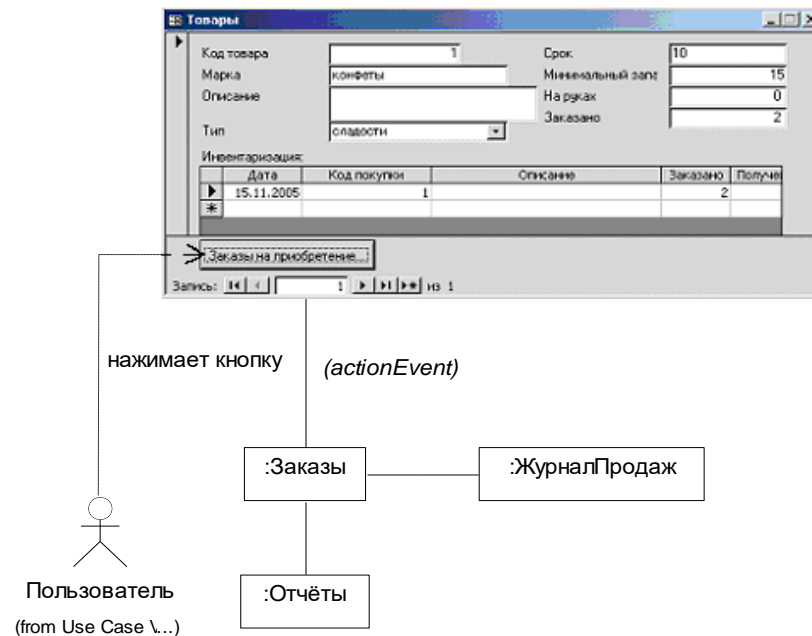


Рис. 26. Пример интерфейса пользователя та схема його взаємодії з користувачем та об'єктами ПЗ

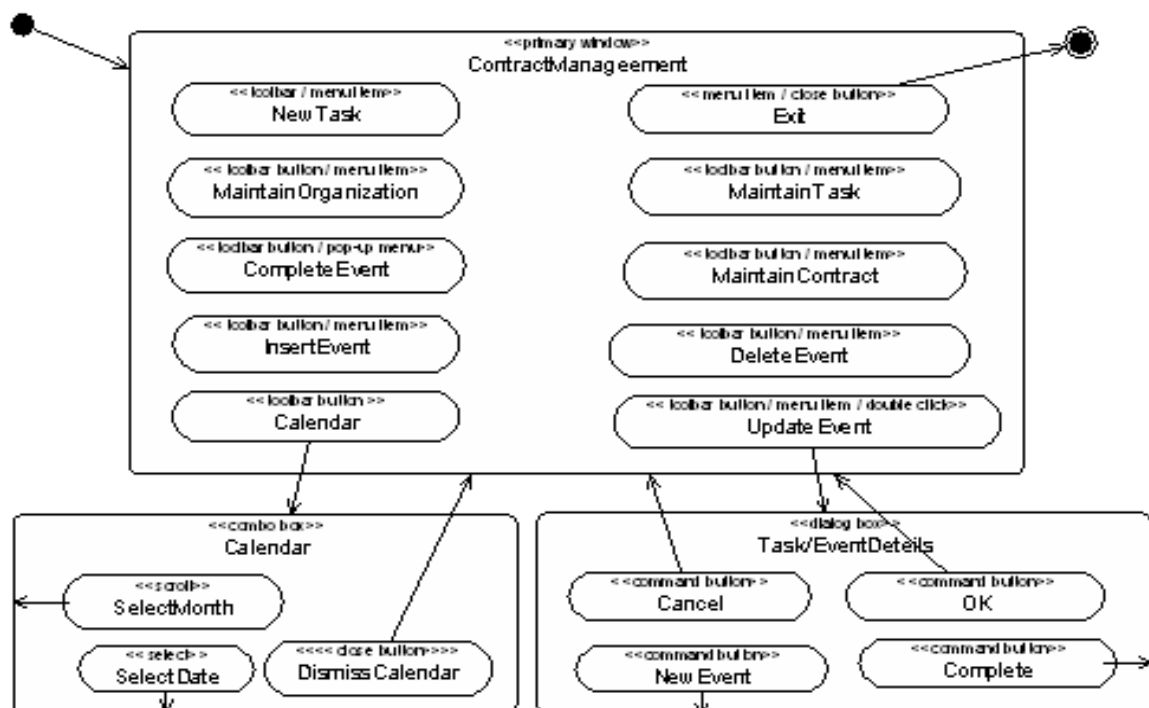


Рис. 27. Пример діаграми навігації по вікнах

2. *Проектування баз даних.* При проектуванні БД здійснюється перехід від загального опису предметної галузі до схеми фізичної БД шляхом поступової деталізації, яка включає такі етапи:

– виділення сутностей предметної галузі, їх основних атрибутів та зв’язків між ними, представлення їх у вигляді діаграми “сутність-зв’язок” або діаграми класів UML (рис.28);

– специфікування схеми логічної БД у вигляді моделі реляційної бази даних;

– специфікування схеми фізичної БД з урахуванням особливостей реалізації у обраної СУБД.

Перехід до реляційної моделі передбачає нормалізацію, вибір необхідних ключів, уточнення зв’язків та визначення цілісності зв’язків (посилочної цілісності). Специфікація структури фізичної БД є основою для генерації файлів БД у форматі обраної СУБД. Проектування та генерація БД може виконуватися за допомогою CASE-засобів.

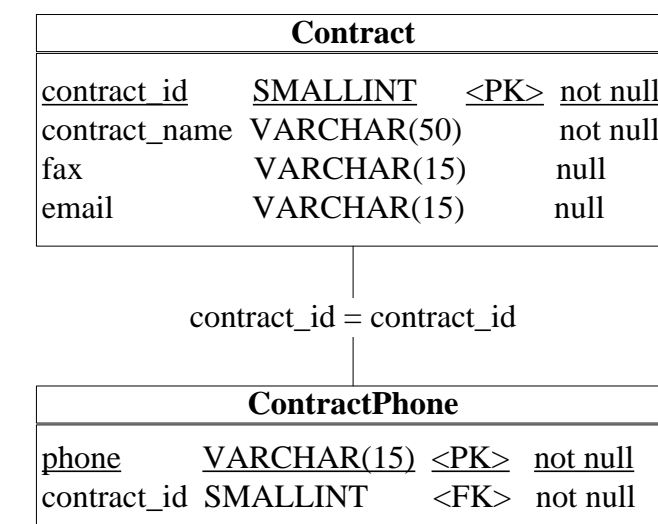


Рис. 28. Приклад діаграми класів UML

### Завдання

1.Спроекувати інтерфейс користувача системи, використовуючи діаграму стану.

2.Описати зв’язки між предметною областю та інтерфейсом користувача, що був спроектований.

3.Використовуючи будь-які засоби дизайну (наприклад, конструктори Access або Delphi), створити прототипи 2-3 графічних форм інтерфейсу користувача, який був спроектований.

4.На основі представлення предметної галузі, отриманого у попередніх роботах, побудувати діаграму „сутність-зв’язок” або діаграму класів БД.

5. Побудувати схеми логічної та фізичної БД.

**ЛІТЕРАТУРА: [1-8].**

## Лабораторна робота 10

### ТЕСТУВАННЯ ТА УПРАВЛІННЯ ЗМІНАМИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

**Мета** – набути навичок в організації тестування та управлінні змінами програмного забезпечення.

#### Теоретичні відомості

1. *Організація тестування ПЗ.* Загальний процес тестування ПЗ визначається стандартом ДСТУ ISO/IEC 12119:2003. Цей стандарт встановлює в якості мінімального базового елемента тестування тестовий варіант (табл.11), який є документованою інструкцією, що містить мету тесту, функції що тестуються, середовище тестування, дані, процедуру та очікувані результати тесту. Визначена сукупність тестових варіантів формує план тестування. Тестові дії передбачають виконання тестів в тестовому середовищі відповідно до плану тестування та отримання результатів тестування, в тому числі і негативних, тобто таких, що не відповідають очікуванім. Результати тестування оформлюються як тестові записи. Спираючись на результати тестування експерти складають звіт про тестування, який містить ідентифікацію ПЗ, конфігурацію апаратного та програмного забезпечення, перелік документів, які використані при тестуванні, результати тестування, список невідповідностей вимогам та дату проведення тестування.

Фази процесу тестування:

1. Визначення цілей (вимог до тестування), що включає: які частини системи будуть тестуватися, які аспекти роботи будуть вибрані для перевірки, яка якість і т.ін.
2. Планування: створення плану тестування (який вміщує графік розробки тестів для кожної підсистеми, що тестується; оцінку необхідних трудових, програмних та апаратних ресурсів; визначення метрик, які необхідно зібрати та проаналізувати).
3. Розробка тестів (тестових варіантів).
4. Виконання тестів.
5. Аналіз результатів.

Таблиця 11

Специфікація тестового варіанта	
<b>Назва взаємодіючих класів:</b> TCommandQueue, TCommand	<b>Назва тесту:</b> TCommandQueueTest1
<b>Опис тесту:</b> тест перевіряє можливість створення об'єкта типу TCommand та додавання його в чергу при визові методу AddCommand	
<b>Початкові умови:</b> черга команд пуста	
<b>Очікуваний результат:</b> у чергу буде додана одна команда	
<b>Отриманий результат:</b> в чергу додана команда	

2. *Управління змінами ПЗ.* В процесі розробки та тестування ПЗ здійснюється його змінення з метою досягнення заданих вимог та усунення виявлених дефектів. При достатньо великій кількості змін та колективної роботі

над проектом особливе значення набуває задача управління змінами, яка передбачає фіксацію необхідних запланованих змін у ПЗ та відстеження їх реалізації співробітниками. Для цього використовуються CASE-засоби, які включають спеціальну БД змін, форми-звіти про реалізовані зміни, які заповнюють співробітники та засоби контролю та аналізу змін керівником проекту. Під час роботи співробітники заповнюють форми описів виявлених дефектів та відмічають реалізацію змін через форми-звіти, які заносяться в базу дефектів. Приклад форми опису дефектів наведено у табл. 12.

Таблиця 12

№ п/п	Значення	Опис
1	Найменування підсистеми (місце знаходження дефекту)	
2	Версія продукту	
3	Опис дефекту	
4	Опис процедури знаходження	
5	Номер тесту	
6	Рівень дефекту (ступень його серйозності з точки зору критеріїв якості продукту або замовника)	
7	Дата знаходження	
8	ПІБ, того хто знайшов дефект	
9	Терміни виправлення	
10	Стан дефекту	

Занесений в базу новий дефект знаходиться в стані "New". Після аналізу дефект переводиться в стан "Open" та призначається конкретний відповідальний за виправлення. Після виправлення дефект переводиться у стан "Resolved". При цьому заповнюється дані, що наведені у таблиці 13.

Після цього стан переводиться у "Verified", якщо виправлення дійсні. У протилежному випадку, якщо факт виправлення не підтверджується, то стан дефекту змінюється на "Open". Якщо приймається рішення не виправляти у подальшому дефект, то його стан позначається як "Postponed" та вказується причини рішення.

Таблиця 13

№ п/п	Значення	Опис
1	Причина виникнення	
2	Місце виправлення	
3	Опис виправлення	
4	Час, який пішов на виправлення.	
5	Відповідальний за виправлення	

### Завдання

1. Створити сценарій тестування згідно варіанту наданого викладачем (використовуючи State diagram).
2. Скласти план тестування свого варіанту.

3. На основі варіантів використання системи та інтерфейсу, розробленому в лабораторній роботі 5 скласти тестові варіанти для тестування інтерфейсу користувача.

4. Розробити набір тестових даних для наступних компонентів (використовуючи Activity diagram):

- програма сортування масивів цілих чисел;
- програма, яка вираховує кількість символів (відмінних від пропусків) в текстових строках;
- програма, яка перевіряє текстові строки і замінює послідовності пропусків одним пропуском, а там де один пропуск – змінює його на символ %.

5. Написати код для вищеперерахованих програм, використовуючи будь-яку мову програмування та зробити їх завершеним програмним продуктом.

**ЛІТЕРАТУРА: [1-8].**

## **Лабораторна робота 11**

### **ГЕНЕРАЦІЯ ВХІДНИХ КОДІВ ПЗ. ВИКОРИСТАННЯ ЗВОРОТНОЇ ІНЖЕНЕРІЇ ДЛЯ ВІДНОВЛЕННЯ СПЕЦИФІКАЦІЙ ПЗ**

**Мета** – набути навичок у створенні графіку робіт проекту, плануванні його ресурсів, аналізі ризиків та створенні і конструюванні звітів.

#### **Теоретичні відомості**

##### *1. Генерація вхідних кодів ПЗ*

Специфікації детального архітектурного проектування ПЗ є високорівневими представленнями, на основі яких програміст пише вхідні коди ПЗ на визначеній мові програмування. Цей процес може бути автоматизованим за допомогою спеціальних CASE-засобів, що дозволяє скоротити час на написання коду та контролювати його відповідність специфікаціям. Більшість сучасних CASE-засобів дозволяють генерувати коди, які описують класи, відповідно до специфікацій класів та зв'язків між ними.

Загальні етапи, які виконуються для генерації програмного коду у середовищі Rational Rose наступні:

- перевірка моделі на відсутність помилок.
- створення компонентів для реалізації класів.
- відображення класів на компоненти.
- вибір мови програмування.
- встановлення властивостей генерації програмного коду.
- вибір класу, компонента чи пакета.
- генерація програмного коду.

У загальному випадку перевірка моделі може виконуватися на будь-якому етапі роботи. Але після завершення розробки графічних діаграм вона обов'язкова.

Для перевірки моделі проводять: Tools → Check Model (Інструменти → Перевірити модель). Результати перевірки на наявність помилок відображаються у вікні журналу (рис. 29).



Рис. 29. Приклад результатів перевірки на наявність помилок

Для генерації використовується раніше створена діаграма компонентів. Для відображення класів на компоненти використовується вікно специфікації властивостей компонента (вкладка **Realizes** (Реалізує)). Для включення реалізації класу в компонент треба виділити клас на цій вкладці та виконати операцію контекстного меню **Assign** (Призначити) (рис.30).

Для вибору мови потрібно виконати операцію: **Tools** → **Options** (Інструменти → Параметри), у результаті чого викликається діалогове вікно настройки параметрів моделі, де на вкладці **Notation** (Нотація) потрібно вибрати мову генерації.

Далі потрібно змінити мову реалізації кожного компонента моделі. Для цього треба виставити потрібну мову на вкладці **General** (Загальні) вікна специфікації властивостей компонента.

Після вибору мови програмування потрібно привести у відповідність типи атрибутів, типи аргументів та типи виразів що повертають значення операцій. Тобто змінити ті типи, які не являються синтаксично прийнятними у вибраній мові програмування.

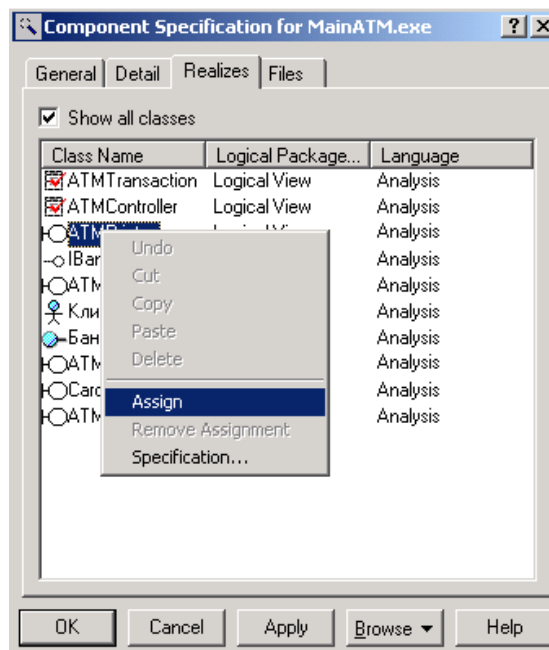


Рис. 30. Вікно специфікації властивостей компонента

Наприклад, у мові ANSI C++ треба змінити типи Integer на int, Boolean на bool, Currency на float. У протилежному випадку виправлення потрібно буде робити після генерації коду.

Для генерації коду потрібно вибрати меню **Генерація коду** (Generate Code). У вікні вибрати класи для генерації.

Після генерації програмного коду кожному класу, реалізованому в компоненті, буде відповідати файли: заголовочний з розширенням «h» та

реалізації з розширенням «срр». Заголовочний файл вміщує об'явленням всіх операцій та атрибутів класу. При цьому інформація про документування операцій та атрибутів розміщується у коментарі перед відповідними елементами програми. Файл реалізації вміщує заготовку для реалізації усіх операцій класу. При цьому кожна з операцій має пусте тіло реалізації, яке треба написати додатково.

## *2. Використання зворотної інженерії для відновлення специфікацій ПЗ*

Послідовний перехід від аналізу предметної області до високорівневих специфікацій проекту, вхідних кодів програм та кодів, що виконуються реалізується в рамках процесів прямої інженерії ПЗ. В процесах супроводження та оцінки ПЗ виникає необхідність відновлення специфікацій проекту (високорівневих представлень) з кодів програм, фізичних БД та інших програмних ресурсів (наприклад, при доробці існуючих програмних систем). Така задача реалізується в рамках зворотної інженерії ПЗ і також підтримується засобами CASE.

Етапи проведення зворотної інженерії:

1. Створення проекту.

2. Опис проекту.

Проект аналізатора, так само як модель і вихідний програмний код, підлягає документуванню. Кожен проект повинен бути забезпечений відповідним змістовним описом, що включає інформацію про найменування і призначення. Подібні відомості можуть виявитися корисними для інших розробників, які намагаються з'ясувати, чи володіє проект потенціалом повторного застосування в якості самостійного або базового рішення.

3. Включення бібліотек і базових проектів.

Найменування каталогів, що переглядаються аналізатором, заносяться в спеціальний список. У список обов'язково включаються каталоги з файлами, які підлягають аналізу або містять додаткову інформацію, яка використовується в процесі аналізу.

4. Аналіз файлів проекту.

Аналізатор розподіляє файли, що підлягають обробці, за трьома категоріями - Type1, Type2, Type3. Файл, який додається до списку, відноситься до категорії Type1: файл синтаксично сповнений і контекстно-незалежний, тобто представляє безліч вичерпних C ++ оголошень і містить всю необхідну інформацію. Файл категорії Type2 синтаксично наповнений, але контекстно-залежний; іншими словами, він так само представляє C ++ оголошення, але містить символи, визначення яких обумовлені контекстом. Файл категорії Type3 не є синтаксично повним; система робить спробу обробити такий файл в кожному сеансі аналізу.

Аналізатор C ++ здатний обробляти єдиний файл або групу файлів. Для кожного файлу створюється спеціальний файл даних з результатами аналізу. Статус кожного з переглядаються файлів відображається у вікні проекту:

Stale Data - дані аналізу файлу, можливо, застаріли.

Analyzed - аналіз файлу завершено успішно (для файлів категорії Type1, Type2)

No Source - файл не знайдений.



Has Errors - в процесі аналізу виявлено помилки.

#### 5. Робота над помилками.

Аналізатор виводить інформацію про помилки в вікно протоколу. Кожна помилка підлягає розбору з урахуванням ступеня її серйозності. Найбільш типові помилки:

Unresolved references - аналізатор не в змозі знайти адресовані файли. Слід поповнити список каталогів проекту каталогами, що містять такі файли.

Missing language extensions - існують розширення мови, не розпізнані аналізатором. Слід визначити подібні розширення як дійсні символи препроцесора.

Context-sensitive source files - адресується код з іншого каталогу, але в поточному файлі відсутні необхідні директиви включення. Необхідно віднести файл до однієї з категорій - Type2 або Type3.

#### 6. Експорт результатів.

Опції експорту визначають, які сутності коду (наприклад, класи, коментарі, асоціації) - підлягають відображенню в підсумковому файлі і включенню в модель Rational Rose.

#### 7. Оновлення моделі.

Дані файлу .red використовуються для поновлення моделі шляхом заміщення її окремих елементів.

### **Завдання**

1. Використовуючі CASE-засіб побудувати специфікацію модулю (компоненту) ПЗ, на основі діаграми класів, яка була побудована у попередніх роботах.

2. Здійснити генерацію вхідного коду ПЗ для обраного модулю на визначеній мові програмування.

3. Дослідити отриманий вхідний код ПЗ та співставити його зі специфікаціями модулю та класів.

4. На основі обраних кодів програм, використовуючі CASE-засіб, здійснити зворотне відновлення специфікацій ПЗ у вигляді UML-моделі. Дослідити відповідність отриманих специфікацій кодам програм.

**ЛІТЕРАТУРА: [1-8].**

## ДОМАШНІ ЗАНЯТТЯ

### 1. Постановка завдання

Метою виконання домашньої роботи є отримання навиків у використанні методів та засобів розробки програмного забезпечення (ПЗ) автоматизованих систем. Під час виконання домашньої роботи студент працює в групі (до 4-х осіб) та проходить усі фази життєвого циклу ПЗ, а також презентує результати своєї діяльності.

### 2. Зміст пояснювальної записки проекту

Домашня робота повинна включати наступні розділи:

#### 2.1. Зміст

#### 2.2. Вступ

#### 2.3. Завдання (варіант).

Завдання містить опис предметної галузі, для якої повинно виконатися проектування ПЗ. Завдання обирається за узгодженням з викладачем. Також треба вказати обрану модель життєвого циклу розробки ПЗ, пояснити її переваги над іншими та вказати доцільність використання в обраному проекті.

#### 2.4. Стислий опис методів та засобів, які застосовуються при проектуванні.

У цьому розділі необхідно відобразити базові поняття про обрані методи та засоби проектування (методи об'єктного аналізу, візуальні мови моделювання та специфікування та т.ін.)

#### 2.5. Специфікація проекту.

Специфікація проекту повинна містити вербальні описи та графічні специфікації (діаграм), які розкривають ті чи інші аспекти предметної галузі, реалізації розроблюваного ПЗ та процесу проектування та розробки.

Специфікація проекту повинна містити наступне:

#### 1. Встановлення вимог:

- документ опису вимог (технічне завдання), який повинен вміщувати:
  - діаграми варіантів використання;
  - календарне планування та планування ресурсів у Microsoft Project;
  - робочу та організаційну структури (WBS, OBS) та двонаправлену модель управління проектом;
- вербальні специфікації прецедентів.

#### 2. Специфікування вимог:

- Class diagram (діаграма класів);
- Package diagrams (діаграма пакетів).

#### 3. Моделювання поведінки системи:

- Activity diagram (діаграма діяльності);
- Interaction diagrams (діаграми взаємодії);
- State diagram (діаграма стану).

#### 4. Проектування архітектури ПЗ:

- Component diagram (діаграма компонентів);
- Deployment diagram (діаграма розгортання).

#### 5. Детальне проектування:

- проектування інтерфейсу користувача;
- проектування баз даних.

## **2.6. Генерація програмного коду**

У цьому розділі необхідно провести генерацію програмного коду згідно загальних етапів на мові програмування C++ у середовищі IBM Rational Rose.

## **2.7. Висновок**

Розділ повинен містити стислий висновок про зміст та результати проектування.

## **2.8 Список літератури**

## **3.Оформлення та захист домашньої роботи.**

Пояснювальна записка оформлюється на аркушах формату А4 у відповідності з діючими стандартами. Титульний аркуш повинен містити тему роботи, назву дисципліни, яка вивчається, прізвище, ім'я та по батькові студента і номер групи. Всі аркуші повинні бути пронумеровані. Номер розміщується в правому нижньому куту аркушу (окрім титульного). Всі малюнки і таблиці повинні бути пронумеровані. Об'єм записки 20 - 25 аркушів.

**До захисту роботи допускаються студенти, які виконали завдання відповідно до варіанту, представили правильно оформлену пояснювальну записку, мають діючий прототип та підготували презентацію та доповідь**

**(пояснювальна записка + презентація зберігаються на CD або флешпам'яті)!**

**Критерієм оцінювання роботи є:**

- повнота і правильність рішення задачі,
- знання методів та інструментальних засобів, використаних для проектування і розробки
- якість презентації продукту;
- відповіді на питання.

## **Додаток 1. Варіанти завдань для виконання лабораторної роботи 1.**

1. *Розрахувати за базовим рівнем моделі COSOMO трудовитрати (E) і визначити час розробки (TDEV). Визначити середню чисельність персоналу (SS) і рівень продуктивності (P), якщо:*

- Варіант 1. розмір проекту, який розроблюється, оцінюється в 10 KLOC.
- Варіант 2. розмір проекту, який розроблюється, оцінюється в 300 KLOC.
- Варіант 3. розмір проекту, який розроблюється, оцінюється в 50 KLOC.
- Варіант 4. розмір проекту, який розроблюється, оцінюється в 55 KLOC.
- Варіант 5. розмір проекту, який розроблюється, оцінюється в 320 KLOC.
- Варіант 6. розмір проекту, який розроблюється, оцінюється в 25 KLOC.
- Варіант 7. розмір проекту, який розроблюється, оцінюється в 72 KLOC.
- Варіант 8. розмір проекту, який розроблюється, оцінюється в 85 KLOC.
- Варіант 9. розмір проекту, який розроблюється, оцінюється в 400 KLOC.
- Варіант 10. розмір проекту, який розроблюється, оцінюється в 7,5 KLOC.

2. *Визначити режим складності системи за проміжним рівнем моделі COSOMO, якщо:*

Варіант 1. розмір проекту за першим завданням відповідно варіанту; значення множників (драйверів) витрат ACAP, PCAP, TIME, DATA, PLEX змінюються до високих, всі інші значення номінальні.

Варіант 2. розмір проекту за першим завданням відповідно варіанту; значення множників (драйверів) витрат RELY, DATA, PVOL, PCAP, змінюються до низьких, всі інші значення номінальні.

Варіант 3. розмір проекту за першим завданням відповідно варіанту; значення множників (драйверів) витрат ACAP, CPLX змінюються до високих, TIME, DATA, PLEX змінюються до низьких, всі інші значення номінальні.

Варіант 4. розмір проекту за першим завданням відповідно варіанту; значення множників (драйверів) витрат TIME, PLEX, CPLX, змінюються до дуже високі, всі інші значення номінальні.

Варіант 5. розмір проекту за першим завданням відповідно варіанту; значення множників (драйверів) витрат TOOL, SCED змінюються до низьких, PLEX, STOR змінюються до дуже високі, всі інші значення номінальні.

Варіант 6. розмір проекту за першим завданням відповідно варіанту; значення множників (драйверів) витрат CPLX, STOR, DOCU, PCAP змінюються до дуже високих, всі інші значення номінальні.

Варіант 7. розмір проекту за першим завданням відповідно варіанту; значення множників (драйверів) витрат ACAP, APEX, PCAP, PLEX змінюються до низьких, всі інші значення номінальні.

Варіант 8. розмір проекту за першим завданням відповідно варіанту; значення множників (драйверів) витрат CPLX, SCED змінюються до дуже низькі, ACAP змінюються до низьких, всі інші значення номінальні.

Варіант 9. розмір проекту за першим завданням відповідно варіанту; значення множників (драйверів) витрат RELY, DATA, ACAP, PCAP, STOR змінюються до низькі, всі інші значення номінальні.

Варіант 10. розмір проекту за першим завданням відповідно варіанту; значення множників (драйверів) витрат SITE, TOOL змінюються до дуже низькі, SCED змінюються до низьких, всі інші значення номінальні.

*Примітка:* Показники множників (драйверів) витрат знаходяться в таблиці 1.

3. Оцінити трудовитрати, тривалість і середню чисельність персоналу проекту по моделі COSOMO II (для попередньої оцінки). Значення  $S$  згідно варіанту завдання 1. Показник  $R_j$  середній рівень (таблиця 4),  $Z_i$  – високий рівень (таблиця 5).

Таблиця 1

Значення драйверів витрат при розробці ПЗ в рамках моделі COSOMO

Параметри вартості	Показники					
	Дуже низький	Низький	Номіна- льний	Висо- кий	Дуже високий	Над- високий
Атрибути продукту						
Необхідна надійність ПЗ (RELY)	0,82	0,92	1,00	1,10	1,26	
Розмір бази даних (DATA)		0,90	1,00	1,14	1,28	
Складність програмного продукту (CPLX)	0,73	0,87	1,00	1,17	1,34	1,74
Необхідний рівень повторного використання (RUSE)		0,95	1,00	1,0,7	1,15	1,24
Документація (DOCU)	0,81	0,91	1,00	1,11	1,23	
Атрибути персоналу						
Здібності аналітика (ACAP)	1,42	1,19	1,00	0,85	0,71	
Досвід створення додатків (APEX)	1,22	1,10	1,00	0,88	0,81	
Здібності програміста (PCAP)	1,34	1,15	1,00	0,88	0,70	
Досвід в області віртуальних машин (PLEX)	1,19	1,09	1	0,91	0,85	
Досвід в області мов програмування (LTEX)	1,20	1,09	1	0,91	0,84	
Наступність персоналу (PCON)	1,29	1,12	1	0,90	0,81	

Атрибути проекту						
Використання практики сучасного програмування (SITE)	1,22	1,09	1,00	0,93	0,86	0,80
Сучасні інструменти програмування (TOOL)	1,17	1,09	1,00	0,90	0,78	
Необхідний графік розробки (SCED)	1,43	1,14	1,00	1,00	1,00	
Атрибути платформи						
Обмеження часу виконання (TIME)			1,00	1,11	1,29	1,63
Обмеження головного сховища (STOR)			1,00	1,05	1,17	1,46
Змінність платформи (PVOL)		0,87	1,00	1,15	1,30	

Таблиця 2

*Класифікатори проекту створення інформаційної системи*

<b><math>K_1</math> - масштаб об'єкту автоматизації</b>	<b><math>K_2</math> - тип замовника</b>	<b><math>K_3</math> - тип програмного забезпечення</b>
Автоматизація бізнес процесу одного структурного підрозділу – 1	Місцевий виконавчий орган – 8	Готове програмне забезпечення, яке потребує налаштування – 1
Автоматизація бізнес-процесів одного відомства – 8	Центральний державний орган – 14	База даних – 6
Автоматизація бізнес-процесів одного відомства з територіальними підрозділами – 9	Державний орган, діяльність якого пов'язана з безпекою для життя – 15	Клієнт – серверне (товстий клієнт) – 8
Автоматизація бізнес-процесів відомства та інтеграція із зовнішніми інформаційними системами – 10	Приватний замовник – 5	Клієнт – серверне (тонкий клієнт) – 11
Автоматизація бізнес-процесів декількох відомств – 12		Сервіс – орієнтоване – 15
Автоматизація бізнес-процесів декількох відомств та інтеграція із зовнішніми інформаційними системами – 13		

Таблиця 3

Коефіцієнт переводу балу функціональності в кількість логічних рядків коду

Мова програмування	КП (кількість логічних рядків коду )	Мова програмування	КП (кількість логічних рядків коду )
Basic Assembler	320	.NET	60
Autocoder	320	JSP	59
Netron/CAP	296	LOGO	58
Macro Assembler	213	C#	58
C	128	J2EE	57
Пакетні файли DOS	128	Розширений LISP	56
Basic	107	RPG III	56
Макроси Lotus	107	ASP	56
ALGOL	105	Java	55
COBOL	105	JavaScript	54
FORTRAN	105	C++	53
JOVIAL	105	YACC	53
Змішані мови програмування	105	Culprit	51
JCL	96	Natural	51
VPF	95	KML	50
Pascal	91	Visual Basic	50
COBOL (ANSI 85)	91	REXX	50
APS	86	Ada 95	49
Slogan	81	PL/SQL	47
RPG	80	CICS	46
Modula-2	80	SIMULA	46
PL/1	80	Taskmate	45
Паралельний Pascal	80	Focus	45
Fortran 95	71	Web Scripts	44
Mantis	71	Pacbase	42
Sabretalk	70	Мови баз даних	40
Mapper	69	Clipper DB и dBase III	40
ColdFusion	68	Informix	40
Datastage	67	Oracle и SYBASE	40
Ideal	66	Openroad	39
Basic (ANSI)	64	Access	38
FORTH	64	VBScript	38
LISP	64	Advantage	38
PROLOG	64	PeopleSoft	37
Powerhouse	63	Cool:Gen/IEF	37
Uniface	61	DBase IV	36

<b>Мова програмування</b>	<b>КП (кількість логічних рядків коду )</b>	<b>Мова програмування</b>	<b>КП (кількість логічних рядків коду )</b>
Мови підтримки прийняття рішень	35	Стандартні мови 4-го покоління (4GL)	20
FoxPro 2.5	34	OR3 (4GL)	20
APL	32	Application Builder	20
Статичні мови (SAS)	32	CORBA	20
Maestro	30	Cristal Reports	20
DELPHI	29	Datatrieve	20
Стандартні об'єктно-орієнтовані мови	29	CLIPPER	19
Powerbuilder	28	ABAP (SAP)	18
VB.Net	28	HTML 3.0	15
OBJECTIVE-C	27	Siebel Tools	13
Lotus Script	23	SQL	13
Oracle Developer /2000	23	Easytrieve+	13
Smalltalk	21	SQL Forms	11
awk	21	Excel	6
EIFFEL	21	QUATTRO PRO	6
Shell-сценарії (Perl)	21	Мови створення піктограм	4

Таблиця 4

*Значення показників розробки*

<b>Показники розробки, R<sub>j</sub></b>	<b>Низький рівень</b>	<b>Середній рівень</b>	<b>Високий рівень</b>
передбачуваність проекту для розроблювача, R <sub>1</sub>	4,96	3,72	2,48
гнучкість процесу розробки, R <sub>2</sub>	4,05	3,04	2,03
ступінь знищення ризиків, R <sub>3</sub>	5,65	4,24	2,83
згуртованість команди проекту, R <sub>4</sub>	4,38	3,29	2,19
зрілість процесів в організації розроблювача, R <sub>5</sub>	6,24	4,68	3,12



Таблиця 5

## Значення множників витрат

<b>Множники витрат, <math>Z_i</math></b>	<b>Низький рівень</b>	<b>Середній рівень</b>	<b>Високий рівень</b>
згуртованість персоналу, $Z_1$	1,20	1,00	0,83
надійність і складність прикладного програмного	0,83	1,00	1,33
складність платформи, $Z_3$	0,87	1,00	1,29
необхідний рівень повторного використання, $Z_4$	0,95	1,00	1,07
досвідченість персоналу, $Z_5$	1,22	1,00	0,87
використання інструментів, $Z_6$	1,10	1,00	0,87
щільність графіка проекту, $Z_7$	1,14	1,00	1,00

## **Додаток 2. ЗРАЗОК ВАРІАНТІВ ЗАВДАНЬ ДЛЯ ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ 2-11**

1. Автоматизована інформаційна система деканату ВНЗ.
2. Автоматизована система приймальної комісії ВНЗ.
3. Автоматизована система складання розкладу занять ВНЗ.
4. Автоматизована система дистанційного навчання у ВНЗ.
5. Автоматизована система бібліотеки ВНЗ.
6. Автоматизована система складського обліку організації.
7. Автоматизована система кадрового обліку організації.
8. Автоматизована система кур'єрської поштової служби.
9. Автоматизована система аптечної мережі міста.
10. Автоматизована система магазину DVD-фільмів.
11. Автоматизована система «Виклик таксі».
12. Автоматизована система обліку транспортних засобів у ДАІ.
13. Автоматизована система обліку тварин у зоопарку.
14. Автоматизована система Інтернет-магазину.
15. Словник-перекладач.
16. Автоматизована система кадрового обліку працівників підприємства
17. Автоматизована система продажу квитків.
18. Автоматизована система довідкової служби
19. Автоматизована система обліку робочого часу працівників підприємства
20. Автоматизована система бібліотеки.
21. Автоматизована система бюро працевлаштування.
22. Автоматизована система документообігу організації.
23. Автоматизована контрольно-пропускна система реєстрації працівників організації.
24. Автоматизована система житлоуправління з обліку мешканців та їх розрахунків.
25. Автоматизована система формування платежів телефонної компанії.
26. Автоматизована банківська система безготівкових електронних платежів.
27. Автоматизована система агентства з нерухомості.
28. Автоматизована система аптечної мережі міста.
29. Автоматизована система продаж торговельного центру (супермаркету).
30. Програмне забезпечення серверу електронної пошти.
31. Автоматизована система обліку пацієнтів лікарні.
32. Програмне забезпечення системи керування мобільним телефоном.
33. Автоматизована система перевірки руху маршрутів міського транспорту.
34. Автоматизована система продажу залізничних квитків.
35. Автоматизована система станції технічного обслуговування автомобілів.

## СПИСОК ЛІТЕРАТУРИ

1. АРЧИБАЛЬД Р. Управление высокотехнологичными программами и проектами. – М.: ДМК Пресс; Компания АйТи, 2006. – 472с.
2. БАТЕНКО Л. П., ЗАГОРОДНІХ О. А., ЛІЩИНСЬКА В. В. Управління проектами. – К.: КНЕУ, 2003. – 232с.
3. БУЧ Г., РАМБО Д., ЯКОБСОН И. Язык UML. Руководство пользователя. – М.: ДМК Пресс, 2007. – 496с.
4. ДСТУ ISO/IEC 12119:2003. Пакети програм. Тестування і вимоги до якості
5. КВАТРАНИ Т. Визуальное моделирование с помощью Rational Rose 2002 UML. – М.: Вильямс, 2003. – 192с.
6. МАКГРЕГОР Д., САЙКС Д. Тестирование объектно-ориентированного программного обеспечения. – М: Diasoft, 2002. – 220с.
7. СОММЕРВИЛ И. Инженерия программного обеспечения. – М.: Вильямс, 2002. – 624с.
8. ЯКОБСОН А, БУЧ. Г., РАМБО Дж. Унифицированный процесс разработки программного обеспечения. – СПб.: Питер, 2002. – 496с.

Навчально – методичне видання

## **ПРОФЕСІЙНА ПРАКТИКА ПРОГРАМНОЇ ІНЖЕНЕРІЇ**

Лабораторний практикум  
для студентів спеціальності  
121 «Інженерія програмного забезпечення»