Міністерство освіти і науки України
Чернівецький національний університет
імені Юрія Федьковича

ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Методичні вказівки до лабораторних робіт

(електронне видання)

Чернівці
Чернівецький національний університет ім. Ю. Федьковича
2022

Рекомендовано вченою радою
Навчально-наукового інституту
фізико-технічних та комп'ютерних наук
Чернівецького національного університету
імені Юрія Федьковича
(протокол № 9 від 27 жовтня 2022)

Інженерія програмного забезпечення : методичні вказівки до лабораторних робіт / укл.: Танасюк Ю.В, Одайська Х. С. Чернівці : Чернівецький національний університет ім. Ю. Федьковича, 2022. 146 с.

Дисципліна "Інженерія програмного забезпечення" входить до переліку обов'язкових для студентів спеціальності 123 "Комп'ютерна інженерія", що навчаються за освітньо-професійними програмами "Комп'ютерна інженерія" та "Програмування мобільних і вбудованих комп'ютерних систем та засобів інтернету речей". Методичні вказівки сприятимуть підготовці до виконання семи лабораторних робіт, які дадуть змогу практично засвоїти теоретичний матеріал дисципліни. Лабораторні роботи адаптовані до кредитно-модульної системи навчання та передбачають як індивідуальну так і командну роботу.

Для студентів вищих навчальних закладів.

ВСТУП

Дисципліна «Інженерія програмного забезпечення» має на меті формування у майбутніх інженерів сучасного рівня інформаційної та цифрової культури, оволодіння основними принципами створення програмних продуктів; набуття практичних навичок самостійного складання професійного програмного забезпечення і використання сучасних інформаційних технологій для розв'язання різноманітних задач прикладного характеру.

Мета навчальної дисципліни: забезпечити засвоєння студентами основних понять і методів системного керування процесом розроблення програмного забезпечення, оволодіння базовими принципами програмної інженерії; формування умінь створення технічного завдання, планування виконання робіт по проекту, набуття навичок прийняття рішень та розподілу обов'язків у команді і використання сучасних інформаційних технологій при створенні професійного програмного забезпечення.

Завдання дисципліни: формування теоретичних знань та практичних умінь у сфері розробки програмного забезпечення на всіх етапах життєвого циклу. У результаті вивчення навчальної дисципліни студенти повинні:

знати: основні поняття інженерії програмного забезпечення; підходи до управління процесом розробки програмного забезпечення; принципи архітектурного та об'єктно-орієнтованого проектування програмного забезпечення; основні типи інструментарію для розробки ПЗ; принципи та моделі розробки ПЗ, методології програмування; засоби управління вимогами до розробки ПЗ; основні методи забезпечення якості програмного забезпечення та тестування.

вміти: формулювати вимоги до програмного продукту; розв'язувати задачі з використанням декомпозиції; створювати діаграми різних типів; розробляти структуру проекту ПЗ; проектувати та реалізовувати зручний користувацький інтерфейс; оформляти документацію до програмного проекту; працювати з кількома версіями програмного проекту; виконувати різного роду тестування ПЗ; визначати техніко-економічні

показники програмного продукту; організовувати та підтримувати роботу в команді.

Даний навчальний посібник охоплює низку завдань до лабораторних робіт, передбачених силабусом навчальної дисципліни «Інженерія програмного забезпечення».

Лабораторний практикум складається з 7 лабораторних робіт та курсової роботи, які передбачають виконання практичних завдань на різних стадіях розробки програмного продукту. Кожна лабораторна робота містить з теоретичний матеріал, практичне завдання, рекомендації щодо виконання, вимог до оформлення та подання результатів і запитань для самоперевірки.

Лабораторна робота 1. «Розроблення специфікації вимог до програмного продукту» розглядає етапи розробки програмного продукту, вимоги та зміст технічного завдання. Лабораторна робота № 1 допоможе студентам набути навичок з розробки технічного завдання, яке буде корисним при наступних етапах розробки ПП.

Лабораторна робота 2. «Розробка ескізного проекту ПП. Створення діаграми прецедентів» ознайомлює із засадами візуального моделювання потреб користувачів та оформлення їх у графічному вигляді. Лабораторна робота № 2 дасть змогу розпочати проектування програмного продукту, визначити користувачів та їхню роль в системі, уточнити вимоги до програми, наводить рекомендації щодо розробки діаграми прецедентів та її реалізації для конкретного прикладу.

Лабораторна робота 3. «Взаємодія об'єктів. Створення діаграми послідовності» присвячена відображенню процесів обробки інформації у варіанті використання та взаємодії об'єктів у системі, що проектується. Лабораторна робота № 3 допоможе змоделювати/візуалізувати виконання операцій, їх взаємодію та стан об'єктів у час.

Лабораторна робота 4. «Проектування інтерфейсу користувача» знайомить з класифікацією прототипів та прикладами шаблонів користувацького інтерфейсу. Лабораторна робота № 4 допоможе розробити прототип майбутнього

інтерфейсу та відобразити основні функціональні можливості користувачів з різним рівнем доступу.

Лабораторна робота 5. «Створення діаграми класів» розглядає поняття класу, його атрибути, методи та стереотипи. Описані типи зв'язків між класами, їх взаємодія та візуалізація на діаграмі класів. Лабораторна робота № 5 формує кінцевий результат проекту програми і відправну точку процесу розробки.

Лабораторна робота 6. «Розрахунок техніко-економічних показників програмного продукту» надає інформацію про основні техніко-економічні показники, які необхідно врахувати при створенні програмного продукту. Лабораторна робота № 6 допоможе студентам набути навичок розрахунку основних параметрів, які слід враховувати у процесі розробки, як-от загальна тривалість робіт, собівартість, зарплата, ціна продукту.

Лабораторна робота 7. «Тестування програмного продукту» присвячена розробці тест-кейсів та створенню сценарію для проведення тестування програмного забезпечення. Лабораторна робота № 6 допоможе набути навичок планування і проведення тестування, а також оцінки його результатів.

Для виконання лабораторних робіт рекомендовано використовувати online або community версії середовищ моделювання як-от IBM Rational Rose, Visual Paradigm https://online.visual-paradigm.com, Drawio https://drawio-app.com Lucidchart https://www.lucidchart.com, diagrams.net .

Дисципліна «Інженерія програмного забезпечення» складається з двох модулів. Оцінювання навчальної діяльності студентів ґрунтується на результатах виконання та захисту лабораторних робіт, а також написанні модульних контрольних робіт.

Завдання, виконані під час лабораторних робіт, складатимуть частину курсового проекту.

Для досягнення цілей лабораторного практикуму студенти можуть працювати індивідуально або в складі команди з двох (в окремих випадках, трьох) осіб з чітким розподілом відповідальностей. Спосіб виконання, склад команди та варіант завдання (Додаток А) здобувачі освіти обирають самостійно. У

межах академічної групи обрані варіанти завдань не можуть повторюватися.

Викладачі, що проводять заняття з дисципліни, є зацікавленими особами, які оцінюватимуть результати роботи, а також надають консультації щодо уточнення вимог та робіт за проектом.

Рішення щодо мови, технологій та середовищ програмування, використаних у розробці, студенти ухвалюють самостійно. Інформаційна система, що розробляється, повинна забезпечувати централізоване зберігання даних, при цьому СУБД може бути як реляційною так і NoSQL, за вибором команди.

У ході розробки слід дотримуватися принципів візуального, об'єктно-орієнтованого та/або веб-програмування. Програмний продукт може набувати вигляду настільного, мобільного або веб-застосунку.

Результати розробки, які подаються учасниками команди під час захисту лабораторних робіт та курсового проекту, повинні бути оформлені документально відповідно до встановлених вимог (Додаток Б – Е).

Проміжні результати курсової роботи демонструються під час двох попередніх захистів. Оцінювання відбувається шляхом перехресної перевірки учасниками різних проектів. Критерії оцінювання результатів виконання курсової роботи наведено у додатках З, І, К.

Лабораторна робота № 1. Розроблення специфікації вимог до програмного продукту

Мета: набути навичок аналізу вимог до програмного продукту та оформлення основних цілей та функціональних можливостей проекту у вигляді специфікації.

Постановка задачі

Специфікація вимог до програмного забезпечення (англ. *Software Requirements Specification, SRS*) – це документ, який містить повний опис поведінки та умов експлуатації системи, що розробляється. Специфікація формується після консультації з усіма зацікавленими особами на етапі аналізу вимог до ПЗ. Опис вимог виконується природною мовою із використанням символьних і графічних позначень, зрозумілих кожній зі сторін.
Призначення специфікації:

- ✓ Допомагає переконатися, що всі зацікавлені сторони мають спільне розуміння системи, яка має бути розроблена.
- ✓ Дає змогу уникнути непорозумінь на наступних етапах розробки.
- ✓ Служить орієнтиром для всіх зацікавлених сторін у процесі розробки.
- ✓ Допомагає виявити будь-які прогалини у вимогах на ранніх стадіях.
- ✓ Містить погоджений перелік функціоналу, який необхідно реалізувати.
- ✓ Є ключовим документом для перевірки якості продукту на етапі тестування та введення в експлуатацію.
- ✓ Захищає юридично від претензій замовника на стадії приймання.

Структура специфікації може відрізнятися, в залежності від проекту та правил, визначених у компанії. У цій лабораторній роботі ми пропонуємо вам за обраним варіантом завдання скласти спрощений варіант специфікації, що міститиме такі основні розділи:

Вступ
1. Підстави для розробки
2. Призначення розробки
3. Вимоги до програмного продукту
4. Техніко-економічні показники
5. Порядок контролю і приймання.

Приклад оформлення специфікації наведено у Додатку Е.

У разі потреби можна уточнювати зміст розділів, вводити нові розділи або об'єднувати окремі з них.

Поставтесь до цього завдання, як до можливості сформувати обсяг робіт, який допоможе зорієнтуватися при плануванні та розподілі завдань по проекту.

У розділі "***Вступ***" вказують найменування, коротку характеристику області застосування програми або програмного продукту й об'єкта, в якому використовують програму або програмний продукт.

У розділі "***1. Підстави для розробки***" потрібно вказати:
- документ (документи), на підставі яких виконується розробка;
- організацію, що затвердила цей документ, і дату його затвердження;
- найменування і (або) умовне позначення теми розробки (див. Додаток Б).

Розділ "***2. Призначення розробки***" повинен містити обґрунтування **корисності** проекту і складатися з таких підрозділів:

2.1. Мета і призначення розробки – головний результат виконання та впровадження проекту з точки зору потреб ринку, бізнесу та/або користувачів. Для чого призначена програма, яку користь вона приноситиме.

2.2. Область застосування – визначення галузей/сфер, у яких буде використано розробку.

2.3. Зацікавлені сторони – стейкхолдери (англ. *stakeholders*), конкретні особи, які зацікавлені у якісній роботі майбутнього продукту, замовляють та/або фінансують розробку, оцінюватимуть результат її виконання на виході. Хто цільова аудиторія проекту: його користувачі та/або споживачі. Для кожної зацікавленої особи потрібно вказати посаду та роль у проекті. Саме із зацікавленими особами ведуться консультації щодо уточнення вимог.

Виокремити та ввести позначення груп користувачів, які працюватимуть з майбутньою програмою.

2.4. Перспективи впровадження продукту – опис того, як завдання, що має втілити проект, наразі забезпечуються, та які задачі/функції/дії можна буде виконувати ефективніше завдяки впровадженню розробленого продукту. Яка стратегія монетизації проекту: що виграє організація від його впровадження (**не лише фінансово**). Яких результатів вдасться досягнути одразу після впровадження проекту?

2.5. Огляд ринку – назвіть щонайменше три рішення, які використовують компанії-конкуренти (у межах міста/області/країни) для виконання аналогічної задачі. Який з проектів конкурентів вам подобається найбільше, які його сильні сторони. Які ключові особливості вашого проекту викликають сумніви чи можуть стати джерелами додаткових проблем/ризиків.

Розділ "***3. Вимоги до програмного продукту***" повинен містити наступні підрозділи:

3.1. Функціональні вимоги – цей підрозділ повинен містити перелік функціональних можливостей, які майбутня програмна розробка повинна забезпечити для **кожної групи користувачів** програми. Вимоги краще згрупувати за типом користувачів, а спільні вимоги – зазначити на початку.

3.2. Організації вхідних і вихідних даних – описати джерела та сховища даних, їх формат і тип, якщо можливо.

3.3. Вимоги до складу і параметрів технічних засобів указують необхідний склад апаратних характеристик пристрою, на якому розроблена програма працюватиме належним чином.

3.4. Вимоги до інформаційної і програмної сумісності – зазначають:

- Мови програмування і технології, що будуть використані для написання програмного коду.
- Сервіси, системи та будь-які сторонні програм, з якими взаємодіятиме програма,
- Операційні системи (ОС), браузери та інші програми, необхідні для стабільного користування програмним продуктом.
- Методи, стандарти, протоколи передавання та захисту даних.

3.4. *Експлуатаційні вимоги* – повинні зазначати вимоги для забезпечення надійного функціонування (забезпечення стійкого функціонування, контроль вхідної і вихідної інформації, час відновлення після відмови і т. п.).

Розділ *"**4. Очікувані техніко-економічні показники**"* дає змогу розрахувати орієнтовані показники розробки, як-от трудомісткість, тривалість, кількість виконавців та продуктивність, на основі рівня складності й оригінальності рішення та використаних технічних засобів.

У розділі "***5. Порядок контролю і приймання***" потрібно вказати види випробувань і загальні вимоги до приймання роботи.

У ***додатках*** до специфікації при необхідності наводять: перелік науково-дослідних і інших робіт, що обґрунтовують розробку; глосарій із поясненням введених термінів та позначень; схеми та діаграми, таблиці, документи, які можуть бути використані при розробці, а також посилання на корисні інформаційні джерела.

Порядок виконання роботи:

1. Визначити формат виконання практичних завдань: індивідуально або в складі команди.
2. Сформувати команду розробників (якщо в п.1 обрано другий формат). Учасники команди можуть бути з різних підгруп, проте мають спільно працювати на лабораторних та над курсовим проектом.
3. Обрати варіант завдання на курсову роботу та погодити його з викладачем (див. Додаток А).
4. Проаналізувати можливості майбутньої інформаційної системи. Вітаються дослідження інформації з додаткових джерел, консультації з викладачами, одногрупниками та фахівцями з розробки програмного забезпечення.
5. Розробити специфікацію вимог до програмного продукту (див. зразок у Додатку Е).
6. Оформити звіт до лабораторної роботи (вимоги до оформлення у Додатках Б-Д).

Звіт з лабораторної роботи повинен містити:

1. Номер та зміст обраного варіанта завдання.
2. Специфікацію вимог до програмного продукту в електронному та роздрукованому вигляді.
3. Короткі відповіді на запитання для самоперевірки.

Запитання для самоперевірки:

4. На які етапи поділяється процес розробки програмного продукту? Яка мета досягається на кожному з них?
5. Опишіть класичний життєвий цикл розробки ПЗ, у чому його переваги та недоліки.
6. Які ще моделі життєвого циклу вам відомі. Опишіть кожну з них за трьома найвиразнішими характеристиками.
7. У чому проявляється гнучкість методології Agile і чи завжди вона здатна замінити класичні підходи до розробки ПЗ?

8. Чим особисто для вас буде корисна специфікація вимог до програмного продукту? Які пункти потребували більше часу на підготовку?
9. Що собою являє статут проекту (Project Charter), що входить до його складу та яке його призначення?
10. Для чого визначати стейкхолдерів розробки, як з ними працювати та зберігати корисну інформацію про них? Що таке реєстр стейкхолдерів?

Рекомендовані інформаційні джерела:

1. SDLC (Software Development Life Cycle) Phases, Process, Models. URL : https://www.softwaretestinghelp.com/software-development-life-cycle-sdlc/
2. Agile in a nutshell. URL : http://www.agilenutshell.com/
3. Специфікація вимог. URL : https://qalight.ua/baza-znaniy/cpetsifikatsiya-vimog/
4. Що таке специфікація вимог: визначення, найкращі інструменти та методи. URL: https://visuresolutions.com/uk/blog/requirements-specification/
5. Як виглядає специфікація вимог і як її тестувати. URL : https://training.qatestlab.com/blog/technical-articles/what-the-requirements-specification-looks-like-and-how-to-test-it/
6. How to write software requirements specifications: best practices and SRS tools. URL: https://www.altexsoft.com/blog/software-requirements-specification/
7. Software requirements specification document with example. URL : https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database
8. Your 2023 guide to writing a software requirements specification (SPS) document. URL: https://relevant.software/blog/software-requirements-specification-srs-document/
9. Stakeholder analysis https://www.pmi.org/learning/library/stakeholder-analysis-pivotal-practice-projects-8905
10. How to write a Project Management Charter https://www.youtube.com/watch?v=I4JsU42IO6g&list=PLF1064CD7B0A98261&index=38&ab_channel=ProjectManager

Лабораторна робота № 2. Розробка ескізного проекту ПП. Створення діаграми прецедентів

Мета: розробити діаграму варіантів використання для системи за індивідуальним варіантом завдання і розвинути навички моделювання засобами мови UML.

Постановка задачі

UML (*Unified Modeling Language* – Уніфікована мова моделювання) – один з найбільш поширених засобів візуального моделювання, тобто подання моделі програмного продукту за допомогою деякого стандартного набору графічних позначень.

UML – це не мова програмування і може бути застосовано на всіх етапах життєвого циклу розроблення прикладних програм.

Rational Rose – це потужний інструмент аналізу та проектування об'єктно-орієнтованих програмних систем. Дане середовище дає змогу моделювати систему до написання коду, аби від самого початку забезпечити відповідність її архітектури. За допомогою готової моделі недоліки проекту легко виявити на стадіях, які не потребують значних витрат.

Типи діаграм UML

Графічні зображення моделей системи в UML називають **діаграмами**. В термінах мови UML визначено такі їх типи:

- **діаграма варіантів використання або прецедентів** (*use case diagram*)
- **діаграма класів** (*class diagram*)
- **діаграми поведінки** (*behavior diagrams*)
- **діаграма станів** (*statechart diagram*)
- **діаграма діяльності** (*activity diagram*)
- **діаграма взаємодії** (*interaction diagrams*)
- **діаграма послідовностей** (*sequence diagram*)
- **діаграма співпраці** (*collaboration diagram*)
- **діаграми реалізації** (*implementation diagrams*)
- **діаграма компонент** (*component diagram*)
- **діаграма розгортання** (*deployment diagram*)

Середовище Rational Rose дає змогу проектувати варіанти використання системи у вигляді діаграми для відображення її

функціональних можливостей. Діаграма Взаємодії зображає як об'єкти співпрацюють, виконуючи надані функціональні можливості Для відображення об'єктів системи та їх відношень використовуються діаграми Класів. Діаграма компонент відображає класи, що відповідають готовим фізичним компонентам системи. І, нарешті, діаграми Розгортання застосовують для візуалізації проекту розподілених систем.

Модель Rose – це всебічне подання системи. Вона містить всі діаграми UML розробки, її дійових осіб, варіанти використання, об'єкти, класи, компоненти і вузли системи. Вона детально описує що містить система і як вона функціонує, тому розробники можуть використовувати її як ескіз або креслення системи, яка створюється.

Діаграма Варіантів Використання або Прецедентів (Use Case Diagram)

Діаграми варіантів використання зображають функціональне призначення системи або те, що система повинна робити.

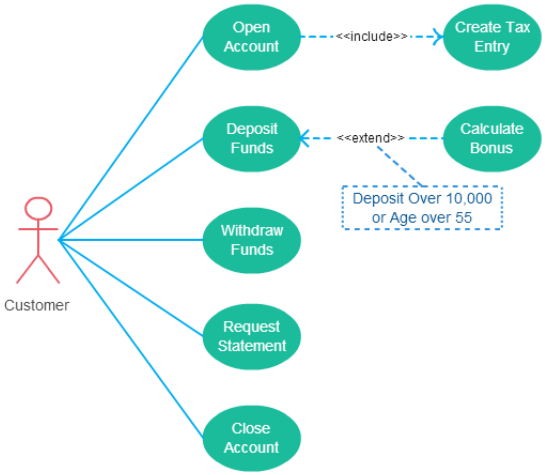


Рис. 2.1. Приклад діаграми варіантів використання

Розробка діаграми має на меті:

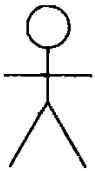
- визначити загальні границі і контекст предметної області, яка моделюється;
- сформулювати загальні вимоги до функціональної поведінки системи, яка проектується;

- розробити вихідну концептуальну модель системи для її подальшої деталізації у формі логічних і фізичних моделей;
- підготувати вихідну документацію для взаємодії розробників програми з її замовниками і користувачами.

Діаграма варіантів містить деякі варіанти використання системи, дійових осіб та зв'язки між ними (рис. 2.1).

Дійові особи (Актори)

Дійова особа або актор (*actor*) – це все, що може взаємодіяти з системою ззовні, як-от людина, технічний пристрій або будь-яка інша система, яка може впливати на систему так, як визначить сам розробник за вимогами замовниками. Стандартним графічним позначенням актора мовою UML на діаграмах є фігурка людини, під якою записують назву актора:



Назва актора

- ✓ Актор має бути пов'язаний принаймні з одним варіантом використання.
- ✓ Актор може бути пов'язаний із кількома варіантами використання.
- ✓ Кілька акторів можуть бути пов'язані з одним варіантом використання.

Прикладами акторів можуть бути: клієнт автосервісу, банківський службовець, продавець магазину, менеджер відділу продажу, пасажир авіарейсу, водій автомобіля, адміністратор готелю, мобільний телефон, термінал, платіжна система, сервер та інші сутності, які мають відношення до концептуальної моделі відповідної предметної області.

Дійові особи поділяються на три типи: *користувачі, які взаємодіють з системою, інші системи, які взаємодіють з даною системою та час.*

Перший тип користувачів – це фізичні особи, для позначення яких використовуються ролі, яку користувач виконує при використанні системи, а не їх посади.

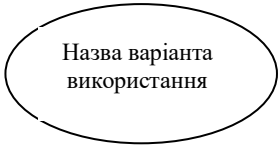
Другий тип дійових осіб – інша система, наприклад для банка – це кредитна система, яка використовується для роботи з інформацією про кредитні справи клієнтів, аутентифікаційний сервер, який ідентифікує клієнта при вході до системи, білінгва система, яка перевіряє стан рахунку користувача Інтернет, тощо.

Третім, найбільш поширеним *типом акторів* є час. Час стає дійовою особою, якщо від нього залежить запуск деяких подій в системі, наприклад оновлення бази антивірусної системи при запуску, або узгодження налаштувань банківських терміналів у визначений час, або автоматичне від'єднання від системи вразі відсутності запитів від користувача, тощо.

Варіанти використання (прецеденти)

Варіанти використання (*use case*) – це узагальнений опис функціональності системи на «високому рівні», тобто з точки зору користувача. Варіант використання служить для опису сервісів, які система надає акторові. Варіанти використання – це функції, які виконують актори під час взаємодії із системою.

Окремий **варіант використання** мовою UML позначається на діаграмі е***ліпсом***, всередині якого міститься його короткий опис: назва або ім'я у вигляді *дієслова* латинськими буквами.



Прикладами варіантів використання можуть бути такі дії, як: перевірка стану поточного рахунку клієнта, оформлення замовлення на купівлю товару, отримання додаткової інформації про студентів, формування звіту, відображення графічної форми на екрані та багато інших дій.

Наприклад, як зображено на рис. 2.1 клієнту доступний деякий набір функціональних можливостей:

1) відкрити рахунок
2) переглядати депозитні кошти
3) зняти кошти
4) запит заявки
5) закрити рахунок

На початку роботи над проектом виникає питання: «*Як виокремити варіанти використання?*» Найкращий спосіб: уважно перечитати документацію замовника, врахувати побажання кожної з зацікавлених осіб, подумати, що вони очікують від готового проекту. Слід кожній зацікавленій особі задати такі запитання:

Що вона збирається робити з системою?

Чи буде вона з її допомогою працювати з інформацією (вводити, отримувати, оновлювати, видаляти)?

Чи потрібно інформувати систему про деякі зовнішні події?

Чи повинна система інформувати користувача про будь-які зміни чи події?

Відношення (зв'язки)

Між елементами діаграми варіантів використання можуть існувати різноманітні ***відношення***, які зображають взаємодію екземплярів акторів і варіантів використання.

В мові UML існує декілька стандартних типів відношень між акторами і варіантами використання:

- **комунікації** (*association relationship*) зображає зв'язки між дійовими особами і варіантами використання;
- **розширення** (*extend relationship*) і включення (*include relationships*) подають зв'язки між варіантами використання;
- **узагальнення** (*generalization relationship*) – зв'язки між дійовими особами;

Відношення комунікації

Мовою UML позначається у вигляді суцільних стрілок (рис. 2.2):

Рис. 2.2. Приклад відношення комунікації між актором та варіантом використання

Напрямок стрілки відображає хто розпочинає взаємодію. У наведеному прикладі Клієнт за допомогою системи звертається до функції «Відкрити рахунок». Один або більше варіантів використання можуть бути спільними для декількох акторів.

Відношення розширення

Мовою UML відношення розширення позначається пунктирною стрілкою зі словом “extend”. Дане відношення додає більше функціональності системі за рахунок розширення базового варіанту використання. Розширений варіант використання залежить від розширеного (базового) варіанту використання, і не є обов’язковим.

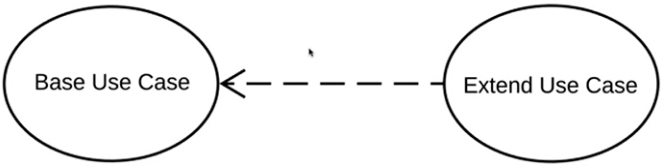


Рис. 2.3. Відношення розширення

У наведеному прикладі варіант використання «Нарахування бонусу» не має особливого сенсу без варіанту використання «Депозитні кошти» і активується лише для депозитів понад 10 000 або для осіб старше 55 років (рис. 2.4).

Рис. 2.4. Приклад відношення розширення між варіантами використання

Відношення включення

Відношення включення показує, що поведінка долученого варіанту використання є частиною базового варіанту використання. Основною причиною цього є повторне використання звичайних дій у кількох варіантах використання. У деяких ситуаціях це робиться для спрощення складної поведінки. Мовою UML відношення розширення позначається пунктирною стрілкою зі словом “include”. Включений варіант використання є обов’язковим.

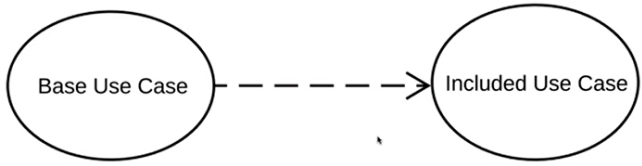


Рис. 2.5. Відношення включення

У прикладі на рис. 2.6 базовий варіант використання «Зняти гроші» входить до складу варіанту використання «Оновити баланс».

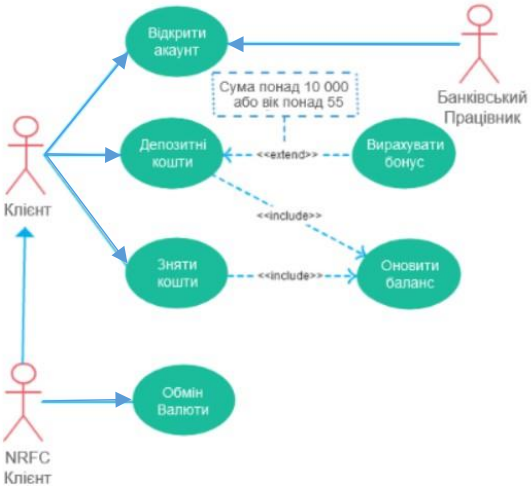


Рис. 2.6. Приклад відношення включення між варіантами використання

Відношення узагальнення

За допомогою зв'язку *узагальнення дійової особи* зображають, що у декількох дійових осіб є спільні риси. Узагальнення дійової особи означає, що один актор може успадкувати роль іншого актора. Наприклад, на діаграмі (рис. 2.7) показано, що існують лікарі різних спеціалізацій – окуліст, дантист, кардіолог та ін. Вони є конкретними дійовими особами. Дійова особа Лікар – абстрактна, підкреслює наявність лікарів різних спеціалізацій.

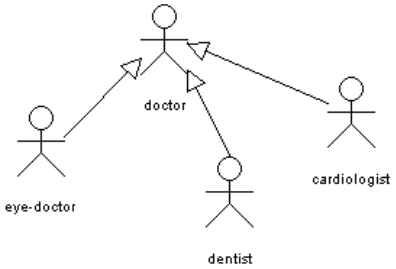


Рис. 2.7. Приклад відношення узагальнення дійової особи

У випадку варіанта використання *узагальнення* схоже на уніфікацію дійової особи. Наприклад, варіанти використання «Пошук за автором» і «Пошук за номером телефону» можна узагальнити у варіант використання «Пошук (рис. 2.8).

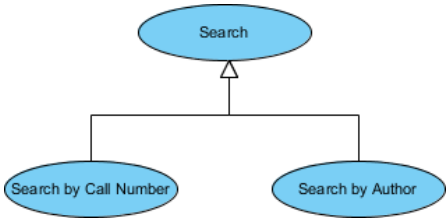


Рис. 2.8. Приклад відношення узагальнення варіантів використання

На діаграмі (рис. 2.9) варіанти використання «Інтернет-замовлення» чи «Замовлення по телефону» можна узагальнити в один варіант використання «Спосіб замовлення»

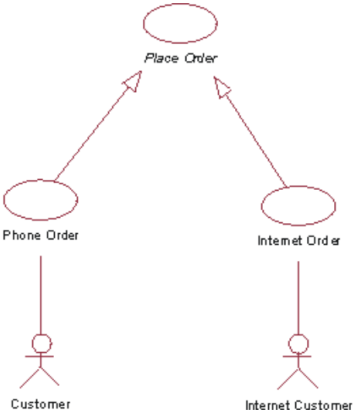


Рис. 2.9. Приклад відношення узагальнення варіантів використання

Правила розробки діаграми варіантів використання (прецедентів):

- Не моделюйте зв'язки між акторами. За визначенням, актор, тобто дійова особа, перебуває поза зоною дії системи, а отже зв'язки між ними не належать до її компетенції.
- Не з'єднуйте безпосередньо два варіанти використання (крім зв'язків використання і розширення).
- Діаграми цього типу подають лише варіанти використання, які доступні в системі, а не послідовність їх виконання.
- Кожен варіант використання повинен ініціюватися актором, тобто зв'язок завжди повинен починатися з актора і закінчуватися на варіанті використання. Винятком є зв'язки використання і розширення.
- За допомогою одного варіанта використання можна вводити дані, а отримувати їх – за допомогою іншого. Для зображення потоків інформації не потрібно малювати стрілки від одного варіанта використання до іншого.

Потік подій

Варіанти завдань описують, що повинна робити ваша система загалом, але для фактичної розробки потрібно більше конкретних відомостей, які визначаються в документі під назвою «**потік подій**» (*flow of event*). Метою потоку подій є документування процесу обробки даних, який реалізується в межах варіанта використання. Цей документ буде детально описувати, що робитиме користувач, а що – сама система. На цьому етапі неважлива мова написання програми. Головне описати ЩО буде робити система, а не ЯК вона буде це робити.

Потік подій містить:

- короткий опис
- передумову
- основний потік подій
- альтернативний потік подій
- післяумову.

Опис повинен бути коротким, стислим поясненням того, що буде робити варіант використання. Наприклад: переказ грошей, перегляд балансу тощо. Може збігатися з його назвою.

Передумова варіанта використання – це така умова, яка повинна виконуватися, перш ніж варіант використання почне свою роботу. Наприклад: переш ніж переказати гроші клієнт повинен ввести свої аутентифікаційні дані (пін-код), щоб купити книжку потрібно зареєструватися, тощо.

Основний і альтернативний потоки подій відображають конкретні деталі варіанта використання. Потік подій поетапно описує, що повинно відбуватися під час закладеної у варіант використання функціональності. Первинний (основний) і альтернативний потоки подій містять:

- опис того як запускається варіант використання;
- різноманітні шляхи виконання варіанту використання;
- нормальний (основний) потік подій варіанта використання;
- відхилення від основного потоку подій (альтернативні потоки);
- потік помилок;
- опис того, яким чином завершується робота варіанта використання.

Прикладом основного потоку буде опис процесу зняття грошей з рахунку.

Прикладом альтернативного потоку буде перелік подій при введенні неправильного пін-коду, або коли на рахунку недостатня сума грошей або закінчився терміну дії банківської картки, тощо.

Післяумова - це умова, яка виконується після завершення варіанта використання (встановлений прапорець, натиснута клавіша чи виконання попереднього варіанта використання).

ПРАКТИЧНІ РЕКОМЕНДАЦІЇ

Створення діаграми прецедентів Use Case

2.1. Для запуску програми Rational Rose натисніть: Пуск>Программы> Rational Software>Rational Rose Enterprise Edition.

2.2. Екран середовища Rational Rose показано на рис. 2.10.

В його складі можна виокремити п’ять основних елементів: рядок інструментів, панель «інструменти діаграми», вікно діаграми, браузер, і вікно документації.

Як показано на рис. 2.11, кнопки рядка інструментів дозволяють виконувати стандартні і спеціальні дії.

Вміст панелі інструментів діаграми змінюється в залежності від типу активної діаграми, для діаграми прецедентів, даний рядок має вигляд (рис. 2.12).

У вікні діаграми можна створювати, відображати і змінювати діаграму мовою UML.

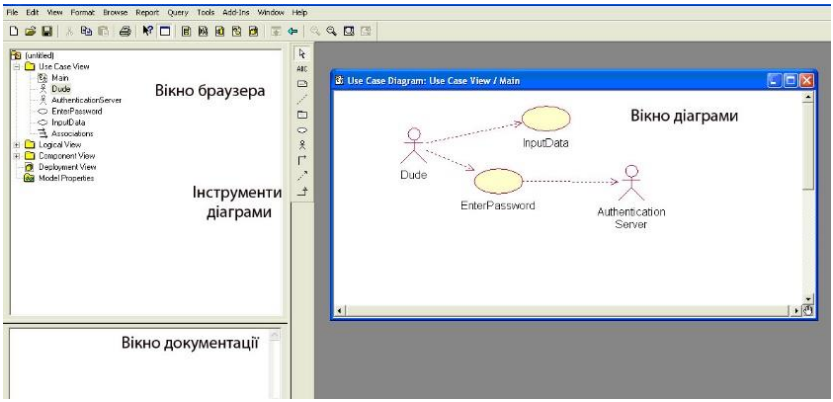


Рис. 2.10. Екран середовища Rational Rose

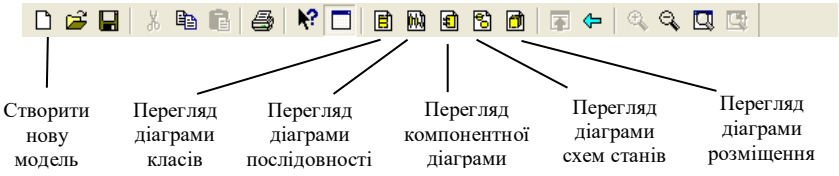


Рис. 2.11**.** Кнопки панелі інструментів Rational Rose

Браузер Rational Rose – це інструмент ієрархічної навігації, яка подає назву і піктограми, які відображають діаграми і елементи візуальної моделі.

Знак плюс (+) поруч із папкою означає, що всередині папки містяться додаткові елементи.

Вікно специфікації дозволяє задавати характеристики елемента діаграми (рис. 2.13).

В полі Documentation цього вікна вводять опис даного елемента. Цей самий опис можна вводити у вікні документації Rational Rose (коли цей елемент виділений у діаграмі).

- Стрілка вказівника для виділення об'єкта
- Додає текст до діаграми
- Додавання примітки
- Зв'язує примітку з варіантом використання або актором
- Розміщує новий пакет
- Створює варіант використання
- Створює нового актора
- Малює зв'язок між актором і варіантом використання
- Зображає залежність між елементами діаграми
- Позначає зв'язок використання або розширення між варіантами використання або успадкування між акторами

Рис. 2.12. Панель інструментів діаграми варіантів

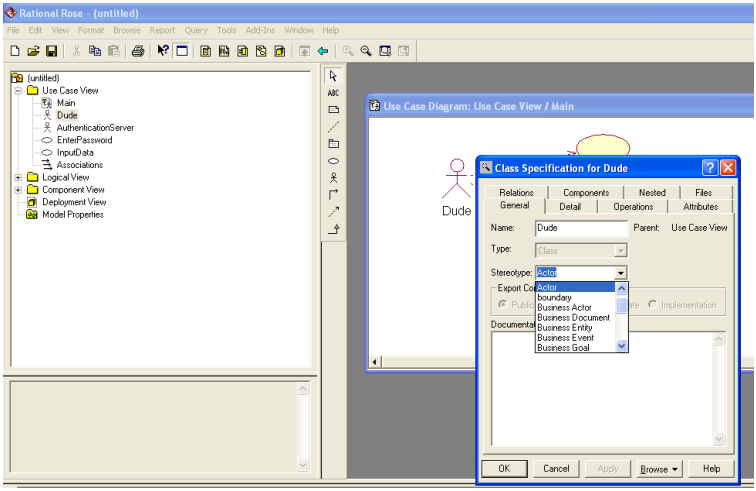


Рис. 2.13. Вікно специфікації Rational Rose

2.3. Створення діаграми прецедентів Use Case

Для доступу до Головної діаграми Варіантів Використання (Прецедентів):

1. В браузері клацніть мишкою на позначці + поруч з позначкою **Use Case View**. Дане вікно буде відкритим.
2. Двічі натиснувши на позначці **Main** (Головній діаграмі), відкрийте її.

Для створення нової діаграми Варіантів Використання:

1. Клацніть правою кнопкою миші на пакеті Зображення Варіантів Використання **Use Case View** в браузері.
2. У вікні, що з'явилося, оберіть пункт **New**> **Use Case Diagram** (Створити-Діаграма Варіантів Використання).
3. Виділити нову діаграму і введіть її ім'я.
4. Двічі клацнувши на іменованій діаграмі в браузері, відкрийте її.

Для відкриття створеної діаграми Варіантів Використання :

Двічі клацніть на її зображенні в браузері або в меню вибравши команду **Browse**> **Use Case Diagram** і у списку, що з'явився, оберіть потрібну діаграму.

2.4. Налаштування акторів

Для додавання нового актора:

1. На панелі інструментів натисніть на позначці актора.
2. Для додавання актора до діаграми, клацніть у потрібному місці діаграми.
3. Поки актор залишається виділеним, змінюйте його ім'я.

Зауважте: у назвах використовуються ЛАТИНСЬКІ літери. Назва позначає не посаду, а ***РОЛЬ****, яку дійова особа виконує по відношенню до системи.*

Для акторів можна також налаштовувати Специфікацію, вказавши його стереотип, або додавши опис актора, або зробивши його абстрактним.

Визначення кількості можливих екземплярів дійової особи (Множинність):

1. Клацніть правою кнопкою миші на акторі в браузері або на діаграмі Варіантів використання.
2. У меню, що з'явилося, оберіть пункт Open Specification.
3. Перейти у закладин ку Detail (Детально).
4. Оберіть потрібну множинність зі списку Cardinality, або введіть її самостійно, використавши один з форматів, вказаних у таблиці 2.1.

2.5. Налаштування варіантів використання

Для додавання нового варіанту використання на діаграму:

1. Натиснувши на кнопку Use Case на панелі інструментів або Вибравши в меню пункт Tools>Create>Use Case.
2. Клацніть мишкою на вільному місці діаграми Варіантів Використання щоб додати новий варіант використання, якому за замовчуванням надається ім'я NewUseCase.
3. Виділіть цей варіант використання і введіть нове ім'я
4. Новий варіант використання автоматично додається в браузер і розміщується нижче зображення Варіантів Використання.

Зауважте: у кожного варіанта використання моделі повинно бути унікальне ім'я. Звичайно – це дієслово або фраза, яка позначає дію. Назва задається ЛАТИНСЬКИМИ літерами. Назву за замовчуванням можна також змінити у вікні Специфікації.

Для створення специфікації варіанта використання:

Специфікація варіанта використання дозволяє документувати такі атрибути варіантів використання, як імена, пріоритети, стереотипи, опис, приєднувати документи, тощо.

Щоб відкрити специфікацію варіанта використання:

1. Клацніть правою кнопкою миші на варіантів використання в браузері або на діаграмі.
2. У вікні, що з'явилося, оберіть пункт Open Specification.

До варіанта використання можна додати опис того, для чого він передбачений (поле Documentation).

Щоб зробити варіант використання абстрактним клацніть правою кнопкою миші на варіанті використання, у вікні, що відкрилося, оберіть пункт Open Specification і встановіть прапорець Abstract (Абстрактний).

Закладинка Files (Файли) вікна Специфікації дозволяє додавати файли до варіанта використання. Файл може містити документ, який описує потік подій даного варіанта використання, документи з описом сценаріїв для даного варіанта використання, перелік вимог або інші документи з приблизними прототипами варіанта використання.

Щоб зв'язати файл з варіантом використанням:

1. Відкрийте вікно Специфікації для обраного варіанта використання.
2. Перейдіть до закладки Files (Файли) і натисніть правою кнопкою миші на її вільному полі.
3. У меню, що з'явилося, оберіть пункт Insert File (Додати файл).
4. В діалоговому вікні Open знайдіть і оберіть потрібний файл.
5. Натисніть кнопку Open щоб під'єднання до варіанта використання зазначений файл.

Налаштування відношень

Для створення відношень між актором і варіантом використання:

1. На панелі інструментів клацніть на позначці однонаправленної асоціації (Unidirectional Association, суцільна стрілка).

2. Клацніть на акторі і перетягніть лінію до потрібного варіанта використання.

Для додавання відношення розширення

Дане відношення може створюватися між окремими варіантами використання. За допомогою кнопки Generalization (Узагальнення, суцільна стрілка з трикутником на кінці) панелі інструментів можна намалювати зв'язок між варіантами використання, перший з яких доповнює функціональні можливості другого.

Клацнувши правою кнопкою миші у вікні, що з'явилося, оберіть пункт Open Specification. У вікні специфікації у полі Стереотип можна обрати, або ввести самому слово extends, яке позначатиме, що даний зв'язок є розширенням. Також можна додати опис відношення.

Порядок виконання роботи

1. Ознайомитися з теоретичним матеріалом до лабораторної роботи.
2. Ознайомитися з можливостями та компонентами середовища розробки Rational Rose.
3. У таблиці 2.2. навести перелік акторів вашої майбутньої системи та варіанти використання, з якими вони пов'язані. *Варіанти використання повинні узгоджуватися з функціональними вимогами до проекту, прописаними у технічному завданні.* В примітці потрібно зазначити номер(и) функціональної(их) вимог(и), якій відповідає варіант використання.

Таблиця 2.2

Актор	**Варіант використання**	**Примітка**

4. За даними табл. 2.2 у середовищі створити діаграму прецедентів вашої програмної системи.

5. Кожному учасникові команди описати потік подій (основний та альтернативні) для одного з найбільш важливий варіант використання.
6. Зазначити специфікації всіх елементів діаграми прецедентів.
7. Підготувати звіт до лабораторної роботи № 2.

Звіт з лабораторної роботи повинен містити:

1. Перелік функціональних вимог до програми зі специфікації ПЗ.
2. Заповнену табл. 2.2.
3. Потоки подій (п. 5)
4. Розроблену діаграму прецедентів.
5. Діаграму Ганта (див. зразок у Додатку Є).
6. Відповіді на запитання для самоперевірки.

Запитання для самоперевірки:

1. Що таке UML?
2. Для чого використовується візуальне моделювання?
3. Назвіть основні призначення діаграми варіантів використання (прецедентів).
4. Які основні графічні елементи діаграми прецедентів? Як вони позначаються і описуються?
5. Охарактеризуйте акторів вашої діаграми прецедентів.
6. Які бувають типи відношень на діаграмі прецедентів. Наведіть приклади кожного з них. Які відношення використані на вашій діаграмі?
7. У чому важливість етапу аналізу та формування вимог у життєвому циклі ПЗ?
8. Поясніть відмінності функціональних і нефункціональних вимог. З яких елементів вони складаються?
9. Опишіть один зі способів з’ясування вимог до розробки. Які його основні принципи та рекомендації щодо застосування?

10. Яким чином діаграма прецедентів допомагає уточнити вимоги до програмного забезпечення?

Рекомендовані інформаційні джерела:

1. UML Use Case Diagram. URL: https://www.javatpoint.com/uml-use-case-diagram
2. UML Use Case Diagram Tutorial. URL: https://www.youtube.com/watch?v=zid-MVo7M-E&list=PLUoebdZqEHTxpGCwKrb82cIvHNoNaBb4R&index=3&t=618s&ab_channel=LucidSoftware
3. Чотири типи зв'язків у діаграмі варіантів використання. URL: https://blog.visual-paradigm.com/the-four-types-of-relationship-in-use-case-diagram/
4. Use Case Diagram Relationships Explained with Examples. URL : https://creately.com/blog/diagrams/use-case-diagram-relationships/
5. Відношення випадків використання. URL: https://binaryterms.com/use-case-relationship.html
6. Як будувати UML-діаграми. URL: https://dou.ua/forums/topic/40575/
7. Booch G. Unified modeling language user guide. Addison-Wesley Professional, 2017. 504 p.
8. Rumpe B. Modeling with UML: language, concepts, methods, 1st edition. Springer, 2016. 295 p.
9. Developing UML diagrams. URL : https://francescolelli.info/generic/developing-uml-diagrams/

Лабораторна робота № 3. Взаємодія об’єктів. Створення діаграми послідовності

Мета: зобразити динаміку поведінки системи, що розробляється, за допомогою діаграми Послідовності.

Постановка задачі

Діаграми взаємодії

Діаграми взаємодії відображають один з процесів обробки інформації у варіанті використання. Наприклад для варіанта використання можна зобразити декілька альтернативних потоків. Це означає, що для цього варіанта використання потрібно створити декілька діаграм Взаємодії. В результаті на одній діаграмі буде показано, що відбувається, коли всі добре. На інших буде відображено хід подій за альтернативними потоками: що буде, коли клієнт ввів невірний пін-код, якщо на рахунку недостатньо грошей, ніж він хоче зняти, тощо.

Існує два типи діаграм Взаємодії – діаграми Послідовності (*Sequence*) і корпоративні діаграми (*Collaboration*).

Діаграми Послідовності відображають виконання операцій та стан об’єктів у часі. Вони зосереджені на управлінні та відображають ті самі деталі, які були описані в потоці подій, однак подають їх у більш зручній формі. Головними на цій діаграмі є об’єкти, які потрібно створити для реалізації функціональних можливостей, визначених на діаграмі прецедентів (рис. 3.1).

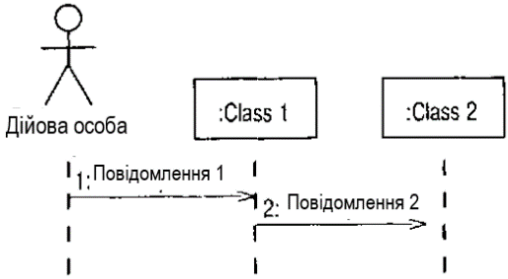


Рис. 3.1. Фрагмент діаграми Послідовності

На Корпоративні діаграмі зображується та сама інформація, лише в інший спосіб. Вона відображає потоки даних (рис. 3.2).

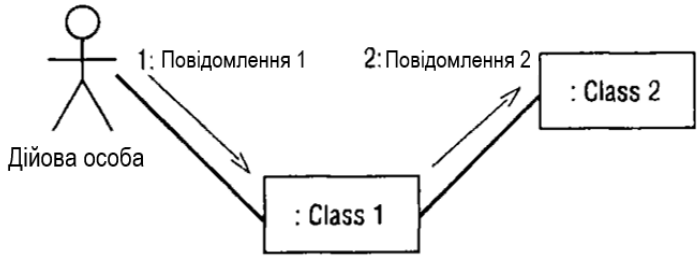


Рис. 3.2. Фрагмент Корпоративної діаграми

Діаграми обох типів можуть відображати об’єкти і класи майбутньої програми.

Об’єкт предметної області поєднує в собі деякі дані та поведінку системи, тобто описує реальні, конкретні компоненти. Прикладами об’єктів банківської системи є: банківський рахунок, екран банкомату, пристрій для зчитування пластикових карток, чек після зняття грошей, сторінка авторизації тощо.

Дані об’єкта називають атрибутами. Вони описують деякі характеристики об’єкта. Набір атрибутів для об’єкта є незмінним, змінюватися можуть лише дані самих атрибутів. Приклади атрибутів: баланс на рахунку, пін-код, дата і час зняття грошей, персональні дані власника, адреса банкомата, тип картки, поля реєстраційної форми тощо.

Поведінка об’єкта проявляється в його операціях : зміна суми на балансі, переказ грошей, перевірка персональних даних клієнта, видача чеку тощо.

Клас – це модель (схема) однотипних об’єктів предметної області. Він визначає атрибути і поведінку, які притаманні об’єктам (рис. 3.3).

Можна сказати, що клас – це проект будинку, в якому визначається кількість поверхів, планування квартир, розміщення та розміри кімнат, схема підведення комунікацій тощо. За цим проектом потім створюються будуються однотипні будівлі-об’єкти житлового масиву, кожному з яких надається своя адреса, номер, розміщення на карті.

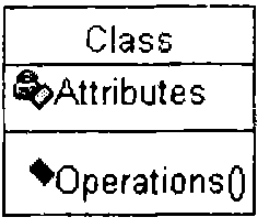


Рис. 3.3. Зображення класу на діаграмах Взаємодії

Як обрати потрібні об'єкти ?

Гарний спосіб для первинного вибору об'єктів – це виявлення іменників у потоці подій. Можна вивчити документацію, яка описує деякий сценарій використання. Одні іменники у ваших сценаріях будуть дійовими особами, другі – об'єктами, а треті – атрибутами об'єктів. Аби виокремити об'єкт подумайте чи виконує він якісь дії, функції. Якщо ні, тоді це – атрибут об'єкта, якщо так, тоді, швидше за все, це об'єкт.

За допомогою діаграма Взаємодій проектувальники і розробники системи можуть визначити класи, які потрібно створити, зв'язки між ними, а також операції кожного класу.

Діаграми Послідовності

Діаграма Послідовності – це впорядкована у часі діаграма взаємодії, інтерпретувати яку слід зверху донизу.

Якщо у Варіанта Використання є декілька альтернативних сценаріїв, кожен з них потрібно описати за допомогою окремої діаграми Послідовності.

На рис. 3.4 наведено приклад діаграми Послідовності, яка демонструє процес зняття готівки клієнтом банку через банкомат [1].

Об'єкти, які беруть участь в потоці, зображуються прямокутниками у верхній частині діаграми.

У кожного об'єкта є своя лінія життя (*lifeline*), яка подається вертикальною пунктирною лінією під об'єктом.

Повідомлення, які відповідають зв'язкам між об'єктами, зображають стрілками між лініями життя об'єктів. Повідомлення позначають, що один об'єкт викликає функцію іншого об'єкта.

Після визначення операцій класів, кожне повідомлення стане зверненням до відповідної функції класу. Повідомлення можуть бути рефлексивними, коли об’єкт звертається до своєї ж операції.

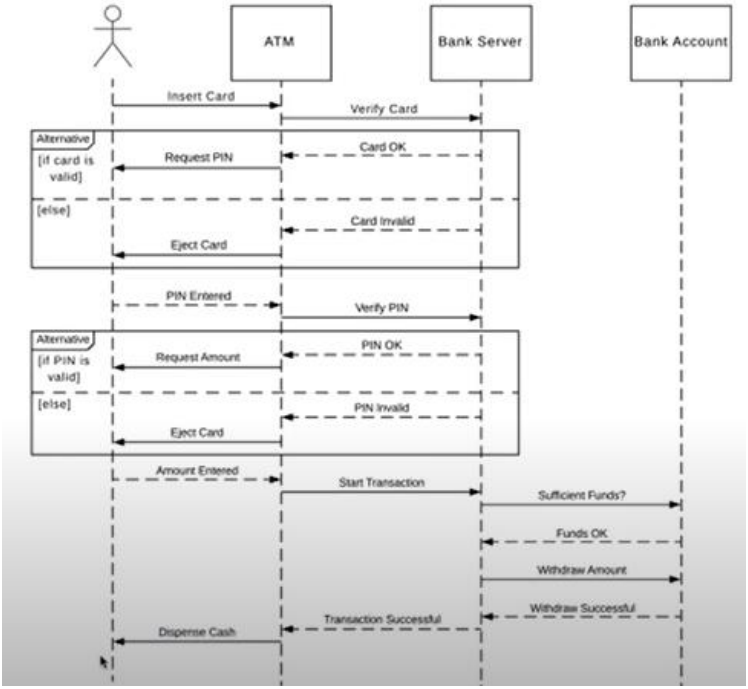


Рис. 3.4. Приклад діаграми Послідовності зняття готівки через банкомат (ATM)

- Стрілка вказівника для виділення об’єкта
- Додає текст до діаграми
- Додавання примітки
- Зв’язує примітку з варіантом використання або актором
- Розміщує на діаграму новий об’єкт
- Створює повідомлення між двома об’єктами
- Звернення до самого себе
- Зворотне повідомлення
- Інструмент для видалення

Рис. 3.5. Панель інструментів діаграми Послідовності

ПРАКТИЧНІ РЕКОМЕНДАЦІЇ

2.1. Створення діаграми Послідовності

Діаграму Послідовності можна створювати у представленні Варіантів Використання або в Логічному представленні браузера.

Для створення нової діаграми Послідовності:

1. Клацніть правою кнопкою миші на відповідному варіанті використання в браузері.
2. У меню, яке відкрилося, оберіть пункт New>Sequence Diagram (Створити>Діаграма Послідовності).
3. Надайте ім'я новій діаграмі.
4. Відкрийте нову діаграму, двічі клацнувши на ній в браузері.

Для відкриття вже створеної діаграми :

1. Знайдіть її у поданні Варіантів Використання браузера
2. Відкрийте її, двічі клацнувши на ній.

З конкретною діаграмою Послідовності можна зв'язувати файли і посилання. Наприклад, можна приєднати документ, який описує сценарій, зображений на діаграмі, або файл, який містить код програми, або документ зі специфічними для даної діаграми вимогами.

Для приєднання файлу до діаграми Послідовності:

1. Клацніть на діаграмі в браузері правою кнопкою миші.
2. У вікні, яке відкрилося, оберіть пункт New>File (Новий>Файл).
3. У діалоговому вікні відкриття файлу вкажіть той файл, який потрібно прикріпити.
4. Для завершення процедури натисніть кнопку Open.
5. Щоб відкрити приєднаний файл, двічі клацніть по ньому в браузері. Rational Rose запустить відповідний додаток і завантажить до нього файл.

2.2. Робота з об'єктами

Для додавання нового об'єкта на діаграму:

Скористайтеся кнопками панелі інструментів діаграми Послідовності (рис. 3.5). З їх допомогою можна додавати на діаграму нові об'єкти і повідомлення.

Для додавання об'єкта, який створений на діаграмі Варіантів Використання:

1. Виділіть зображення потрібного об'єкта (наприклад, актора) в браузері.

2. Перетягніть об'єкт з браузера на діаграму.

3. Клацніть на вільному місці у верхній частині діаграми, щоб розмістити туди новий об'єкт.

Кожен об'єкт діаграми Послідовності можна зіставити з класом. Наприклад, об'єкт клієнта банку можна пов'язати з класом Account (Рахунок), адже нам необхідно виконувати дії із рахунком, а не над користувачем. Для призначення об'єктові класу можна скористатися полем Class вікна специфікації даного об'єкта. За замовчуванням об'єктові призначається клас Unspecified (Не визначений).

Можна пов'язувати об'єкти з уже створеними класами моделі, або створювати нові класи.

Щоб пов'язати об'єкт діаграми Послідовності з класом:

1. Клацніть правою кнопкою миші на об'єкті.

2. У вікні, що з'явилося оберіть пункт Open Specification.

3. Якщо класи були створені раніше, то у спадному списку класів можна обрати назву потрібного класу. Для того, щоб створити новий клас оберіть пункт <New>. З'являтиметься вікно специфікації класів.

4. В полі Name введіть назву нового класу.

5. Натисніть ОК. Ви повертаєтеся до вікна специфікації об'єкта.

6. Зі списку класів оберіть щойно створений клас.

7. Натисніть ОК, щоб повернутися до діаграми. Після цього об'єкт має назву у форматі *Назва об'єкта: Назва класу.*

До моменту генерації коду всі об'єкти потрібно зіставити з класами.

Якщо потрібно пов'язати об'єкт з новим класом:

1. Клацніть правою кнопкою миші на об'єкт на діаграмі Послідовності.
2. У меню, що з'явилося, оберіть пункт Open Specification.
3. У списку класів оберіть пункт New.
4. З'являється вікно специфікації для нового класу.

Щоб переконатися, чи всі об’єкти співвіднесені з класами оберіть в меню моделі пункт Report>Show Unresolved Objects.

2.3. Робота з повідомленнями

Повідомлення (*massage*) – це зв’язок між об’єктами, я яких один з них (клієнт) вимагає від іншого (сервера) виконання деяких дій.

При генерації коду повідомлення транслюють у виклику функції.

У наведеному нижче прикладі (рис.3.6) одна форма просить іншу, аби та вивела її на екран.

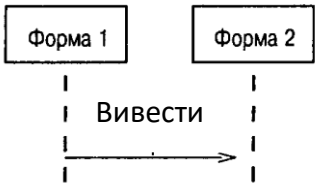


Рис.3.6. Зв’язок між об’єктами за допомогою повідомлень

Повідомлення зображають у вигляді стрілки, яку малюють між лініями життя двох об’єктів або від лінії життя об’єкта до самої себе *(рефлексивне повідомлення).* Повідомлення розташовують у хронологічному порядку, починаючи зверху.

Для додавання повідомлення на діаграму Послідовності:

1. Натисніть на кнопку Object Message (Повідомлення об’єкта) на панелі інструментів.
2. Тримаючи натиснутою ліву кнопку миші, проведіть нею від лініє життя об’єкта або дійової особи, які направляють повідомлення, до об’єкта, який отримує повідомлення.
3. Вдрукуйте текст повідомлення.

Якщо потрібно додати на діаграму повідомлення об’єкта до самого себе (рефлексне повідомлення), оберіть піктограму Message to self і клацніть на лініє життя об’єкта, який одночасно і відправляє і отримує повідомлення.

Зміна порядку повідомлень на діаграмі Послідовності:

1. Виділять повідомлення, яке ви бажаєте перемістити.
2. Перетягніть його у потрібне місце. При цьому

повідомлення на діаграмі будуть автоматично перенумеровані.

Якщо потрібно додати або прибрати нумерацію повідомлень:

1. У меню оберіть пункт Tools>Options.
2. Перейдіть на вкладку Diagram.
3. Встановіть або видаліть прапорець Sequence numbering (Нумерація повідомлень).

На діаграмах Послідовності можна зображати активацію (focus of control). *Активація* – це маленький прямокутник на лінії життя для позначення того, який з об'єктів активний і отримує керування в конкретний проміжок часу (рис. 3.7).

Для того щоб активувати або вимкнути активізацію:

1. В меню оберіть пункт Tools>Options.
2. Перейдіть на закладку Diagrams.
3. Встановіть або зніміть прапорець Focus of control.

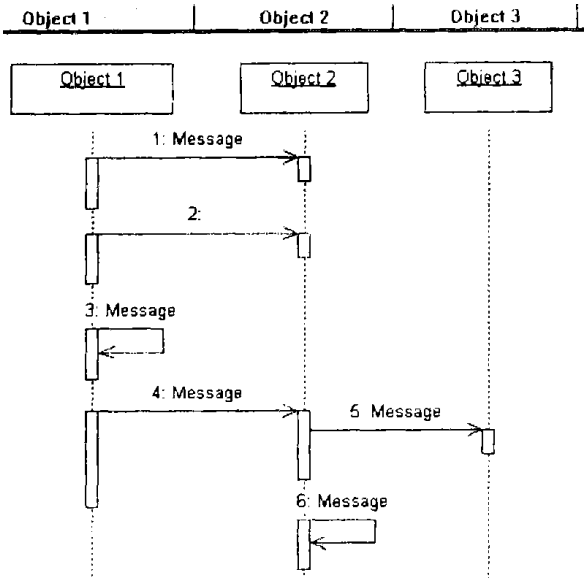


Рис. 3.7. Активізація на діаграмі послідовності

Отже, об'єкти на діаграмі Послідовності – це моделі майбутніх класів програми. Відомо, що класи взаємодіють між собою за допомогою обміну повідомлень: при надходженні повідомлення клас відповідає виконанням функції та поверненням результату виконання. Отже, перш ніж перейти до генерації коду, слід зіставити повідомлення діаграми Послідовностей з операціями (функціями, методами) класів. Наприклад, повідомлення «Запит деяких дій» буде зіставлено з операцією класу сервера.

Слід пересвідчитися, що об'єкти пов'язані з відповідними класами.

Для того, щоб пов'язати повідомлення з існуючою операцією:

1. Клацніть правою кнопкою миші на повідомленні на діаграмі Послідовностей
2. З'являтиметься перелік операцій сервера.
3. Оберіть необхідну операцію зі списку.

Якщо потрібно створити нову операцію для повідомлення:

1. Клацніть правою кнопкою миші на повідомленні діаграми Послідовності.
2. Оберіть пункт New Operation.
3. Введіть ім'я і деталі нової операції.
4. Натисніть ОК, щоб закрити вікно специфікації і закінчити створення нової операції.
5. Клацніть на повідомленні правою кнопкою миші.
6. У списку, що з'являється, оберіть нову операцію.

Аби переконатися, що кожне повідомлення пов'язане з операцією. Оберіть в меню моделі пункт Report>Show Unresolved Messages (Звіт-Показати вільні повідомлення). З'явиться перелік повідомлень, які ще не були пов'язані з класами.

При розробці діаграм Взаємодії часто використовують двоетапний підхід. Перш за все відображають інформацію високого рівня, яка потрібна для користувачів системи, що проектується. Повідомлення ще не зіставляють із операціями, а об'єкти – з класами. Ці діаграми дають змогу аналітикам, користувачам і всім зацікавленим в бізнес-процесі особам побачити як розвиватимуться події в системі.

Отримана на першому етапі діаграма може мати вигляд, наведений на рис. 3.8.

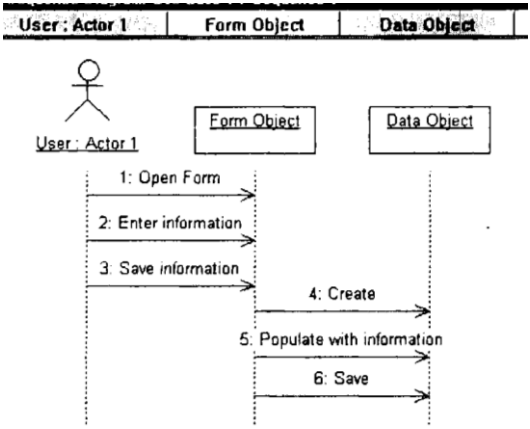


Рис. 3.8. Приклад простої діаграми Послідовності

На другому етапі на діаграмі розміщують деякі нові керуючі об'єкти, які відповідають за управління послідовністю подій сценарію. Для всіх варіантів використання і діаграм Послідовності це може бути один спільний об'єкт.

Після додавання керуючого об'єкта діаграма Послідовності матиме вигляд, зображений на рис. 3.9. Як бачимо з діаграми, керуючий об'єкт (Control Object) не реалізує ніяких бізнес-процесів, він лише надсилає повідомлення іншим об'єктам. Він відповідає за координацію дій інших об'єктів і розподіляє відповідальності. Такі об'єкти називають об'єктами-менеджерами.

Перевага використання керуючого об'єкта полягає у відокремленні бізнес-логіки від логіки, яка втілюється послідовністю подій. Якщо треба буде змінити послідовність, це торкнеться лише керуючого об'єкта.

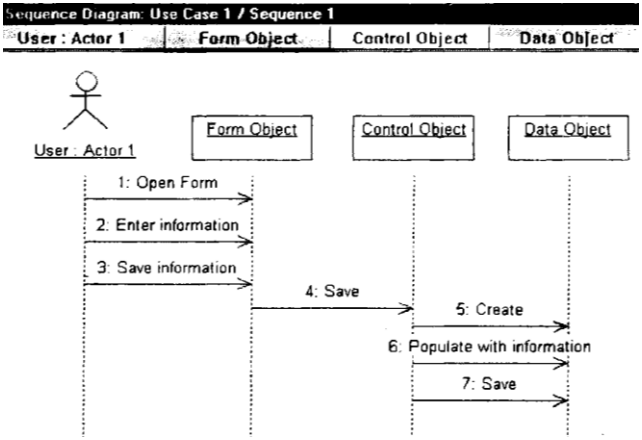


Рис. 3.9. Використання контролюючого об’єкту

На діаграмі можна розмістити інші об’єкти, які відповідають за безпеку, обробку помилок або за зв’язок із базою даних. Розглянемо приклад останньої. Припустимо, що нам необхідно зберігати в базі даних інформацію про нового співробітника – Джон Доу. Слід відокремити новий об’єкт від логіки бази даних, тоді його збереженням повинен займатися інший об’єкт «менеджер трансакцій». Він знає як коректно запитати інформацію з бази даних і як її зберегти (рис. 3.10).

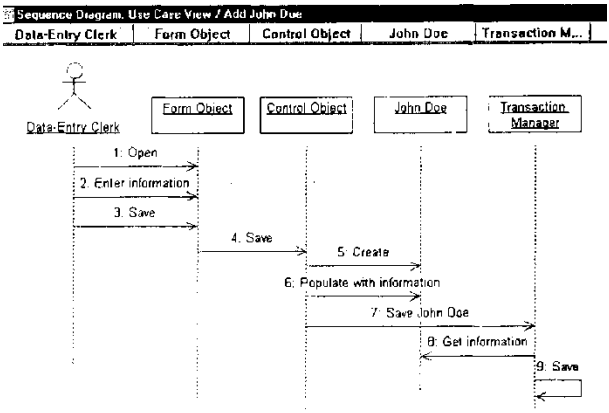


Рис. 3.10. Використання менеджера транзакцій

Перевагою такого підходу є можливість повторного використання об'єкта Джон Доу в іншому додатку та з іншою базою даних. До того ж зменшуються наслідки змін в системі. Зміни в базі даних не впливатимуть на логіку програми, і навпаки.

Порядок виконання роботи

1. Ознайомитися з теоретичним матеріалом до лабораторної роботи.
2. На основі діаграми Прецедентів, розробленої в лабораторній роботі № 2, для кожного варіанту використання створити окрему діаграму Послідовності, зокрема:

- визначити об'єкти (компоненти програмної системи), які взаємодіють, додавши, якщо потрібно, нові керуючі об'єкти;
- пов'язати об'єкти діаграми за допомогою повідомлень;
- зіставити створені об'єкти з відповідними класами;
- повідомлення між об'єктами необхідно зіставити з операціями відповідних класів.

3. Підготувати звіт до лабораторної роботи № 3.

Звіт з лабораторної роботи повинен містити:

1. Набір діаграм послідовності, позначені відповідним варіантом використання.
2. Короткі відповіді на запитання для самоперевірки.

Запитання для самоперевірки:

1. Яке призначення та різновиди діаграм Взаємодії?
2. Що таке об'єкт і як його ідентифікувати на діаграмі Послідовності? На що перетворюється об'єкт у майбутній програмі?
3. Яким чином об'єкти діаграми Послідовності взаємодіють?
4. Що таке архітектура програмного забезпечення і з яких етапів вона складається?
5. Коротко опишіть основні моделі структурування програмної системи?

6. В чому полягає моделювання управління і які підходи при цьому використовуються?
7. Які шаблони архітектури найбільш популярні на сьогодні?
8. Як обрати відповідний архітектурний стиль?

Рекомендовані інформаційні джерела:

1. How to Make a UML Sequence Diagram. https://www.youtube.com/watch?v=pCK6prSq8aw&list=PLUoebdZqEHTxpGCwKrb82cIvHNoNaBb4R&index=4&ab_channel=LucidSoftware
2. Sequence Diagram Tutorial – Complete Guide with Examples. URL: https://creately.com/guides/sequence-diagram-tutorial/
3. Як будувати UML-діаграми. URL: https://dou.ua/forums/topic/40575/
4. Free sequence diagram online tool. URL: sequencediagram.org
5. The clean architecture – Beginner's guide. URL : https://betterprogramming.pub/the-clean-architecture-beginners-guide-e4b7058c1165
6. 10 common software architectural patterns in a nutshell. URL : https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013
7. Types of software architecture patterns. URL : https://www.geeksforgeeks.org/types-of-software-architecture-patterns/
8. 10 best architecture patterns you must know about. URL : https://www.simform.com/blog/software-architecture-patterns/
9. Software architecture patterns. URL : https://www.decipherzone.com/blog-detail/software-architecture-patterns-type
10. The top 5 software architecture patterns: How to make the right choice. URL : https://techbeacon.com/app-dev-testing/top-5-software-architecture-patterns-how-make-right-choice

Лабораторна робота № 4. Проектування інтерфейсу користувача

Мета: навчитися створювати прототипи користувацьких інтерфейсів та продемонструвати варіанти реалізації функціональних можливостей програми.

Постановка задачі

Створення прототипів програми дає змогу побачити за формальними рядками ТЗ фрагменти реальної системи і виявити вимоги до користувацького інтерфейсу перед початком розробки.

Основні цілі, які вимагають застосування прототипів:

- з'ясування вимог до системи;
- вибір одного з багатьох концептуальних рішень;
- аналізу здійсненності розробки.

1. Нечіткі вимоги. Часто замовникові складно сформулювати вимоги до того, що саме він очікує від системи. В цьому випадку прототип інтерфейсу користувача (*User Interface, UI*), оперативно створений за результатами консультацій, допоможе побачити схематичну реалізацію того, як виконавець зрозумів відповідну частину системи. При цьому будь-який результат створення прототипів матиме свою цінність: якщо виконавець добре зрозумів вимоги – користь очевидна; якщо не зовсім – користь полягає в тому, що замовник може вказати на невідповідності та зробити незрозуміле зрозумілим.

2. Різноманітні варіанти рішення. Будь-яку технічну задачу можна розв'язати декількома способами, це стосується реалізації UI. Можна створювати декілька альтернативних сценаріїв і обрати найбільш прийнятний варіант, або ж об'єднати запропоновані варіанти для створення комбінаційного рішення, яке б об'єднувало переваги наданих варіантів.

3. Аналіз здійсненності. Часто буває так, що перелік функціональних, нефункціональних вимог та обмежень може викликати ризик неможливості реалізації системи. Зазвичай такий ризик пов'язаний з вимогами до швидкодії системи при відомих обмеженнях середовища її реалізації. В цьому випадку

створюються прототипи (не обов'язково пов'язані з UI), які реалізують відповідні частини системи, що імітують потоки даних, які надходять на її вхід та їх обробку.

Класифікація прототипів:

- горизонтальні та вертикальні;
- одноразові та еволюційні;
- паперові й електронні.

Горизонтальний або поведінковий прототип (*horizontal prototype, behavioral prototype*) моделює інтерфейс користувача, не торкаючись логіки обробки та звернень до бази даних.

Якщо у розробленому інтерфейсі використовується взаємодія з БД – вона імітується в програмному коді і не повинна відображатися в реалізації інтерфейсу. При цьому на екрані слід використовувати терміни, знайомі користувачеві, які відображають реальну специфіку предметної області, інакше користувачеві буде важко зосередитися. Якщо при натисненні на елемент управління повинні виконуватися якісь розрахунки або запити – результати також імітуються. Бажано реалізувати переміщення між екранами/ вікнами/сторінками в процесі виконання функцій, аби користувач зміг зрозуміти, як буде діяти система у відповідь на його дії.

Горизонтальні прототипи слід використовувати для досягнення мети з'ясування нечітких, або багато альтернативних вимог.

Вертикальний або структурний прототип (*vertical prototype, structural prototype*) не обмежується інтерфейсом користувача. Він реалізує вертикальний «зріз» системи, торкаючись усіх рівнів її реалізації. При створенні прототипів цього типу рекомендовано використовувати ті самі мови та середовища реалізації, що і при створенні цілісної системи.

Основні цілі таких прототипів – аналіз здійсненності та перевірка архітектурних концепцій.

Одноразовий або дослідницький прототип (*throwaway prototype, exploratory prototype*) створюється, коли потрібно швидко створити макет, що відображає ті чи інші аспекти та компоненти системи.

Еволюційний прототип (*evolutionary prototype*) створюється як перше наближення системи, покликане згодом стати самою системою. Код еволюційного прототипа повинен поступово, протягом однієї чи більше ітерацій, перетворитися в код цільового застосунку.

Паперовий прототип (*paper prototype*) – чудова альтернатива, коли розробник обмежений у ресурсах. Нариси інтерфейсів на папері, звичайно, не замінять інтерфейс, створений у середовищі розробки, проте, у таких прототипів є дві суттєві переваги:

1. Замовник не стане акцентувати увагу на кольоровому рішенні, формі кнопок і т.д., відволікаючись від аналізу функціональності.

2. Замовник ніколи не скаже, дивлячись на паперовий інтерфейс: «Бачу, що ви вже виконали систему на 85%! Нумо завершимо її протягом тижня».

Електронний прототип є аналогом паперового, проте він виконується у вигляді схеми чи презентації за допомогою засобів візуалізації, як-от Microsoft Visio, Microsoft PowerPoint, Adobe Photoshop, Figma та ін. [1, 5]). В цьому випадку користувач позбавлений свободи вибору, яку надає поведінкові або паперові прототипи. Але ідею покрокової зміни екранів в процесі демонстрації сценарію варіанта використання цілком можливо реалізувати.

Порядок виконання роботи

1. Проаналізувати аналоги програмних інтерфейсів застосунків за вашим варіантом завданням. Які рішення найбільш вдалі, а яких варто уникати?
2. На основі діаграм послідовностей, створених у лабораторній роботі № 3, розробити паперовий або електронний прототип користувацького інтерфейсу майбутнього програмного продукту. Допускається доповнення та зміна раніше створених моделей. Приклади створення паперових інтерфейсів можна переглянути в [6–9].

3. Розроблений макет повинен відображати основні функціональні можливості користувачів з різним рівнем доступу.
4. На порталі [3] наведено рекомендації щодо оформлення типових компонентів застосунків електронної комерції.
5. При створенні прототипу слід дотримуватися рекомендацій щодо оформлення інтерфейсів та розміщення елементів [10–14].
6. Необхідно забезпечити подання не лише основних вікон (сторінок) програми, але й пунктів меню, елементів навігації, попереджувальних повідомлень та сповіщень, які супроводжують дії користувача.
7. Прототип користувацького інтерфейсу може створювати як на папері так і в довільному середовищі візуального проектування або графічному редакторі.

Звіт з лабораторної роботи повинен містити:

1. Прототип користувацького інтерфейсу програми
2. Посилання на приклади найбільш вдалих і найбільш незручних інтерфейсів.
3. Відповіді на запитання для самоперевірки.

Запитання для самоперевірки:

1. Назвіть основні принципи, яких потрібно дотримуватися при розробці користувацьких інтерфейсів.
2. Якими рисами, на вашу думку, повинен володіти інтерфейс, дружній до користувача (user friendly interface)?
3. У чому різниця між UI- та UX-дизайном? Які їх основні цілі?
4. Яких рекомендацій слід дотримуватися при створенні зручних інтерфейсів?
5. Назвіть основні помилки при оформленні (зокрема, web-сайтів, мобільних додатків), наведіть приклади невдалого оформлення з власного досвіду.
6. Які засоби підтримки користувача та взаємодії з ним є ознаками якісно розробленого інтерфейсу?

7. За допомогою яких підходів можна досягти відповідності інтерфейсу потребам користувачів?
8. Що таке A/B тестування?
9. Як оцінити зручність використання програми у реальних умовах та зібрати відгуки користувачів? Чи брали ви участь у таких засобах зворотного зв'язку?
10. Окрім зручності, які ще характеристики інтерфейсу потрібно тестувати?

Рекомендовані інформаційні джерела:

1. Як створити прототип сайту на папері, в програмі та онлайн. Для чого він вам потрібен? URL : https://web-promo.ua/ua/blog/kak-sozdat-prototip-sajta-na-bumage-v-programme-ili-onlajn-dlya-chego-vam-nuzhen/
2. Тестування UI (інтерфейсу користувача) URL : https://wezom.com.ua/ua/blog/testing-ui-user-interface
3. Jakub Linowski. Improve your UI with winning & loosing A/B tests. URL : https://goodui.org/
4. Головне про зручність в інтерфейсах. URL : https://cases.media/article/golovne-pro-zruchnist-v-interfeisakh
5. Learn design. URL : https://www.figma.com/resources/learn design/
6. Paper prototype UX (User Experience). URL : https://www.youtube.com/watch?v=GrV2SZuRPv0&ab_channel=ChannyYun-%EC%9C%A4%EC%84%9D%EC%B0%AC
7. How to paper prototype an App. URL : https://www.youtube.com/watch?v=dVcK4pYbNRg&ab_channel=comerge
8. Mobile application design: Paper prototype video. URL : https://www.youtube.com/watch?v=y20E3qBmHpg&ab_channel=CormacR
9. Paper prototyping. URL : https://www.youtube.com/watch?v=_g4GGtJ8NCY&ab_channel=lukenwarm
10. The 15 rules every UX designer should know. URL : https://xd.adobe.com/ideas/career-tips/15-rules-every-ux-designer-know/

11. 27 UI/UX design principles and best practices 2021. URL : https://729solutions.com/ux-ui-best-practices/
12. Best practices for rocking UX design. URL : https://lucidspark.com/blog/ux-design-best-practices
13. The 5 golden rules of UI/UX design. URL : https://www.jobsity.com/blog/5-best-practices-for-ux-ui-design
14. Правильна типографія = зручність сприйняття вашого контенту! URL : https://goldwebsolutions.com/uk/blog/pravilna-tipografiya-u-sviti-didzhetel/

Лабораторна робота № 5. Створення діаграми класів

Мета: розробили логічну модель взаємодії складових компонент програмної системи за допомогою діаграми Класів.

Постановка задачі

Клас визначає атрибути і методи набору об'єктів. Всі об'єкти цього класу (екземпляри класу) мають спільну поведінку і однаковий набір атрибутів, водночас кожен об'єкт матиме свій власний набір значень. Класи використовуються в процесі аналізу та моделювання предметної області для формування словника або метаданих розроблюваної системи, який міститиме як абстрактні (узагальнені) поняття зі сфери застосування програми, так і ключові класи, що описують програмні або апаратні сутності.

Діаграма класів (*class diagram*) – це набір статичних, декларативних елементів моделі. Діаграми класів можуть застосовуватися і при прямому проектуванні (*forward engineering*), тобто в процесі розробки нової системи, і при зворотному проектуванні (*reverse engineering*) для опису вже створених систем, які використовуються. Інформація з діаграми класів безпосередньо перетворюється на вихідний код програми – у деяких інструментах UML-моделювання доступна функція генерування коду для певної мови програмування (зазвичай Java або C++). Таким чином, діаграма класів – це кінцевий результат проектування і відправна точка процесу розробки. За замовчуванням у середовищі Rose існує одна головна діаграма класів на ім'я Main.

У UML класи позначаються прямокутниками з назвою класу, у цих прямокутниках у вигляді двох частин: для зображення атрибутів та операції класу (рис. 4.1).

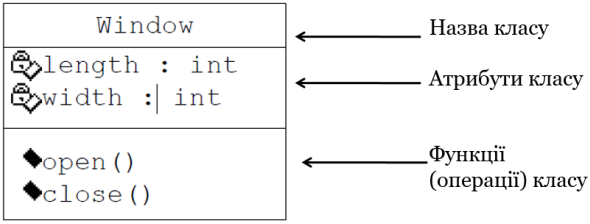


Рис. 4.1. Зображення класу в UML

Призначення стереотипу для класу

Стереотип – це механізм, який дає змогу створювати категорії класів. Припустимо, нам потрібно позначити всі форми у вашій моделі. Для цього можна створити стереотип Form (Форма) і призначити його усім вікнам вашої прикладної програми. Надалі при пошуку форм потрібно шукати класи з цим стереотипом.

Мовою UML визначено три основні стереотипи: Boundary (Границя), Entity (Сутність) і Control (Управління) (рис. 4.2).

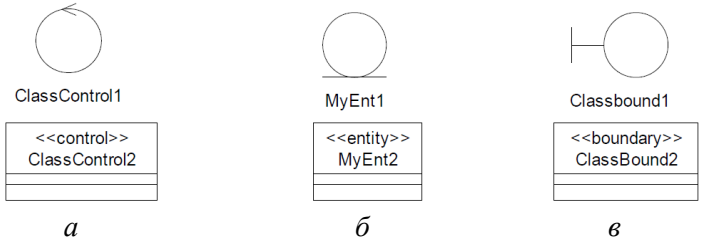


Рис.4.2. Графічні позначення основних стереотипів класів в UML: *а* – керуючий клас, *б* – клас-сутність, *в* – граничний клас

Граничні класи (*Boundary classes*)

Це класи, які розташовуються на границі системи з оточуючим середовищем. До них належать форми, звіти, користувацькі та апаратні інтерфейси, як-от сторінка авторизації та принтер, а також канали зв'язку з іншими системами.

На діаграмі прецедентів між дійовою особою та варіантом використання має бути принаймні один граничний клас, який може бути спільним для декількох акторів.

Класи-сутності (*Entity classes*)
Містять інформацію, яка постійно зберігається. Наприклад, у системі обробки даних про співробітників прикладом класу-сутності є клас Employee (Співробітник). Вони найбільш важливі для користувача, найчастіше – це батьківські класи, які містять базову інформацію для системи. У їх назвах використовують терміни, характерні для конкретної предметної області.

Часто для кожного класу-сутності створюють таблицю в базі даних. На основі об'єктної моделі можна розробити структуру бази даних. Адже вимоги до роботи системи зображуються у потоці подій, і в подальшому на діаграмі Послідовності. Потік подій визначає об'єкти, класи і атрибути класів. Кожен атрибут класу-сутності стає полем у базі даних.

Керуючі класи (*Control classes***)**

Відповідають за координацію дій інших класів. Завичай у кожного варіанта використання повинен бути один проміжний керуючий клас, що контролює послідовність подій цього варіанта використання. Можна застосовувати один керуючий клас для декількох варіантів використання. Наприклад, клас SecurityManager (Менежер безпеки) може відповідати за всі події, пов'язані з захистом, а клас TransactionManager (Менеджер транзакцій) – займається координацією повідомлень при обміні даними із базою даних.

Атрибути

Атрибут – це фрагмент інформації, пов'язаної з класом. Атрибути ще називають полями або компонентними даними класу. Наприклад, у класу Company (Компанія) можуть бути атрибути Name (Назва), Address (Адреса), NumbOfEmpl (Кількість працівників). Rational Rose дає можливість додавати атрибути до моделей класів.

Існує декілька джерел, в яких можна виявити атрибути. Для початку можна вивчити потік подій варіанта використання. Іменники, які в ньому використовуються, можуть відповідати як об'єктам класів ти дійовим особам, так і їх характеристикам. Останні відповідають атрибутам. Наприклад, у потоці подій

записано: «Користувач вводить ім'я співробітника, його адресу, номер страховки та номер телефону». Це означає, що у класі Співробітник наявні атрибути Ім'я, Адреса, Номер страховки та Номер телефону.

Атрибути можна виявити, вивчивши документацію, яка надає опис вимог до системи. Шукайте вимоги, які визначають дані, які зберігаються системою. Будь-який такий елемент може бути атрибутом класу.

Нарешті, погляньте на структуру бази даних, якщо вона вже визначена. Поля таблиць – це атрибути класів. Часто існує однозначна відповідність між таблицями БД і класами-сутностями. Якщо повернутися до попереднього прикладу, то в таблиці Співробітник повинні бути поля Ім'я, Адреса, Номер телефону, Номер страховки. Відповідний цій таблиці клас Співробітник, повинен мати такі самі атрибути.

У UML атрибути позначають щонайменше назвою, також може бути вказано їх тип, початкове значення і інші властивості. Крім того, для атрибутів можна вказати область видимості атрибута:

- + відповідає *публічним (public)* атрибутам
- # відповідає *захищеним (protected)* атрибутам
- - відповідає *приватним (private)* атрибутам

Операції

Операція – це поведінка, пов'язана з класом. Операцію ще називають функцією або методом класу. Вона складається з трьох частин: імені, параметрів (вхідних аргументів) і типу значення, яке повертається (результат виконання операції).

При виявленні операцій під час аналізу класів слід звернути увагу на таке:

- З підозрою поставтеся до будь-якого класу, який містить одну або дві операції. Можливо, клас записаний правильно, але його доцільно було б об'єднати з деяким іншим класом.
- Не може бути класу без операцій. Краще його подати як набір атрибутів деякого іншого класу.
- З обережністю ставтеся до класів з надто великою

кількістю операцій. Таким класом важко керувати. В такому випадку слід розбити даний клас на два і більше дрібніших класів.

Для операцій, як і для атрибутів, можна вказати область видимості:

- + відповідає *публічним (public)* операціям
- # відповідає *захищеним (protected)* операціям
- - відповідає *приватним (private)* операціям

Зв'язки між класами

Наслідування є однією з фундаментальних основ об'єктно-орієнтованого програмування, у якому клас «успадковує» всі атрибути й операції класу, нащадком якого він є, і може перевизначати або змінювати деякі з них, а також поповнюватися власними атрибутами й методами. Саме завдяки такій взаємодії між класами утворюються зв'язки.

Виявлення зв'язків

Майже вся необхідна інформація про зв'язки описана на діаграмах Послідовності.

Розглянемо основні типи зв'язків між класами.

Узагальнення (*generalization*)

В UML зв'язок узагальнення між класами моделює взаємодію типу «конкретний клас–супер клас», при якій конкретний клас-нащадок успадковує інтерфейс та основні властивості від базового супер класу. У UML узагальнення буде показано у вигляді лінії, яка поєднує два класи, зі стрілкою, яку спрямовано від підкласу до супер класу, як зображено на рис. 4.3.

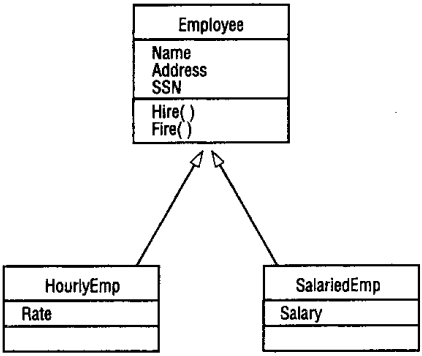


Рис. 4.3. Зображення зв'язку узагальнення у UML

Асоціації (*association*)

Асоціація позначає один з найбільш поширений взаємозв'язків між класами. Це механізм, який дає об'єктам змогу обмінюватися даними між собою. Асоціацію між об'єктами можна виявити на діаграмі Послідовності. Якщо один об'єкт надсилає повідомлення іншому – це одностороння асоціація, якщо другий об'єкт також зі свого боку надсилає повідомлення – йдеться про двосторонню асоціацію.

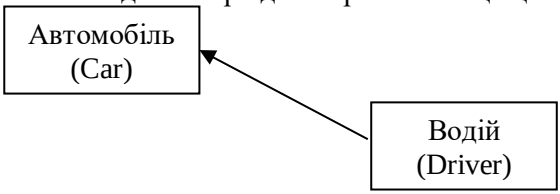


Рис. 4.4. Приклад однонаправленої асоціації між класами

У прикладі на рис. 4.4 зображено односторонню асоціацію між класом Водій та класом Автомобіль. При цьому, Водій може використовувати авто і виконувати на ньому певні дії. До того ж Водієві відомі атрибути і операції Автомобіля, проте Автомобіль не знає про функції та характеристики Водія, саме тому асоціація однонаправлена. Повідомлення на діаграмах Послідовності можуть надсилатися Водієм і отримуються Автомобілем, але не навпаки.

Залежність (*dependency***)**

Зв'язок залежності також відображає зв'язок між класами, але трохи в інший спосіб. Залежність завжди однонаправлена, вона зображає те, що один клас залежить від іншого класу. Спеціальні атрибути для класів, пов'язані залежністю, не створюються. Залежність зображується у вигляді стрілки пунктирною лінією. Так, у прикладі на рис. 4.5 клас Клієнт залежить від сервісів, які надає Сервер і може викликати функції або дані з його боку.

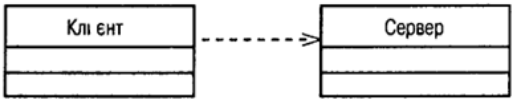


Рис. 4.5. Зв'язок залежності в UML

Агрегація (*aggregation*)

Агрегації – це особливий тип асоціацій, за якого два типи класів, що беруть в ньому участь, нерівнозначні, а взаємодіють за принципом «ціле-частина». За допомогою агрегації можна описати, як клас, що виконує роль цілого, складається з інших класів-частин. При цьому клас-ціле завжди має множинність 1. У UML агрегації буде показано стрілками, вістрями яких є незамальовані ромби з боку цілої частини (рис. 4.6).

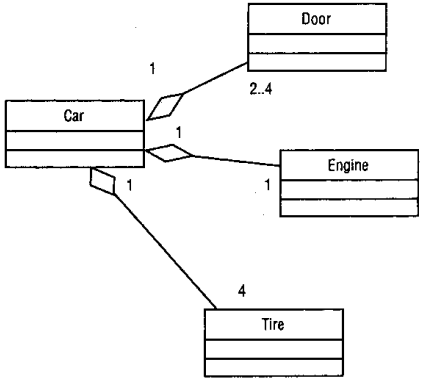


Рис. 4.6. Приклад агрегації, де клас-ціле Авто (Car) складається з частин – Двері (Door), Двигун (Engine), Колеса (Tire)

При генерації коду для агрегації автоматично створюються додаткові атрибути, які її реалізують. Так, клас-ціле Car отримає атрибути, що відповідають класам Door, Engine, Tire та всім іншим частинам зв'язку агрегації (рис. 4.6).

Проста агрегація передбачає, що частини, відокремлені від цілого, можуть продовжувати своє незалежне існування. Проте існує ще один тип агрегації – **композиція**. У цьому випадку ціле складається з частин і володіє їх властивостями, проте час життя компонентів дорівнює часу існування цілого, тобто частини неможна використовувати незалежно від цілого.

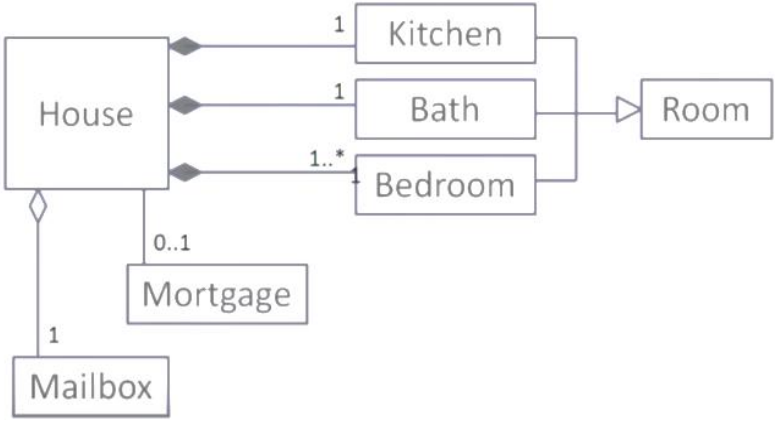


Рис. 4.7. Приклад різних типів зв'язків між класами на UML-діаграмі

Так, у прикладі на рис. 4.7 складові класи Кухня (Kitchen), Ванна (Bathroom) та Спальня (Bedroom) пов'язані композицією з класом Будинок (House) і можуть існувати самостійно.

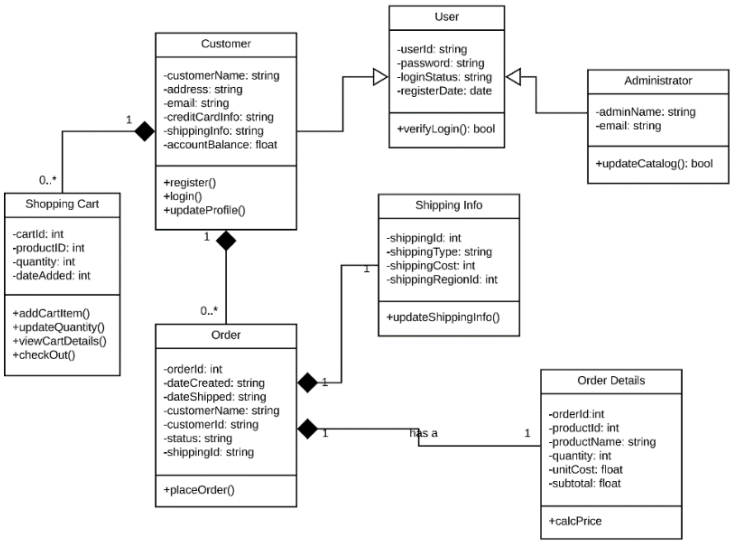


Рис. 4.8. Діаграма класів «Онлайн замовлення»

На рис. 4.8. зображена діаграма класів, яка описує систему оформлення онлайн-замовлень. Зверніть увагу на позначення множинності, а також на типи зв'язків. Зокрема, класи Customer (Клієнт) і Administrator (Адміністратор) пов’язані відношенням узагальнення із супер класом User, що містить спільні для цих підкласів властивості та метод з перевірки ідентичності. А клас Order (Замовлення) поєднує в собі особливості класів Order Details (Деталі замовлення) та Shipping Info (Інформація про доставку) за допомогою композиції. Водночас клас Order разом із Shopping Cart (Корзина) пов’язані з клієнтом (Customer), який їх оформив.

Для з’ясування типу зв’язку слід вжити таких дій:

1. Почніть з вивчення діаграми Послідовності. Якщо клас А надсилає повідомлення класу В, між цими класами потрібно встановити зв’язок. Зазвичай, це зв’язки асоціації або залежності.
2. Вивчіть наявність у класах зв’язку ціле-частина. Будь-який клас, що складається з інших класів, може брати участь у зв’язках агрегації.

3. Дослідіть класи на предмет узагальнення. Знайдіть всі класи, з які складаються з декількох типів або варіантів. Наприклад, у вас може бути клас Employee (Співробітник). А в компанії є два типи співробітників – які отримують погодинну оплату (HourlyEmp) і фіксований оклад (SalaryEmp). Це означає, що треба створити два класи для цих двох випадків, які є нащадками класу Employee. Спільні для всіх співробітників атрибути, операції та зв'язки слід розмістити у класі Employee, тоді як атрибути, характерні для класів-нащадків, розміщаються в них.
4. І, нарешті, виявіть класи, які мають багато спільного. Зокрема, якщо є два класи CheckingAccount (Рахунок за вимогою) та SavingsAccount (Рахунок з заощадженнями). Їх дані та поведінка схожі. Для спільних атрибутів двох класів можна створити третій клас Account (Рахунок).

Типовий вигляд діаграми класів у середовищі Rational Rose зображено на рис. 4.9.

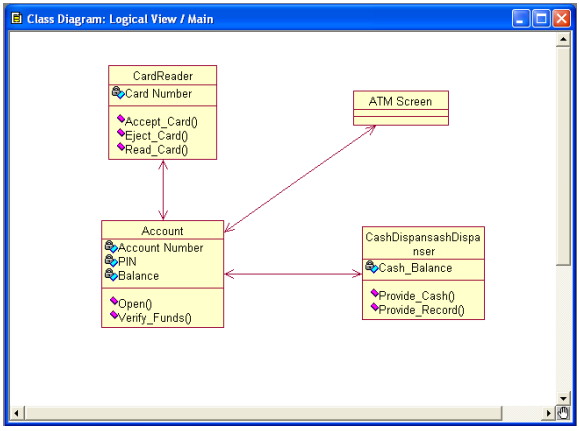
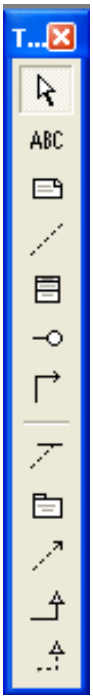


Рис. 4.9. Приклад діаграми класів в середовищі Rational Rose



- стрілка-вказівник для виділення об'єкта
- додає до діаграми текст
- додає до діаграми нотатку
- зв'язує нотатку із сценарієм чи об'єктом на діаграмі
- додає на діаграму новий клас
- додає на діаграму новий інтерфейсний клас
- малює однонаправлену асоціацію
- зв'язує клас асоціацій із зв'язком асоціації
- додає до діаграми новий пакет
- малює зв'язок залежності
- малює зв'язок узагальнення
- малює зв'язок реалізації

Рис. 4.10. Панель інструментів діаграми класів в середовищі Rational Rose

ПРАКТИЧНІ РЕКОМЕНДАЦІЇ

2.1. Створення діаграми Класів

Діаграму Класів можна створювати у Логічному представленні (Logical View) браузера.

Для створення нової діаграми Класів:

1. Клацніть правою кнопкою миші на Логічному поданні браузера.
2. У меню, яке відкрилося, оберіть пункт New>Class Diagram (Створити>Діаграма Класів).
3. Зазначте ім’я нової діаграми.

4. Відкрийте нову діаграму, двічі клацнувши на ній в браузері.

Для відкриття вже створеної діаграми :

1. Знайдіть її у Логічному поданні браузера.
2. Відкрийте її, двічі клацнувши на ній.

Для видалення елемента із діаграми:

1. Виділіть його на діаграмі.
2. Натисніть клавішу Delete.

Для видалення діаграми класів:

1. Клацніть правою кнопкою миші на діаграмі в браузері.
2. У меню, що відкрилося виберіть пункт Delete (Видалити).

2.2. Організація елементів на діаграмі Класів

Для розміщення елементів на діаграмі Класів:

Виберіть в меню пункт Tools>Layout Diagram (Інструменти>Розмістити Діаграму). Всі класи на діаграмі будуть автоматично розміщені найбільш оптимальним способом.

Для зміни розмірів елементів на діаграмі Класів:

Виберіть в меню пункт Tools>Autosize All (Інструменти>Налаштувати розміри всіх елементів). Розмір, що відповідає кожному класу прямокутника буде автоматично змінено так, вмістилось ім'я, атрибути і методи класу.

2.3. Пов'язування файлів і посилань із діаграмою Класів

Для прикріплення файлу до діаграми Класів:

1. Клацніть правою кнопкою миші на діаграмі Класів в браузері.
2. Виберіть пункт New>File (Створити>Файл).
3. В діалоговому вікні відкриття файлу вкажіть файл, що прикріплюється.
4. Для завершення процесу натисніть кнопку Open (Відкрити).

Для прикріплення посилання до діаграми Класів:

1. Клацніть правою кнопкою миші на діаграмі Класів в браузері.
2. Виберіть пункт New>URL (Створити>Посилання).
3. Введіть адресу посилання, що прикріпляється.

2.4. Робота із класами

Додавання класів:

1. Натисніть кнопку Class (Клас) на панелі інструментів.
2. Клацніть мишкою будь-де всередині діаграми Класів. Ваш новий клас матиме ім'я NewClass.
3. Rose відображає список усіх наявних класів. Щоб розмістити на діаграмі клас, двічі клацніть на ньому в списку. Щоб створити новий клас, замініть NewClass в списку на нове ім'я класу. Зверніть увагу, що він з'явиться не лише на діаграмі, але й в Логічному поданні браузера.

Додавання класів за допомогою діаграми Взаємодії:

1. Відкрийте діаграму Послідовності, розроблену в лабораторній роботі № 3.
2. Клацніть правою кнопкою миші на об'єкті діаграми.
3. У меню, що з'явилося, виберіть Open Specification (Відкрити Специфікацію).
4. У спадному переліку класів оберіть пункт New (Новий). Перед вами з'явиться вікно специфікації нового класу**.**
5. У полі Name введіть ім'я вашого класу.

Видалення класу з діаграми Класів:

1. Виділіть його на діаграмі.
2. Натисніть клавішу Delete.
3. Зауважте, попри те, що клас зникає з діаграми, він залишиться на інших діаграмах та в браузері.

Видалення класу з моделі:

1. Виділіть його на діаграмі.
2. У меню моделі оберіть Edit>Delete from Model (Правка>Видалити із моделі) або натисніть комбінацію клавіш CTRL+D.

Для визначення стереотипу класу:

1. Відкрийте вікно специфікації класу.
2. У списку, що розкрився, оберіть потрібний стереотип (Boundary, Entity, Control) або введіть його вручну.

2.5. Робота з атрибутами класів

Додавання до класу властивостей (нових полів):

1. Виділіть клас на діаграмі.

2. Клацніть правою кнопкою миші на класі та оберіть у переліку пункт Open Specification.
3. У вікні, що відкрилося, оберіть вкладку Attributes і натисніть клавішу Insert.
4. Можна додавати атрибути безпосередньо клацнувши на зображенні класу. При цьому потрібно ввести ім'я нового атрибуту (властивості) класу у форматі:

Ім’я : Тип даних = Початкове значення.

Наприклад:

Address: String
IDNumber: Integer =0

Тип даних обирайте відповідно до середовища і мови програмування. Він необхідний для генерування коду програми, водночас початкове значення необов’язкове.

5. Щоб додати ще атрибути, натисніть Enter і введіть нові атрибути безпосередньо на діаграмі Класів.
6. Щоб вказати або змінити тип вже створених атрибутів клацніть правою кнопкою миші на атрибуті в браузері. У меню оберіть пункт Open Specification і перейдіть у вікно специфікації атрибуту. Тип даних оберіть зі списку або введіть самостійно.

Для визначення статусу доступу атрибутів:

1. Клацніть правою кнопкою миші на атрибуті в браузері.
2. Відкрийте специфікацію атрибуту.
3. У полі Export Control оберіть один зі специфікаторів доступу атрибуту: Public, Private, Protected або Implementation. За замовчуванням усі атрибути мають статус доступу Private.

або

1. Виділіть атрибут на діаграмі Класів.
2. Якщо для позначення видимості ви використовуєте нотацію UML, клацніть на позначці «+», «-» або «#» поруч із атрибутом. У списку, що з’явився, оберіть потрібний статус доступу.

2.6. Робота з операціями класів

Додавання операцій (нових методів) до класу:

1. Клацніть правою кнопкою миші на класі діаграми Класів.
2. У вікні, що з'явилося, оберіть пункт New > Operation (Нова> Операція).
3. Введіть ім'я нової операції у форматі:

   Ім'я (Аргумент1: Тип): Тип результату

   Наприклад:

Add(X: Integer, Y: Integer): Integer

4. Якщо ви хочете додати нові операції, натисніть Enter і введіть назву нової операції безпосередньо на діаграмі.

Або

1. Клацніть правою кнопкою миші на класі в браузері.
2. Оберіть пункт New>Operation
3. Під назвою цього класу в браузері з'явиться нова операція opname. Змініть її назву, як для випадку атрибутів.

Задайте область видимості для всіх операцій (див. атрибути).

2.7. Створення зв'язків між класами

1. Оберіть на панелі інструментів кнопку потрібного з'єднання.
2. Зв'язок утворюється між класами, один з яких посилається, або використовує властивості іншого класу, шляхом їх з'єднання.
3. Для кожного кінця з'єднання можна вказати множинність. Для цього потрібно клацнути правою кнопкою миші на з'єднання між класами і у меню, що з'явилося, обрати пункт Multiplicity і потрібне значення.

Порядок виконання роботи

1. Ознайомитися з теоретичним матеріалом до лабораторної роботи.
2. На основі діаграми Послідовності, розробленої в лабораторній роботі № 3, створити діаграму Класів, зокрема:

- визначити об'єкти, які взаємодіють, як використавши попередньо розроблену діаграму Послідовності, так і

додавши, якщо потрібно, нові керуючі об'єкти. **Замініть усі написи кирилицею на відповідні латинські позначення.**

3. Для кожного об'єкта визначити набір атрибутів та операцій, додавши їх до відповідних класів.
4. Встановити зв'язки між класами.
5. Заповнити табл. 5.1, вказавши характеристики зв'язків між класами:

Таблиця 5.1

Батьківський клас	Дочірній клас	Тип зв'язку	Множинність

6. За даними табл. 5.1 утворити зв'язки необхідних типів між класами.
7. Для кожного класу визначити стереотип.
8. Підготувати звіт до лабораторної роботи № 5.

Звіт з лабораторної роботи повинен містити:

1. Діаграму Класів.
3. Заповнену таблицю 5.1.
4. Контекстну діаграму IDEF 0 або DFD для вашого варіанта завдання.
5. Короткі відповіді на запитання для самоперевірки.

Запитання для самоперевірки:

1. Яке призначення діаграми Класів?
2. Назвіть і коротко опишіть три принципи об'єктно-орієнтованого програмування.
3. Що таке клас і як його ідентифікувати на діаграмі Класів?
4. Які основні графічні елементи діаграми Класів? Як вони позначаються і описуються?
5. Які бувають типи зв'язків на діаграмі класів. Наведіть приклади кожного з них. Які відношення використані на вашій діаграмі?
6. Що таке атрибут класу. Як його можна виявити і які характеристики для нього вказуються на діаграмі Класів.

7. Що таке статус доступу елементів класу? На що він впливає і як позначається на діаграмі класів?
8. Чим відрізняються основні стереотипи класів та як їх визначити?
9. Для чого призначена діаграма IDEF0 та які її графічні елементи?
10. Яка діаграма використовується для моделювання сховища даних?

Рекомендовані інформаційні джерела:

1. What is a Class Diagram in UML? URL : https://online.visual-paradigm.com/diagrams/tutorials/class-diagram-tutorial/#multiplicity
2. UML Class Diagram Tutorial. URL : https://www.youtube.com/watch?v=UI6lqHOVHic&ab_channel=LucidSoftware
3. UML Class Diagram Examples. URL : https://www.edrawsoft.com/example-uml-class-diagram.html
4. UML 2 Class Diagramming Guidelines. URL : http://agilemodeling.com/style/classDiagram.htm
5. UML class relationship diagrams. https://www.cs.odu.edu/~zeil/cs330/latest/Public/classDiagrams/index.html
6. Create UML class diagrams. URL : https://www.diagrams.net/blog/uml-class-diagrams

Лабораторна робота № 6. Розрахунок техніко-економічних показників програмного продукту

Мета: набути навичок з визначення основних техніко-економічних показників програмного продукту**:**

- трудомісткість розробки
- продуктивність
- час розробки
- чисельність виконавців
- заробітна платня виконавця та допоміжного персоналу
- витрати на утримання та обслуговування ПК
- собівартість 1-єї машино-години роботи ПК
- собівартість
- вартість програмного продукту.

Постановка задачі

1. Розрахунок трудомісткості та часу розробки програмного продукту (ПП)

Загальний час на створення програми складається з різних компонентів. Загальний часу на створення програмного продукту складові, подані в табл. 6.1

Таблиця. 6.1

Позначення періодів часу на створення програмного продукту

№ етапу	Позначення часу етапу	Зміст етапу
1.	$T_{ПО}$	Підготовка опису завдання
2.	$T_{О}$	Опис завдання
3.	Ta	Розробка алгоритму
4.	$T_{БС}$	Розробка блок-схеми алгоритму
5.	$Tн$	Написання програми мовою програмування
6.	$T_{нт}$	Налагодження та тестування програми
7.	$T_{д}$	Оформлення документації

Час розраховується в годинах, причому $T_{ПО}$ береться за фактично відпрацьованим часом, а тривалість інших етапів обчислюється за умовною кількістю команд Q.

Умовна кількість команд Q визначається за формулою (6.1):

$$Q = q * C, \quad (6.1)$$

де q - коефіцієнт, який враховує умовне число рядків коду в залежності від типу задачі. Значення даного коефіцієнта визначається з табл. 6.2 та 6.3.

Таблиця 6.2

Значення коефіцієнта **q** для визначення кількості команд в програмному коду

Код виду	Назва виду ПЗ	Межі зміни коефіцієнта
1.0	ПЗ загального призначення	до 1500
2.0	ПЗ технології автоматизації програмування та проектування АСУ	від 4500 до 5500
3.0	ПЗ методо-орієнтованих розрахунків	від 2500 до 3500
4.0	ПЗ організації обчислювального процесу	від 3500 до 4500
5.0	ПЗ функціонального призначення	від 1500 до 2500

Таблиця 6.3

Класифікація видів програмного забезпечення (доповнення до табл. 6.2)

Код виду	Назва виду ПЗ	Склад та вміст виду ПЗ
1	2	3
1.0	ПЗ загального призначення	1.1. ПЗ СУБД; 1.2. ПЗ систем ведення лінійних файлів; 1.3. ПЗ ведення БД і лінійних файлів; 1.4. ПЗ інформаційно-пошукових та інформаційно-довідкових систем; 1.5. ПЗ введення інформації;

Код виду	Назва виду ПЗ	Склад та вміст виду ПЗ
1	2	3
		1.6. ПЗ моніторів телеобробки і мереж ПЕОМ; 1.7. ПЗ оточення СУБД, які розширюють можливості існуючих СУБД; 1.8. ПЗ, які розширюють можливості обробки.
2.0	ПЗ технології автоматизації програмування та проектування АСУ	2.1. ПЗ автоматизації проектування для автоматизації проектування різних АСУ; 2.2. ПЗ технології програмування; 2.3. ПЗ автоматизації програмування (для автоматизації процесів обробки та виводу інформації); 2.4. ПЗ, які розширюють існуючі мови програмування для підвищення їх компактності та простоти користування; 2.5. ПЗ загального призначення, функціонально-орієнтовані; 2.6. ПЗ автоматичного програмування.
3.0	ПЗ методо-орієнтованих розрахунків	3.1. ПЗ оптимізаційних розрахунків (забезпечують рішення різного класу задач оптимального планування та управління виробництвом);
3.0	ПЗ методо-орієнтованих розрахунків	3.2. ПЗ статистичного аналізу та прогнозування (для прогнозування техніко-

Код виду	Назва виду ПЗ	Склад та вміст виду ПЗ
1	2	3
		економічних показників, попиту, тощо); 3.3. ПЗ мережевого планування та управління; 3.4. ПЗ загальної математики; 3.5. ПЗ імітаційного моделювання.
4.0	ПЗ організації обчислювального процесу	4.1. Автоматизація процесу введення наборів даних, при забезпечені їх надійного та систематизованого збереження; 4.2. Підвищення продуктивності ПЕОМ і користувачів ПЗ; 4.3. Формування і видача звітів про роботу ПЕОМ; 4.4. Оперативний контроль системи та ресурсів.
5.0	ПЗ функціонального призначення	Для автоматизації обробки економічних даних виділяється ПЗ функціонального призначення. 5.1. ПЗ оперативного управління основним виробництвом; 5.2. ПЗ управління технічною підготовкою виробництва; 5.3. ПЗ бухгалтерського обліку та управління фінансами; 5.4. ПЗ управління кадрами; 5.5. ПЗ, які не ввійшли ні в один із перерахованих вище видів ПЗ.

С – це коефіцієнт, який враховує новизну та складність програми.

Програмні продукти за ступенем новизни можна віднести до однієї з 4-х груп:

- Група А – розробка принципово нових завдань.
- Група Б – розробка оригінальних програм.
- Група В – розробка програм з використанням типових рішень.
- Група Г – разова типова задача.

За ступенем складності програмні продукти можуть бути віднесені до однієї з 3-х груп:

1 – алгоритми оптимізації та моделювання систем;
2 – завдання обліку, звітності та статистики;
3 – стандартні алгоритми.

Коефіцієнт С визначається за табл. 6.4 на перетині показників складності та новизни.

Таблиця. 6.4

Мова програмування	**Група складності**	**Рівень новизни**			
		А	**Б**	**В**	**Г**
Високого рівня	1	1,38	1,26	1,15	0,69
	2	1,3	1,19	1,08	0,65
	3	1,2	1,1	1,00	0,60
Низького рівня	1	1,58	1,45	1,32	0,79
	2	1,49	1,37	1,24	0,74
	3	1,38	1,26	1,15	0,69

Тепер, виходячи з формули (6.1) можна визначити умовне число команд Q.

Трудомісткість розробки програмного продукту (t) визначається за формулою (6.2), люд.-міс.

$$t=3,6*(\eta_{т.р.к})^{1,2}, \qquad (6.2)$$

де $\eta_{т.р.к}$ – число **тисяч** команд програмного коду.

Середня кількість виконавців ($PL_{вик}$) розраховується виходячи з трудомісткості та тривалості розробки ПП за формулою (6.3), люд.:

$$PL_{вик} = t/T, \qquad (6.3)$$

Загальна тривалість розробки ПП (Т) розраховується за формулою (6.4), міс.:

$$T=2,5*t^{0,32}, \quad (6.4)$$

Продуктивність праці групи розробників ПП ($П_р$), вихідних команд/люд.-міс. Визначається за формулою (6.5):

$$П_р=1000*\eta_{т.р.к}/t \quad (6.5)$$

Далі визначаємо час, необхідний для виконання кожного етапу створення програмного продукту:

1) **$Т_{ПО}$ (час на підготовку опису завдання** , год**)** береться за фактом.

2) **То (час на опис завдання,** , год) визначається за формулою:

$$T_0 = \frac{Q*B}{50*K}, \quad (6.6)$$

де В – коефіцієнт урахування змін завдання, коефіцієнт **В** у залежності від складності завдання і кількості змін обирається в інтервалі від 1,2 до 1,5.

К – коефіцієнт враховує кваліфікацію програміста і обирається з таб. 6.5.

Таблиця 6.5

Значення коефіцієнта кваліфікації виконавця

Стаж програміста	Значення коефіцієнта К
до 2-х років	0,8
від 2 до 3 років	1,0
від 3 до 5 років	1,1 – 1,2
від 5 до 10 років	1,2–1,3
Понад 10 років	1,3–1,5

3) **$Т_а$ (час на розробку алгоритму**, год**)** і **$Т_{БС}$ (час на розробку блок - схеми**, год**)** розраховуємо за формулою:

$$T_a = \frac{Q}{50*K}, \quad (6.7)$$

4) **$Т_н$ (час написання програми мовою програмування**, год**)** визначається за формулою:

$$T_H = \frac{Q*1,5}{50*K}, \quad (6.8)$$

5) **$T_{нт}$ (час налагодження та тестування програми**, год**)** визначається за формулою:

$$T_{нт} = \frac{Q * 4{,}2}{50 * K}, \qquad (6.9)$$

6) **$T_д$ (час витрачений на оформлення документації**, год**)** обирається за фактом.

Тепер, знаючи час, витрачений на кожному етапі, можна підрахувати загальний час на створення програмного продукту:

$$T_{заг} = T_{ПО} + T_{о.} + T_а + T_Н + T_{нт} + T_д, \text{ (год)} \qquad (6.10)$$

2. Розрахунок заробітної плати виконавця робіт зі створення програмного продукту

Основна заробітна платня (ЗП, грн) визначається за формулою:

$$ЗП_{осн}^{вик} = \frac{З^1 * K_Т * T_{заг}}{Ч_Р * T_{рд}}\left(1 + \frac{П}{100}\right), \qquad (6.11)$$

де

$З^1$ – мінімальна зарплата 1-го розряду (6700 грн);

$K_Т$ – тарифний коефіцієнт, що відповідає розряду тарифної сітки по якому працює виконавець. Наприклад, розряд інженера-програміста – 9, якому відповідає тарифний коефіцієнт 1,73.

$T_{заг}$ – загальний час на створення програмного продукту (год.), див. формула (10);

$Ч_Р$ – число робочих днів на місяць (в середньому –21 день);

$T_{рд}$. – тривалість робочого дня в годинах (8 год.).

П – відсоток премії, якщо є.

2.1. Додаткова заробітна платня (грн) визначається за формулою:

$$ЗП_{дод}^{вик} = ЗП_{осн}^{вик} * Д, \qquad (6.12)$$

де Д – відсоток додаткової заробітної платні.

Загальна заробітна платня визначається як сума основної і додаткової:

$$З_{заг} = ЗП_{осн}^{вик} + ЗП_{дод}^{вик}, \qquad (6.13)$$

2.2. Розрахунок єдиного соціального внеску

Тепер можна підрахувати єдиний соціальний внесок, який нараховується на заробітну плату і складає 22 %.

$$B_{ЄСВ} = \frac{(ЗП_{осн}^{вик} + ЗП_{дод}^{вик}) * 22}{100} \quad (14)$$

3. Розрахунок витрат на утримання та експлуатацію ПЕОМ

Основою для розрахунку видатків на утримання та експлуатацію ПЕОМ, що відносяться до даного програмного продукту, є **собівартість 1-єї машино-години роботи ПЕОМ**, тобто витрати, які виконуються за годину роботи на комп'ютері при створенні чи експлуатації програми, і визначається за формулою:

$$C_{М.год} = \frac{B_{сум}}{T_{роб}} \text{ (грн./год)}, \quad (6.15)$$

де $B_{сум}$ – сумарні річні витрати (грн), $T_{роб}$ – час роботи комп'ютера, який визначається як добуток кількості робочих днів на час роботи комп'ютера в день (год), помножені на коефіцієнт (0,9) , що позначає ремонт і профілактику обладнання.

$$B_{сум} = B_{ЕН} + B_{М} + B_{проф} + А + ЗП_{осн}^{обсл} + ЗП_{дод}^{обсл} + B_{ЄСВ}^{обсл} \quad (6.15.1)$$

Спочатку ми визначимо **річні витрати кожного компонента собівартості ($B_{сум}$)**, до числа яких входять:

3.1. Витрати на електроенергію:

$$B_{ЕН} = B_{ПК} + B_{ОСВ} \text{ (грн)}, \quad (6.16)$$

де $B_{ПК}$ – витрати електроенергії на роботу ЕОМ, $B_{ОСВ}$ – витрати на освітлення приміщення, які визначаються як:

$$B_{Х} = T_{Роб.} * Ц * Р, \text{ (грн)} \quad (6.17)$$

де $T_{Роб}$ – тривалість роботи за комп'ютером в рік (год.), Ц – вартість 1 кВт електроенергії (1,68 грн), Р – потужність ПК або освітлювальних приладів.

3.2. Витрати на оплату праці (основної та додаткової) працівникам, які забезпечують функціонування ЕОМ. До них належать:

- інженер

- системний програміст
- оператор набору.

Основна заробітна платня ($ЗП_{обсл}^{ОСН}$)для кожного з них визначається за формулою (6.18).

$$ЗП_{осн}^{обсл} = \frac{З^1 * К_Т}{Н_{обсл}}\left(1+\frac{П}{100}\right)*12, \text{ (грн)} \qquad (6.18)$$

де $Н_{обсл}$ – кількість комп'ютерів, які обслуговуються одним працівником.

Додаткова заробітна платня ($ЗП_{дод}^{обсл}$) визначається за формулою (6.12).

Витрати на ЄСВ ($ЗП_{ЄСВ}^{обсл}$) визначаються для кожного працівника обслуговуючого персоналу за формулою (6.14).

3.3. Витрати на витратні матеріали ($В_м$) (папір, CD/DVD-диски, картридж тощо) беруться за фактом і становлять 2 % від балансової вартості обчислювальної техніки (**$В_б$**).

3.4. Витрати на профілактику ($В_{проф}$) становлять 3% від балансової вартості ПЕОМ з периферією ($В_б$).

3.5. Амортизаційні відрахування в рік (А) визначаються як відношення балансової вартості ПЕОМ (**$В_б$**) до кількості років експлуатації ($N_р$):

$$A = B_б / N_р \text{ (грн)} \qquad (6.19)$$

Визначивши загальну суму витрат $B_{сум}$ за формулою (6.15.1), одержимо собівартість 1-єї машино-години роботи ЕОМ за формулою (6.15).

3.6. Витрати на утримання та експлуатацію ПЕОМ

Знаючи собівартість 1 години роботи на ПЕОМ і час створення ПП ($T_{заг}$ за формулою (6.10)), можна визначити **витрати на утримання й експлуатацію ПЕОМ при розробці ПП**:

$$B_{ПП} = C_{М.год} * T_{заг} \text{ (грн)} \qquad (6.20)$$

4. Розрахунок собівартості грн. програмного продукту

Собівартість програмного продукту визначається загальними витратами на виготовлення програмного продукту і обчислюється з використанням таких показників:

1) Основна заробітна плата виконавця робіт зі створення програмного продукту (грн).
2) Додаткова заробітна плата виконавця робіт зі створення програмного продукту (грн).
3) Нарахування на заробітну плату (єдиний соціальний внесок).
4) Витрати на утримання і експлуатацію ПЕОМ, що відносяться до програмного продукту.

Тепер, додавши значення всіх елементів, можна визначити **собівартість (грн.) програмного продукту**:

$$C_{ПП} = ЗП_{осн}^{вик} + ЗП_{дод}^{вик} + В_{ЄСП} + В_{ПП}, \text{ (грн)} \qquad (6.21)$$

5. Розрахунок вартості (ціни) програмного продукту.

Вартість (ціна) програмного продукту, запропонована розробником, визначається за формулою:

$$Ц = C_{ПП} * \left(1 + \frac{P}{100}\right), \text{ (грн)} \qquad (6.22)$$

де $C_{ПП}$ – загальні витрати на створення програмного продукту (грн.), Р – рентабельність розробки (40 %).

Порядок виконання роботи

1. Обчислити наведені вище техніко-економічні показники для програмного забезпечення, показники якого потрібно обирати в таблиці 6.7, відповідно до номера варіанта.
2. Результати розрахунків записати до табл. 6.6.

Таблиця 6.6

Результати обчислення техніко-економічних показників програмного продукту

Показник	**Значення**
Кількість команд вихідного коду (команд)	
Трудомісткість, (люд.-міс.)	
Продуктивність (вих.ком./люд.-міс.)	
Кількість виконавців (люд.)	
Час, потрібний на створення програмного продукту (міс.)	
Основна заробітна платні виконавця (грн)	
Додаткова заробітна платня (грн)	

Загальні витрати на утримання і експлуатацію ПЕОМ при виконанні проекту (грн)	
Собівартість програмного продукту (грн)	
Вартість готового програмного продукту (грн)	

Звіт з лабораторної роботи повинен містити

1. Вихідні дані програмного продукту, обрані в табл. 6.7 відповідно до номера варіанта.
2. Розрахунки техніко-економічних показників у вигляді: початкових формул, формул з підставленими значеннями параметрів та результатом обчислення із зазначенням одиниць вимірювання (для результату).
3. Заповнену табл. 6.6. з отриманими результатами обчислень
4. Короткі відповіді на контрольні запитання.

Запитання для самоперевірки:

1. Що таке метрика з точки зору якості програмного продукту? Яке її призначення?
2. Назвіть основні метрики програмного продукту і наведіть їх приклади.
3. Що таке трудомісткість? В чому вона вимірюється і як визначається?
4. Від чого залежить тривалість розробки програмного продукту і кількість задіяних при цьому виконавців? Як позначається перевищення і скорочення терміну виконання проекту на його вартості?
5. Чим визначається основна заробітна плата виконавця проекту?
6. Що таке собівартість програмного продукту і які показники її визначають?
7. Як визначити ціну готового програмного продукту?

Таблиця 6.7

Вихідні дані для обчислення техніко-економічних показників програмного продукту

Показники	В.1	В.2	В. 3	В.4	В.5	В.6	В.7	В.8	В.9	В.10
1.Час підготовки опису завдання, год	30	35	32	25	33	30	37	36	40	50
2. Типи задачі	Облік	Оперативне упвління	Планування	Багаторівневі задачі	Комплексні задачі	Оперативне управління	Планування	Багаторівневі задачі	Облік	Комплексні задачі
3. Рівень новизни ПП	А	Б	В	Г	Г	В	Б	А	В	Г
4. Рівень складності ПП	1	2	3	1	2	3	1	2	3	1
5. Рівень мови програм.	високий	низький	високий	високий	високий	низький	високий	низький	високий	низький
6. Стаж програміста	1,5	2	2,5	3	3,5	4	4,5	5	6	10
7. Час на оформлення документації, год	40	45	42	35	25	43	48	20	30	50
8. Тарифний коефіцієнт	1,73	1,82	1,82	1,73	1,73	1,82	1,73	1,82	1,7	1,82
9. Місяць початку розробки проекту	січень	лютий	грудень	листопад	жовтень	вересень	серпень	липень	червень	травень
10. Середня тривалість робочого дня	7,9	7,5	7,8	7,7	8	7,95	7,85	7,6	7,8	7,99
11. Середній відсоток премії	10	11	12	13	14	15	16	17	18	20

Показники	В.1	В.2	В. 3	В.4	В.5	В.6	В.7	В.8	В.9	В.10
12. Відсоток додаткової заробітної праці	10	14	11	9	12	10	10	15	14	13
13.Сумарна потужність ПЕОМ, кВт/год	1,3	1,25	1,2	1,15	1,1	1,05	1	0,9	0,8	0,85
14.Сумарна потужність, яка йде на освітлення, кВт/год	0,7	0,6	0,5	0,4	0,7	0,6	0,5	0,4	0,3	0,7
15. Щомісячна зарплатня інженера 1-го розряду , грн	6850	6950	6900	6715	6810	6680	6910	7100	6650	6780
16. Щомісячна зарплатня системного програміста, грн	7400	8250	7350	7280	7200	7450	7310	8110	8115	7750
17. Щомісячна зарплатня оператора , грн	7085	6634	7280	7350	7260	7300	7324	7380	7420	7450
18. Норма обслуговування (інженер)	13	14	16	18	17	16	15	14	13	12
19. Норма обслуговування (системний програміст)	26	25	24	23	22	21	20	19	26	25
20. Норма обслуговування (оператор)	9	10	11	12	13	14	15	9	10	11

Показники	В.1	В.2	В. 3	В.4	В.5	В.6	В.7	В.8	В.9	В.10
21. Тарифний коефіцієнт (інженер)	1,54	1,64	1,7	1,54	1,64	1,7	1,54	1,64	1,7	1,64
22. Тарифний коефіцієнт (системний програміст)	1,54	1,6	1,7	1,54	1,6	1,7	1,54	1,6	1,7	1,54
23. Тарифний коефіцієнт (оператор)	1,36	1,45	1,36	1,45	1,36	1,45	1,36	1,45	1,4	1,45
24. Балансова вартість комплексу ПЕОМ, грн	23000	21000	19000	17000	15000	13000	11000	21000	22000	18000
25. Термін використання ПЕОМ, роки	3	4	5	6	7	8	9	10	3	4

Лабораторна робота № 7. Тестування програмного продукту

Мета: розробити тест-кейси та створити сценарії для проведення тестування програмного забезпечення.

Постановка задачі

Тест – це набір з одного або декількох тест-кейсів (перекладається як варіант використання для перевірки або тестовий випадок).

Тест-кейс (*test case*) – це набір вхідних даних, умов виконання та очікуваних результатів, розроблений задля перевірки деякої властивості поведінки програмного забезпечення. Тест-кейс офіційно фіксується у вигляді відповідного документу.

Високорівневий тест-кейс (*high level test case*) – тест-кейс без конкретних вхідних даних і очікуваних результатів. Зазвичай обмежується загальними ідеями та операціями використання. Часто застосовується при інтеграційному та системному тестуванні, а також на рівні димового тестування (*smoke testing*). Може служити відправним пунктом для дослідницького або більш низькорівневого тестування.

Низькорівневий тест-кейс (*low level test case*) – тест-кейс із конкретними вхідними даними та очікуваними результатами. Являє собою повністю готовий до виконання тест-кейс і вважається найбільш класичною формою для проведення тестування. Найкращий варіант для тестувальників -початківців, оскільки краще створювати тести, у яких всі дані детально описані і не потрібно вирішувати якою інформацією можна знехтувати, не зменшуючи цінність тест-кейса.

Специфікація тест-кейса (*test case specification*) – документ, що описує набір тест-кейсів (зокрема їх цілі, вхідні дані, умови та кроки виконання, очікувані результати) для елемента, який тестується (*test item, test object*).

Специфікація тесту (*test specification*) – документ, який складається зі специфікації проекту тесту (*test design specification*), специфікації тест-кейса (*test case specification*)

та/або специфікації тест-процедури (*test procedure specification*).

Тест-сценарій (*test scenario, test procedure specification, test script*) – документ, який містить послідовність дій, які виконуються при виконанні тесту (також відомий як «тест-скрипт»).

Мета написання тест-кейсів

Тестування можна проводити і без тест-кейсів. Ефективність такого підходу може бути різною та залежить від досвіду та винахідливості тестувальника. Наявність тест-кейсів дає змогу:

- Структурувати та систематизувати підхід до тестування (без чого великий проект майже гарантовано приречений на невдачу).
- Обчислювати метрики тестового покриття (*test coverage metrics*) і вживати заходів аби його збільшити. При цьому тест-кейси становлять головне джерело інформації, без якого існування таких метрик втрачає зміст).
- Відстежувати чи відповідає поточна ситуація плану: скільки приблизно тест-кейсів знадобиться, скільки вже виконано із запланованих та ін.).
- Налагодити взаєморозуміння між замовником, розробниками і тестувальниками (тест-кейси почасти більш наочно демонструють поведінку програми, аніж це описано у вимогах).
- Зберігати інформацію для використання у майбутньому та обміну досвідом між співробітниками і командами (або, щонайменше, не намагатися утримати в голові всі випадки, результати та сотні сторінок сценаріїв).
- Проводити регресивне тестування і повторне тестування, які без тест-кейсів взагалі неможливо було б виконати.
- Підвищувати якість вимог до ПЗ.
- Швидко вводити у курс справи нового співробітника, який долучається до проекту.

Атрибути тест-кейса

Як вже зазначалося вище, термін «тест-кейс» може позначати формальний запис випадків для тестування у вигляді технічної документації. Цей запис має загальновизначену структуру, компоненти якої називають атрибутами тест-кейса. Залежно від інструмента керування тест-кейсами зовнішній вигляд їх запису може дещо змінюватися.

Найчастіше до атрибутів тест-кейса належать:

Ідентифікатор (*identifier*) – унікальне значення, яке однозначно відрізнятиме один тест-кейс від іншого та використовується при зверненнях та посиланнях. Загалом це може бути деяке унікальне ціле значення, проте залежно від інструментарію керування тест-кейсами це позначення може містити префікси, суфікси та інші змістовні компоненти, які допомагають швидко визначити ціль тест-кейса і частину застосунку (або вимоги), яких він стосується (наприклад, R216_DB_Neg).

Пріоритет (*priority*) позначає важливість тест-кейса. Його можна подавати літерами (A, B, C, D, E), цифрами (1, 2, 3, 4, 5), словами («надвисокий», «високий», «середній», «низький», «вкрай низький») або у будь-який інший зручний спосіб. Кількість градацій має бути уніфікована та найчастіше налічує від трьох до п’яти значень. Пріоритет визначається за важливістю вимоги, користувацького сценарію або функції, з якими пов’язано тест-кейс; потенційною важливістю дефекту або наслідками, які він може мати для решти програми;

Вимога, пов’язана з тест-кейсом (*requirement*) позначає ту цільову вимогу, перевірці якої присвячено тест-кейс.

Модуль і підмодуль застосунку (*module and submodule*) вказує на частину програми, якої стосується тест-кейс, і допомагає краще з’ясувати її ціль.

Таблиця 7.1

Приклади заголовків тест-кейсів

Невдалий варіант	**Гарне рішення**

Тест 1	Запуск, одна копія, відповідні параметри
Тест 2	Запуск однієї копії за неправильними шляхами
Тест 78 (покращений)	Запуск, багато копій, без конфліктів
Зупинка	Зупинка за допомогою Ctrl+C
Закриття	Зупинка шляхом закриття консолі
…	…

Заголовок (зміст) тест-кейса (*title*) покликаний спростити і пришвидшити основні ідеї та зусилля тест-кейса без звернень до решти атрибутів. Саме це поле є найбільш інформативним при перегляді переліку тест-кейсів (табл. 7.1).

Заголовок тест-кейса може бути повноцінним реченням, фразою, набором словосполучень, яке має бути інформативним і унікальним.

Кроки тест-кейса (*steps*) описують послідовність дій, які необхідно реалізувати під час виконання тест-кейса. Загальні рекомендації зі створення таких кроків:

- розпочинати зі зрозумілого і очевидного місця, не писати зайвих початкових кроків, як-от запуск застосунку, очевидні операції з інтерфейсом, тощо;
- навіть якщо тест-кейс складається лише з одного кроку, нумерувати його, інакше зростає ймовірність у майбутньому приєднати цей крок до нового тесту);
- при описі українською мовою використовувати безособові форми, наприклад «відкрити», «ввести», «додати» замість «відкрийте», «введіть», «додайте»), при використанні англійської мови не потрібно використовувати частку «to» (тобто «start application», а **не** «to start application»);
- узгоджувати рівень деталізації кроків та їх параметрів із ціллю тест-кейса, його складністю, рівнем, тощо. Залежно від цих та інших факторів деталізація може

стосуватися як загальних ідей і сягати максимально чітко прописаних значень і вказівок;

- посилатися на попередні кроки та їх межі для скорочення обсягу тексту (наприклад, «повторити кроки 3–5 зі значенням…»);
- описувати кроки послідовно, без умовних конструкцій типу «якщо…тоді…».

Очікувані результати (*expected results*) на кожному кроці тест-кейса описувати реакцію програми на дії, описані в полі «кроки тест-кейса». Номер кроку відповідає номеру результату.

Щодо опису очікуваних результатів існують такі рекомендації:

- описувати поведінку системи так, щоб виключити суб'єктивне тлумачення (наприклад, невдалий варіант – застосунок працює коректно», прийнятний варіант – «з'являється вікно із повідомленням …»);
- вказувати очікуваний результат для всіх кроків без винятку, навіть якщо результат деякого кроку здається очевидним і правильним за замовчуванням;
- писати коротко, проте не на шкоду інформативності;
- уникати умовних конструкцій типу «якщо … тоді…».

Порядок виконання роботи:

1. Для програмної розробки, виконаної у рамках курсової роботи, кожен учасник команди повинен розробити хоча б один тест-кейс для розгорнутої перевірки функціональної вимоги, наведеної у технічному завданні до роботи.
2. Для кожного тест-комплекту потрібно вказати (див. Додаток Ж):
   - Його умовну назву.
   - Функціональну вимогу, яка перевірятиметься, згідно зі специфікацією.
   - Пріоритет.
   - Автора тест-кейса.

- Загальний опис тест-кейса (що перевірятиметься).
- Інформацію, необхідну для виконання перевірки
- Два сценарії проведення перевірки: **позитивний і негативний**, із зазначенням послідовності кроків і очікуваних результатів.
- Результат перевірки.

3. У звіті потрібно навести як тест-кейс, розроблений кожним учасником команди, так і результати перевірки у вигляді екранних форм програми.
4. *При тестуванні інтерфейсу користувача* за позитивним і негативним сценарієм зверніть увагу на перевірку правильності введення інформації (набір символів, довжину, формат введення, правильність відображення інформації, тощо), а також повідомлення, які генерує система.
5. *Виконайте тестування сумісності* для перевірки поводження системи на різних ОС і у різних браузерах. Продемонструйте результати перевірки на екранних формах програми. Зазначте ОС і браузери, у яких ваша розробка працює стабільно без спотворень інформації.

Звіт з лабораторної роботи повинен містити:

1. Розроблені тест-комплекти кожного учасника команди у відповідному оформленні.
2. Результати перевірок, виконаних за позитивних і негативним сценаріями з поясненнями і екранним формами програми.
3. Результати перевірки інтерфейсів користувача з поясненнями і екранним формами програми.
4. Результати тестування сумісності із зазначенням ОС та/або браузерів, для яких призначена програма.
5. Загальні висновки щодо результатів проведених тестувань.

Одержані у цій лабораторній роботі результати слід долучити до документації до курсової роботи, у розділі «**Програма і методика випробувань**».

Запитання для самоперевірки:

1. Звідки походять програмні помилки та на що вони можуть впливати при виготовленні програмного продукту?
2. Порівняйте як і коли виконується тестування у найбільш відомих моделях життєвого циклу програмної розробки, як-от водоспадна, спіральна, V-модель, Agile.
3. У чому різниця між QA, QC і тестуванням?
4. Що таке тест-кейс? Яка його структура і призначення?
5. У чому різниця між функціональним та нефункціональним тестуванням?
6. Чи варто програмісту, який написав код, доручати його тестування?
7. Які типи тестування вирізняють за знанням внутрішньої структури програми та доступом до вихідного коду?
8. Чи завжди перевірка стосується роботи системи загалом? Які ще існують різновиди тестування за рівнем деталізації?
9. Як оцінюється ефективність процесу тестування та виявлення дефектів?
10. Як переконатися у коректності роботи програми у середовищі користувача?

Рекомендовані інформаційні джерела:

1. Writing test cases: examples, best practices & test case templates. URL : https://www.testmo.com/guides/writing-test-cases-examples
2. Software testing tutorial. URL : https://www.guru99.com/software-testing.html
3. Software testing in continuous delivery. URL: https://www.atlassian.com/continuous-delivery/software-testing
4. Звідки беруться помилки в ПЗ. URL : https://qalight.ua/baza-znaniy/zvidki-berutsya-pomilki-v-pz/
5. Авраменко А.С. Тестування програмного забезпечення: навч. посібник / Авраменко А.С., Авраменко В.С., Косенюк Г.В. – Черкаси : ЧНУ імені Богдана

Хмельницького, 2017. 284 с. URL : http://eprints.cdu.edu.ua/1482/1/testyvan.pdf
6. Types of software testing. URL : https://www.geeksforgeeks.org/types-software-testing/
7. QA, QC and testing – The basics of software quality management. URL : https://www.altexsoft.com/whitepapers/quality-assurance-quality-control-and-testing-the-basics-of-software-quality-management/
8. The ultimate guide to software testing. URL : https://www.globalapptesting.com/blog/software-testing
9. Different types of testing in software. URL : https://www.perfecto.io/resources/types-of-testing

ДОДАТОК А.

ВАРІАНТИ ЗАВДАНЬ ДО КУРСОВИХ ТА ЛАБОРАТОРНИХ РОБІТ З ДИСЦИПЛІНИ «ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»

ВАРІАНТ 1. Програмний модуль «Органайзер». Прикладна програма передбачається для запису, зберігання і пошуку контактів, а також автоматизації розкладу подій, як-от зустрічей, доручень, заходів. Для кожної події зазначається її тип, місце проведення, дата, час початку та кінця події, відповідальну особу. Слід передбачити формати введення дати та часу та перевіряти правильність внесених даних з метою уникнення накладок. Необхідно організувати нагадування напередодні запланованої події. На основі введених даних програма повинна автоматично формувати розклад подій за вказаний період (на поточний день чи місяць).

ВАРІАНТ 2. Програмний модуль «Особисті дані студентів». Розробити програмний модуль, передбачений для отримання відомостей про студентів певної групи парламентарем, куратором та співробітником деканату. Список групи та куратора визначає працівник деканату. Куратор групи може переглядати, додавати і редагувати дані про студентів своєї групи. Необхідно зберігати інформацію про адресу проживання та контактні телефони студентів, захоплення та громадські доручення. Окремо слід виконувати облік студентів, які мешкають у гуртожитку. Парламентар повинен вводити дані про кількість пропущених занять в кінці місяця та формувати підсумків з відвідування за звітний період. Також необхідно виконувати облік спортивних, громадських і наукових досягнень, та систему оцінювання цих видів діяльності при нарахуванні стипендії з можливістю формування звіту.

ВАРІАНТ 3. Програмний модуль «Облік виконання лабораторних робіт». Розробити програмний модуль для обробки результатів виконання лабораторних робіт студентами групи.. Користувачами програми є викладачі, які проводять заняття, та студенти, яких цікавлять власні поточні результати. Дана розробка повинна зберігати інформацію про дисципліни, з

яких проводять лабораторні роботи, список студентів, загальний і з розбиттям на підгрупи, та викладачів, які проводять лабораторні заняття. Розподіл викладачів і предметів для групи виконує секретар кафедри. Кожна дисципліна має свій набір лабораторних робіт. До кожної лабораторної роботи викладач повинен внести дані про її номер, назву, короткий опис та максимальну кількість балів. Слід реалізувати електронний журнал виконання лабораторних робіт з різних дисциплін кожним студентом групи зі зазначенням дати виконання та отриманої суми балів. Слід автоматизувати процес формування підсумкових відомостей по результатах виконання лабораторних робіт з визначеного предмета, обчислення середнього та підсумкового балу за вказаний період із можливістю формування звітів. Забезпечте викладачеві функції редагування та видалення даних, а також здійснення пошуку за прізвищем студента та назвою дисципліни. Передбачте контроль за правильністю внесення даних. Студентові програма повинна надавати можливість перегляду власних результатів.

ВАРІАНТ 4. Програмний модуль «Облік успішності студентів». Розробити програмний модуль, передбачений для оперативного обліку поточних результатів навчання студентів по різних дисциплінах працівниками деканату та викладачами. Слід враховувати оцінки за кредитно-модульною системою, як поточні за модулі так і підсумкові. Кожна дисципліна закріплена за викладачем, який визначає кількість модулів, розбиття на бали та графік проведення модульних контрольних. Працівник деканату формує список груп та перелік предметів на семестр, а також розподіляє предмети між викладачами. Викладач може переглядати та редагувати інформацію по своїх предметах, а також заповнювати модульні відомості. Студенти мають змогу переглядати перелік предметів на семестр, викладачів, які проводять заняття та власні результати оцінювання по дисциплінах, які вони вивчають або вивчали.

ВАРІАНТ 5. Програмний модуль «Екзаменаційна сесія». Розробити програмний модуль для збереження, обліку та перегляду результатів сесії. Користувачами системи є студенти

академічної групи, викладачі та працівник деканату. Працівник деканату закріплює предмети між викладачами, формує список групи, розклад заліків та екзаменів, а також може переглядати підсумкові результати сесії. Викладачеві слід забезпечити функцію внесення оцінок студентів по предмету, з якого він/вона приймає залік або іспит. Оцінки потрібно вносити у 100-бальному форматі та здійснювати автоматичне перетворення за визначеною шкалою п'ятибальної та кредитно-модульної системи ECTS. **Передбачте захист від некоректного введення балів**. Розроблена система повинна надавати можливості працівнику деканату автоматичного формувати відомості за результатами іспитів. Студенту потрібно забезпечити перегляд розкладу заліків та іспитів, а також власних результатів навчання.

ВАРІАНТ 6. Програмний модуль «Телефонна служба». Дана програмна розробка повинна містити відомості про номери телефонів та їх власників (ПІБ, адреса, паспортні дані), тарифи та послуги, які надаються користувачам цифрової телефонної мережі. Адміністратор системи виконує облік працівників телефонної служби, а також оформляє нових користувачів телефонної мережі. Оператор АТС здійснює облік оплати за телефонні послуги. У програмі є можливість перегляду заборгованості по оплаті, набрані номери та тривалість дзвінків для конкретного абонента. Після сплати за послуги автоматично формується квитанцію із зазначенням загальної суми, періоду, за який здійснюється оплата, даними абонента та оператора. Абоненту доступна послуга перегляду вартості послуги за місяць при введенні номеру телефону.

ВАРІАНТ 7. Програмний модуль «Облік порушень правил дорожнього руху». Дана програмна розробка передбачається для використання працівниками поліції та водіями. Адміністратор системи повинен зберігати, редагувати та здійснювати пошук даних про автомобілі (марка, номер, номер кузова, рік випуску тощо), їх власників (ПІБ, адреса, номер посвідчення водія, категорія тощо), а також можливі дорожньо-транспортні пригоди та штрафи, які вони передбачають. Працівник поліції може здійснювати пошук даних про автомобіль або водія та фіксувати

порушення. За номером транспортного засобу необхідно автоматично визначати дані його власника. Також при оформленні порушень слід зазначати дату, час, місце пригоди, учасників, тип порушення і розмір штрафу. Окремо адміністратор повинен проводити облік сплати штрафу . Водій може користуватися системою, попередньо пройшовши реєстрацію. Після аутентифікації, водій може переглядати особисті дані, наявні у нього правопорушення та інформацію про їх оплату.

ВАРІАНТ 8. Програмний модуль «Агентство нерухомості». Дана програмна розробка передбачена для використання робітниками агентства з купівлі, продажу та оренди нерухомості, а також всіма зацікавленими клієнтами. Адміністратор системи виконує облік фірм, які займаються нерухомістю, та операцій з нерухомістю (купівля, продаж, оренда). Фірми можуть розміщувати та редагувати відомості про наявні приміщення та детальну інформацію про них. Для всіх користувачів системи слід передбачити можливість перегляду загальної інформації про наявні приміщення та реалізувати пошук за певними критеріями. Детальна інформація про об'єкт нерухомості, а також можливість його бронювання, надаються лише зареєстрованим користувачам, Після підтвердження заявки на операцію з нерухомістю між агентством та клієнтом укладається договір, із зазначенням всіх деталей приміщення, терміну дії договору, осіб, які його укладають та грошової суми. Договір потрібно зберігати в електронному вигляді з можливістю друку.

ВАРІАНТ 9. Програмний модуль «Продаж залізничних квитків». Дана програма дозволяє адміністратору системи зберігати та редагувати інформацію про залізничні маршрути, розклад руху потягів до місця призначення, типи квитків та їх вартість. Касир залізничної каси має змогу переглядати інформацію, введену адміністратором, виконувати пошук маршруту до пункту призначення, здійснювати продаж квитків на задану дату з наступним внесенням інформації до системи. Користувачі програми так само можуть виконувати пошук та перегляд інформації про маршрути (в загальному або на

визначені дату та час) та замовляти електронний квиток, після проходження реєстрації. Копія оформленого квитка зберігається в текстовому документі та може бути відправлена на друк.

ВАРІАНТ 10. Програмний модуль «Книжковий магазин», який містить відомості про книжки (автор, назва, видавництво, рік видання, ціна, кількість екземплярів), фірми постачальників. Адміністратор магазину може вносити дані про нові поступлення товару, редагувати інформацію про вже оформлену продукцію, здійснювати розподіл книжок по категоріях та оформляти договори на постачання продукції з фірмами-постачальниками. Також в програмі необхідно реалізувати електронний каталог, за допомогою якого покупець матиме можливість відшукати потрібну книжку. Також для зареєстрованих користувачів потрібно надати можливість замовлення книжок. При купівлі книжки автоматично формується квитанція. Програма повинна надавати можливість адміністратору автоматично підраховувати суму виторгу за потрібний період, а також формувати звіт з продажу продукції в кінці кожного місяця.

ВАРІАНТ 11. Програмний модуль «Авіа каса». Дана програмна розробка використовується працівниками аеропорту, касирами та клієнтами. Програма повинна надавати адміністратору можливість зберігати та редагувати дані про розклад авіа рейсів, маршрути, категорії квитків та їх вартість. Касир повинен проводити облік продажу квитків, для кожного рейсу надавати інформацію про наявність вільних місць та вартість авіаквитків (для різного класу). Клієнти програми мають змогу переглядати розклад авіа рейсів, здійснювати пошук потрібного рейсу на заданий час, дату чи до потрібного пункту призначення. Для перегляду більш детальної інформації, а також бронювання квитків користувачеві потрібно зареєструватися. При оформленні замовлення безпосередньо або через касира необхідно автоматизувати формування та збереження квитків.

ВАРІАНТ 12. Програмний модуль «Поліклініка» Дана програмна розробка передбачена для використання працівниками та пацієнтами. Програма повинна зберігати загальну інформацію про лікарів, пацієнтів, кабінети та

лікувальні процедури. Адміністратор виконує в програмі формування розкладу роботи лікарів, облік кабінетів, наявних лікувальних процедур. Кожен лікар повинен здійснювати облік звернень від пацієнтів та прописувати лікування. Зі свого боку, пацієнт має змогу переглядати розклад роботи лікарів та кабінетів, здійснювати пошук потрібної інформації (наприклад, за прізвищем лікаря). Запис на прийом до лікаря дозволений лише зареєстрованим користувачам, які також можуть переглядати деталі історії хвороби, призначене лікування та результати аналізів. Необхідно автоматизувати видачу пацієнтам довідок та рецептів.

ВАРІАНТ 13. Програмний модуль «Кадрове агентство». Дана програмна розробка повинна надавати відомості про фахівців, які шукають роботу, їх спеціальність, резюме, фото, тощо. Також потрібно зберігати та обробляти дані про фірми, підприємства та організації, які запрошують на роботу, а також вимоги, які вони висувають до претендентів. Для подачі заявки на пошук/прийом на роботу потрібно пройти реєстрацію. За допомогою програми працівник агентства обробляє подані заявки та здійснювати пошук відповідних варіантів. Необхідно надавати інформацію про вакансії для потенційних робітників. Зареєстрований користувач може самостійно виконувати пошук роботи і має можливість через програму звернутися до потенційного роботодавця. Останній, зі свого боку, може переглядати і обробляти заявки, які до нього надійшли. Потрібно проводити облік заповнених вакансій і передбачити автоматичне формування договору між роботодавцем та працівником.

ВАРІАНТ 14. Програмний модуль «Бібліотека». Користувачами даної програмної розробки є адміністратор бібліотеки, бібліотекари та читачі. За допомогою цієї інформаційної системи адміністратор повинен мати змогу виконувати облік друкованих видань, працівників та читачів бібліотеки, формувати розклад роботи бібліотекарів, розподіляти друковані видання по різних читальних залах. Бібліотекар повинен вести облік видачі та повернення книжок, пошук необхідного видання за назвою, автором чи тематикою. а також

оформляти нових читачів. Необхідно відстежувати процес заборгованостей по книжках на конкретну дату. Користувач системи має можливість переглядати електронний книжковий каталог з метою пошуку потрібної книги. Також, необхідно реалізувати функцію замовлення книжки, для чого потрібно пройти попередню реєстрацію в системі за наявності читацького квитка.

ВАРАНТ 15. Програмний модуль «Автосервіс». Розробити програмний модуль, який використовується працівниками автосервісу та його клієнтами. Адміністратор системи вносить та редагує дані про робітників автосервісу, постійних клієнтів, наявні послуги та їхню вартість. Працівник автосервісу за допомогою програми може фіксувати надходження заявок, здійснювати облік виконання послуг та видачі квитанцію для оплати виконаних робіт. Реєстрація клієнтів проводиться одноразово при першому зверненні до автосервісу. Програма повинна автоматично формувати звіт про виконані роботи за визначений період. Слід передбачити можливості внесення, видалення та редагування даних, що зберігаються. Клієнти мають можливість відстежувати подані заявки та слідкувати за станом їх виконання. Після виконання ремонту потрібно формувати квитанцію на сплату послуг з автоматичним внесенням деталей виконаних робіт та працівника сервісу.

ВАРІАНТ 16. Програмний модуль «Служба доставки». Розробити програму, яка б дозволяла автоматизувати процес оформлення та перевезення вантажів різного типу по Україні. У програмі слід передбачити авторизований доступ для представників фірми-перевізника та користувачів-замовників. Дана розробка повинна надавати можливість працівнику служби доставки виконувати облік вантажів та складських приміщень, до яких доставляється вантаж. Також окремо слід автоматизувати розрахунок вартості доставки, яка залежить від ваги та способу доставки (на склад або за адресою замовника) та її термінів. Зареєстровані користувачі можуть самостійно або через працівників служби подавати заявку на доставку вантажу, відстежувати виконання перевезення, переглядати інформацію

про оформлені перевезення у особистому кабінеті. Фірма-перевізник повинна слідкувати через систему за заявками, що надходять, призначати виконавців та формування звіт про доставку за довільний звітний період.

ВАРІАНТ 17. Програмний модуль «Готель». Даний програмний модуль повинен забезпечувати багатокористувацький доступ до інформації про послуги, які надаються готельно-ресторанним комплексом, зокрема: типи номерів, їх фото та вартість, послуги, які надаються клієнтам (наприклад, замовлення обідів, екскурсії, туристичні походи, кінні прогулянки, катання на лижах, відвідування басейну, тощо). Слід передбачити бронювання номерів та замовлення послуг для клієнтів лише після попередньої реєстрації. Адміністратор готелю повинен мати змогу проводити облік номерів та журнал поселення клієнтів, схвалювати або відхиляти бронювання, відстежувати оплату послуг. При виселенні з готелю клієнтові необхідно видавати квитанцію на основі даних проживання.

ВАРІАНТ 18. Програмний модуль «Інтернет-магазин». Дана програмна розробка пропонує товари (довільного типу), супроводжуючи їх фотографією, коротким описом та ціною. Клієнти можуть переглядати інформацію про товар, здійснювати параметричний пошук, додавати товари до корзини. Для оформлення замовлення необхідно зареєструватися, після чого можна уточнити кількість товарів, спосіб доставки та оплати. Працівники магазину також можуть здійснювати оформлення замовлення. Окрім іншого продавцеві доступна додаткова інформація про товар, як-от кількість одиниць на складі. Адміністратор системи контролює (вносить, редагує) інформацію про постачальників та продукцію. Клієнт може відстежувати замовлення, а після отримання автоматично сформувати чек до сплати.

ВАРІАНТ 19. Програмний модуль «Кабельне телебачення». Дана програмна розробка передбачена для обслуговування абонентів міської кабельної мережі. Програму контролює адміністратор, який здійснює облік наявних пакетів програм, телевізійних каналів, вартість пакета та термін використання.

Також необхідно обробляти облікові дані абонентів, відстежувати оплату послуг, одержувати заявки та перевіряти можливості підключення нових користувачів. Адміністратор повинен мати змогу додавати нових користувачів, видаляти та редагувати облікові записи оформлених абонентів. Для роботи з програмою користувач спершу повинен зареєструватися. Необхідно передбачити можливість формування заявки на підключення пакету для майбутніх абонентів. Зареєстрованим абонентам потрібно надати можливість зворотного зв'язку із службою підтримки.

ВАРІАНТ 20. Програмний модуль «Автовокзал». Дана програмна розробка призначена для автоматизації роботи автостанції. До функціональних можливостей, які дана програмна розробка забезпечує адміністратору автовокзалу, належать: облік перевізників, транспортних засобів, маршрутів, формування розкладу руху. Оператор-касир може переглядати інформацію, яка стосується транспортних перевезень, та здійснювати пошук рейсів за вказаним напрямком та на визначену дату, а також виконувати продаж квитків. Клієнтам повинен бути доступний розклад руху, наявність кількості вільних місць, інформації про вартість та тривалість подорожі. Замовлення квитка передбачає попередню реєстрацію з автоматичним формуванням електронного квитка, який можна зберегти або роздрукувати.

ВАРІАНТ 21. Програмний модуль «Туристична фірма». Дана програмна розробка повинна автоматизувати роботу туристичного агентства. Для менеджера турфірми слід передбачити функції внесення та редагування інформації про запропоновані тури, визначення програми подорожі, вартості послуг, розкладу виїздів груп. Також необхідно обробляти інформацію про готелі, в яких можуть проживати туристи. Клієнти, які використовують програму, повинні мати змогу переглядати інформацію про запропоновані тури та актуальні пропозиції. Оформлення заявки на тур доступне лише зареєстрованим користувачам. Менеджер програми схвалює заявки клієнтів та відстежує облік оплати послуг. Після

підтвердження замовлення клієнт одержує електронний рахунок-фактуру, що враховує всі деталі туру та його загальну вартість.

ВАРІАНТ 22. Програмний модуль «Ресторан». Дана програмна розробка передбачена для працівників та відвідувачів ресторану. Зокрема, необхідно в електронному вигляді зберігати та редагувати меню із розбиттям страв на категорії із зазначенням назви страви, інгредієнтів, ваги та ціни. До того ж програма повинна містити інформацію про працівників ресторану, їх розклад роботи та столики, які вони обслуговують. Кожне замовлення певного столика офіціант оформляє окремо. Замовлення може доповнюватися, в кінці формується чек із переліком усіх страв та сумою до сплати. Слід автоматизувати підрахунок грошового виторгу за довільний період. Клієнт має змогу переглядати меню ресторану, а пройшовши реєстрацію, може виконувати попереднє бронювання столика та/або замовлення страв.

ВАРІАНТ 23. Програмний модуль «Магазин побутової техніки». Дана розробка повинна надавати можливість адміністратору магазину вносити та редагувати інформацію про працівників магазину, товари, які є на складі магазину, вести облік фірм-постачальників, договорів на постачання продукції, наявних та проданих товарів, постійних клієнтів та запроваджувати свою систему акцій та знижок. Працівникам магазину дана система дає змогу переглядати інформацію про продукцію, здійснювати пошук за різними критеріями, оформляти купівлю товару та гарантію на нього. Клієнт магазину має можливість перегляду та пошуку продукції, а також подавати заявку на придбання товару після проходження реєстрації. При купівлі покупцеві видається електронний чек.

ВАРІАНТ 24. Програмний модуль «Розважальний центр». Дана розробка передбачена для поширення інформації та автоматизації роботи центру розваг, який має у своєму складі кінотеатр, ковзанку, боулінг, більярд, ігрові автомати та інші розважальні заклади. Надану інформацію слід супроводжувати фотографіями закладів, зазначати встановлені тарифи та акції. Для придбання наявні різні типи абонементів. Вихідні дані

вводить та редагує адміністратор системи. Також він виконує облік постійних клієнтів та оплати послуг. Менеджери центру можуть здійснювати продаж квитків та абонементів, пошук і бронювання місць. Зареєстрованим користувачам доступні послуги резервування та придбання квитків з одержанням їх електронних варіантів.

ВАРІАНТ 25. Програмний модуль «Спортивний клуб». Даний програмний продукт призначений для надання інформації та автоматизації роботи спортивно-оздоровчого комплексу, який пропонує користувачам заняття з різних видів спорту. У програмі потрібно надавати інформацію про тренерів, спортивні і тренажерні зали, види послуг. У спортивному клубі виконується продаж різноманітних клубних абонементів. Тренерам потрібно забезпечити функції перегляду розкладу занять у своїх групах, та персональні дані відвідувачів груп. Для клієнтів потрібно передбачити як можливість перегляду загальної інформації, так і реєстрації, запису в групу та купівлю абонементів. Після оформлення заявки та оплати користувачеві видається електронний абонемент

ВАРІАНТ 26. Програмний модуль «Комп'ютерна фірма». Розробіть інформаційну систему для підтримки роботи фірми з продажу та обслуговування комп'ютерної техніки. Користувачами системи є адміністратор, працівники фірми та клієнти. Інформаційна система повинна здійснювати облік комп'ютерного обладнання різного типу, виробництва та потужності. Інформацію про товари слід супроводжувати коротким описом та фото і робити доступною для перегляду для простих клієнтів. Адміністратор системи відповідальний за введення інформації про товари та послуги, працівників та постачальників. Працівники фірми здійснюють підбір та пошук товарів за вимогами клієнтів, оформляють купівлю, приймають замовлення на ремонт та обслуговування. Забезпечте автоматизоване формування квитанції на оплату з можливістю збереження та/або друку. Клієнти повинні мати змогу робити замовлення товарів та послуг, пройшовши попередню реєстрацію, а також переглядати історію замовлень в особистому кабінеті.

ДОДАТОК Б.
Рекомендації щодо оформлення курсової роботи з дисципліни «Інженерія програмного забезпечення»

В оформленні курсової роботи основний текст подається шрифтом Times New Roman, 14, для нового рядка використовується відступ – 1,25, інтервал між рядками – 1,5, вирівнювання основного тексту – по ширині. Для документу встановити параметри полів: ліве – 3, праве – 1,5, верхнє та нижнє – 2 см.

Кожна програмна розробка повинна мати свій **шифр**, який унікально ідентифікує розроблений проект, його виконавців та документацію до нього. Шифр вказують на другій сторінці програмної документації – **Листі затвердження.**

При виконанні студентських проектів використовується така структура шифру:

482.362.ОКР-ЗК НВ-В,

де **482** – код України;
362 – код Чернівецького національного університету;
ОКР – освітньо-кваліфікаційний рівень, для спеціальності «Комп'ютерна інженерія» використовується номер **123**;
ЗК – дві останні цифри номера залікової книжки*;
НВ – номер варіанта завдання (**дві цифри**);
В – умовне позначення виду проекту: 1 – курсовий проект (робота) на 1 курсі, 2 – курсовий проект (робота) на 2 курсі, **3** – курсовий проект (робота) на 3 курсі , 4 – курсовий проект (робота) на 4 курсі, 5 – дипломний проект (магістерська робота).

*Для робіт, які виконуються спільно декількома виконавцями, у шифрі останні цифри номеру залікової зазначаються через слеш «/», наприклад:

482.362.ОКР-ЗК1/ЗК2 НВ-В,

482.362.123-07/18 08-3

ДОДАТОК В.

Титульна сторінка курсової роботи: ПРИКЛАД

Міністерство освіти і науки України
Чернівецький національний університет імені Юрія Федьковича

Навчально-науковий інститут фізико-технічних
та комп'ютерних наук
Відділ комп'ютерних технологій
Кафедра комп'ютерних систем та мереж

КУРСОВИЙ ПРОЕКТ

з __Інженерії програмного забезпечення___
(назва дисципліни)
на тему :__Програмний модуль «Агентство нерухомості»__

Студентів _ 3 _ курсу _342 групи
спеціальності 123_«Комп'ютерна інженерія»
_____Мендришора Л.В. _____
(прізвище та ініціали)
_____Цимбалюк Н.Ю._ _____
(прізвище та ініціали)
Керівник канд. техн. наук, доцент каф.КСМ, Величко Р. І. (викладач, що веде лабораторні роботи)_____
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Національна шкала _____
Кількість балів: _____ Оцінка: ECTS _____

м. Чернівці – 202_ рік

ДОДАТОК Г.

Лист затвердження курсової роботи: ПРИКЛАД

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Навчально-науковий інститут фізико-технічних та
комп'ютерних наук
Відділ комп'ютерних технологій
Кафедра комп'ютерних систем та мереж

ЛИСТ ЗАТВЕРДЖЕННЯ

482.362.123-07/18 08-3

УЗГОДЖЕНО
Керівники проекту
_____Величко Р.І.
"__" _____ 202_ р.

Виконали студенти 3-го курсу
_____ Л.В. Мендришора
_____Н.Ю. Цимбалюк
"__" _____ 202_ р.

202_

ДОДАТОК Д.

Титульна сторінка звіту з лабораторної роботи: ПРИКЛАД

Міністерство освіти і науки України
Чернівецький національний університет імені Юрія Федьковича

Навчально-науковий інститут фізико-технічних та
комп'ютерних наук
Відділ комп'ютерних технологій
Кафедра комп'ютерних систем та мереж

ЗВІТ
з лабораторної роботи № 1
на тему «Специфікація вимог до програмного продукту»
з дисципліни «Інженерія програмного забезпечення»

Варіант 8

Виконали: студент(и) гр. 342
_____ Л.В. Мендришора
_____ Н.Ю. Цимбалюк
"__" _____ 202_ р.

Перевірили:
_____ Р.І. Величко
_____
"__" _____ 202_ р.

202_

ДОДАТОК Е.

ПРИКЛАД СПЕЦИФІКАЦІЇ

Команда: DreamTeam

Програмний продукт: «Замовлення обідів до офісу»

Специфікація вимог до програмного продукту

СПЕЦИФИКАЦІЯ ВИМОГ ДО ПРОГРАМНОГО ПРОДУКТУ

Вступ

Цей документ складено групою розробників DreamTeam для опису програмного продукту «Lunch Manager», а також системних, функціональних і нефункціональних вимог до даної розробки.

1. Підстави для розробки

Завдання на розробку інформаційної системи одержано в рамках лабораторного практикуму з дисципліни «Інженерія програмного забезпечення».

Умовне позначення розробки: 482.362.123-07/18 28-3

Варіант 28: «Замовлення обідів до офісу»

Назва програми: «Lunch Manager»

Виконавці роботи: студенти 3 курсу, гр. 342 Мендришора Л.В, Цимбалюк Н.Ю.

2. Призначення та область застосування

2.1. Мета і призначення розробки

Інформаційна система «Lunch Manager» має на меті створення багатокористувацького онлайн-сервісу замовлень обідів до офісу та автоматизацію обліку й відстеження виконаних замовлень.

Система надає користувачам спектр послуг, пов’язаних із замовленням доставки обідів для організації без потреби відвідування закладу харчування. Необхідно забезпечити користувачам можливість доступу до неї зі стаціонарних та мобільних пристроїв, без необхідності встановлення додаткового програмного забезпечення на робочі комп’ютери.

2.2. Область застосування

Розроблений програмний продукт можна рекомендувати для впровадження закладам харчування для поширення переліку страв та способів обслуговування, які вони пропонують, популяризації серед своїх клієнтів звичок здорового харчування, підтримки бізнесу, зокрема в умовах карантину та перебоїв з електропостачанням, охоплення великої групи користувачів, а

також будь-яким компаніям зі сфери обслуговування для поширення інформації та автоматизації обліку товарів та послуг.

Даний продукт буде корисним для компаній та організацій, які піклуються про своїх працівників та надають можливість з користю провести час обідньої перерви завдяки попередньому замовлення та доставки страв для працівників.

Сфери застосування – обслуговування споживачів, корпорації з необхідністю працювати з офісу, заклади громадського харчування, служби доставки їжі, кейтерингові компанії.

2.3. Зацікавлені сторони

Сторонами, що зацікавлені в розробці є:

- Компанія StarTrack, яка замовлятиме обіди в офіс для працівників. Це головний замовник розробки, який бере на себе фінансові витрати, погодження плану виконання робіт та перевірку їх результатів.
- Мережа ресторанів домашніх страва, з якою у компанії StarTrack укладено договір на постачання обідів. Ця компанія також забезпечує доставку замовлень.
- Представником стейкхолдера від компанії, з яким уточнюватимуться вимоги до розробки та погоджуватимуться ухвалені рішення, буде викладач Величко Р.І.
- Користувачі програми: офіс-менеджер та співробітники фірми.

2.4. Перспективи впровадження продукту

Наразі обіди до офісу замовляються індивідуальним: окремими співробітниками або через офіс-менеджера. Фірму-постачальника кожен обирає самостійно. Якщо замовлення виконує офіс-менеджер, він/вона повинні визначити які співробітники бажають замовити обід, яким стравам вони віддають перевагу та чи мають якісь особливі вимоги до харчування. Далі замовлення передається до компанії-постачальника обідів у телефонному режимі. Окремо потрібно сплачувати за доставку. Оплату обіду кожен співробітник виконує самостійно, тому суму до сплати щоразу потрібно

уточнювати офіс-менеджеру з кожним індивідуально і відстежувати своєчасність оплати.

Впровадження розроблюваного програмного продукту забезпечить такі переваги:

- безпосередній зв'язок із компанією-постачальником обідів;
- централізоване замовлення страв із оповіщенням офіс-менеджера;
- компанія StarTrack оплачуватиме замовлення обідів своїм співробітникам;
- заощадження часу співробітників та офіс-менеджера, який потрібно було витратити на пошук закладу харчування та замовлення страв у ручному режимі;
- централізована оплата замовлень через системи компанії-постачальника;
- заощадження коштів співробітників: за рахунок збільшеного обсягу замовлень компанія-постачальник надає знижку на вартість страв;
- скорочення витрат на доставку замовлень;
- збільшення концентрації на роботі;
- підвищена довіра співробітників до власної компанії.

2.5. Огляд ринку

Серед найбільш відомих сервісів замовлення їжі можна виокремити:

Glovo (https://glovoapp.com/ua) – міжнародний онлайн-сервіс із доставки їжі та продуктів, доступний у веб- та мобільній версії. Співпраця з багатьма закладами харчування. Проте краще підійде для індивідуальних замовлень, аніж для колективних.

Famiglia Grande (https://famigliagrande.ua/) – компанія надає безкоштовну доставку при замовленні від 500 грн, пропонує акції та знижки, проте в асортименті продукції, яку доставляють, відсутні перші страви та гарніри, здебільшого піца та суші, на сайті зазначено досить тривалий час доставки (від 40 до 60 хв), відсутня інформація про групові замовлення.

Mister Am (https://misteram.com.ua) – замовлення від різних закладів харчування по містах країни, різноманітний перелік страва, можливість онлайн-оплати, виконання групових замовлень, безкоштовної доставки. Умови замовлення потрібно уточнювати з кожним закладом окремо. Групові замовлення необхідно робити завчасно (наприклад, за день).

3. Вимоги до програмного продукту

3.1. Функціональні вимоги

Програмний продукт повинен забезпечувати роботу двом категоріям користувачів: офіс-менеджер (далі менеджер) та співробітники.

Загальні функціональні можливості для усіх користувачів:

1.1. Ідентифікація користувачів відповідних категорій повинна здійснюватися за допомогою логіна та паролю, які вводяться на початковій сторінці застосунку.

1.2. При коректному введенні аутентифікаційних даних відбувається перехід на робочу сторінку співробітника або менеджера відповідно.

1.2.1. При некоректному вводі з’являється повідомлення про помилку і прохання перевірити правильність введених логіна та паролю. Кількість спроб для вводу обмежена до трьох разів.

Функціональні можливості співробітника:

2.0. Співробітник повинен мати змогу сформувати необхідний для нього перелік страв на обід.

2.1. Співробітник повинен мати змогу ознайомитися з усім переліком страв з меню на поточний день.

2.1.1. Співробітник повинен мати змогу отримати інформацію про інгредієнти, вагу та калорійність страв з меню.

2.1.2. Співробітник повинен мати змогу переглядати вартість страв.

2.2. Співробітник повинен мати змогу сформувати свій перелік страв.

2.2.1. Співробітник повинен мати змогу додавати страву до свого замовлення.

2.2.2. Співробітник повинен мати змогу видаляти страву зі сформованого списку замовлення.

2.2.3. Співробітник повинен мати змогу надсилати сформований список замовлення менеджерові не раніше ніж за добу та не пізніше ніж за 2 години до обідньої перерви.

2.2.4. Співробітник повинен одержувати підтвердження свого замовлення та інформацію про нього у електронному кабінеті застосунку.

Функціональні можливості менеджера:

3.1. Менеджер повинен мати змогу реєструвати користувача-співробітника.

3.2. Менеджер повинен мати змогу оновлювати дані про меню на поточний день з сайту закладу харчування.

3.2.1. Менеджер повинен мати змогу обирати страви з наперед визначеного переліку для групового замовлення.

3.2.2 Менеджер повинен мати змогу визначати кількість екземплярів страв.

3.2.3 Менеджер повинен мати змогу додавати нові замовлення до раніше сформованого переліку.

3.2.4 Менеджер повинен мати змогу видаляти страву із списку замовлень.

3.2.5. Менеджер повинен мати змогу загальний звіт про замовлення страв на поточну дату, а також замовлення для кожного користувача, із зазначенням вартості, кількості та назв замовлених страв.

3.2.6. Менеджер повинен отримати чек до сплати за поточне замовлення.

3.2.7. Менеджер повинен мати змогу оплатити поточне замовлення через платіжну систему закладу харчування.

3.2.8. Менеджер повинен мати змогу переглядати чек та квитанцію про сплату у своєму електронному кабінеті.

3.3. Організація вхідних і вихідних даних

Вхідні дані: дані подаються користувачам в електронному вигляді у веб- або мобільній версії застосунку. Дані про стави одержуються з меню компанії громадського харчування. Кожен пункт меню активний для перегляду та додавання до замовлення. Реєстрацію користувачів виконує офіс-менеджер. Дані про замовлення зберігаються централізовано у базі даних. Реєстраційні та облікові дані користувачів повинні бути захищені.

Вихідні дані: чек до сплати, підтвердження про оформлення замовлення, звіт по замовлених стравах (кількість, користувачі, сума в грошах) за звітний період, квитанцію про сплату, профіль уподобань конкретного користувача, звіт із витрат на замовлення обідів для компанії за визначений період у форматі csv.

3.4. Вимоги до складу і параметрів технічних засобів

Система являє собою веб-застосунок, створений мовою Java з використанням технології JSP. Як СУБД обрано систему MySQL. Оскільки продукт є кросплатформеним рішенням, його використання не прив'язане до конкретної операційної системи. Прикладна програма повинна працювати по мережі. Необхідні налаштування віртуальної машини Java та контейнер сервлетів Apache Tomcat 6.

Для нормального функціонування програми "Lunch Manager" потрібно дотримуватися таких вимог.

Вимоги до розробки ПЗ:

- мова програмування Java
- середовища: Idea, NetBeans, Eclipse і сервер Apache Tomcat.
- потребує встановлення СУБД MySQL;
- під'єднання до локальної мережі та Інтернет-сервісів.

Мінімальні апаратні вимоги для належної роботи системи:

- Оперативна пам'ять: 4 ГБ
- Процесор: 1.9 ГГц
- Вільне місце на диску: 200 МБ
- Інтернет-з'єднання 100 Мб/с

Вимоги до програмної сумісності:

- Програма доступна для використання в ОС Windows 7/8/10/11 та Linux.
- Може працювати у веб-браузерах Google Chrome, Firefox, Safari, Opera, Brave, Microsoft Internet Explorer, Edge.
- Зв'язок з API для здійснення оплати.
- API для підтримки аутентифікації користувачів JWT-токенів.
- Захист персональних та облікових даних користувачів.

3.5. Експлуатаційні вимоги

Для розробленого програмного продукту визначено такі вимоги щодо експлуатації:

- Простота експлуатації**:**

Час навчання персоналу: 2 год.

3. Стійкість до відмов:

Час відновлення системи після відмови: 1 год.

Відсоток подій, які призводять до відмов: 1.

Ймовірність зміни даних при відмові: 0,1.

- Вимоги до надійності:

5. Середня тривалість часу між двома послідовно проявами помилок в системі повинно складати щонайменше п'ять годин.
6. Ймовірність виходу системи з ладу повинна складати 2–3 %.
7. Коефіцієнт готовності системи – 98 зі 100.
8. Кількість помилок на функцію (в середньому) – 0,1 багів.
9. Доступність системи: 25000 годин безперервної роботи.

- Вимоги до працездатності:

Кількість виконуваних транзакцій за секунду (в середньому): 5 тис.

- Зручність використання

Система повинна бути простою в експлуатації як для досвідченого оператора так і звичайного користувача комп'ютера.

Після двох годин вивчення роботи програми користувач системи повинен ознайомитися з усіма функціями системи, а також правилами користування та обмеженнями. Середня кількість помилок оператора не повинна перевищувати двох на день. Користувач повинен мати доступ до контекстної довідки, інструкції щодо користування програмою та системи зворотного зв'язку.

4. Очікувані техніко-економічні показники

Умовна кількість команд програмного коду Q визначається за формулою:

$$Q = q * C, \quad (1)$$

де q – коефіцієнт, який позначає умовну кількість команд у залежності від типу задачі. Значення цього коефіцієнта визначається з табл. 4 та 4.1. Коефіцієнт С визначається з табл. 4.2.

Таблиця 4

Значення коефіцієнта **q** для визначення кількості команд у коді програми

Код виду	Назва виду ПЗ	Межі зміни коефіцієнта
1.0	ПЗ загального призначення	до 1500
2.0	ПЗ технології автоматизації програмування та проектування АСУ	від 4500 до 5500
3.0	ПЗ методо-орієнтованих розрахунків	від 2500 до 3500
4.0	ПЗ організації обчислювального процесу	від 3500 до 4500
5.0	ПЗ функціонального призначення	від 1500 до 2500

Таблиця 4.1

Класифікація видів програмного забезпечення (доповн. до табл.4)

Код виду	Назва виду ПЗ	Коефіцієнт **q** відповідно до складу та вмісту розробленого ПЗ
1	2	3
1.0	ПЗ загального призначення	1.9. ПЗ СУБД; 1.10. ПЗ систем ведення лінійних файлів; 1.11. ПЗ ведення БД і лінійних файлів;

Код виду	Назва виду ПЗ	Коефіцієнт **q** відповідно до складу та вмісту розробленого ПЗ
1	2	3
		1.12. ПЗ інформаційно-пошукових та інформаційно-довідкових систем; 1.13. ПЗ введення інформації; 1.14. ПЗ моніторів телеобробки і мереж ПЕОМ; 1.15. ПЗ оточення СУБД, які розширюють можливості існуючих СУБД; 1.16. ПЗ, які розширюють можливості обробки.
2.0	ПЗ технології автоматизації програмування та проектування АСУ	2.7. ПЗ автоматизації проектування для автоматизації проектування різних АСУ; 2.8. ПЗ технології програмування; 2.9. ПЗ автоматизації програмування (для автоматизації процесів обробки та виводу інформації); 2.10. ПЗ, які розширюють існуючі мови програмування для підвищення їх компактності та простоти користування; 2.11. ПЗ загального призначення, функціонально-орієнтовані; 2.12. ПЗ автоматичного програмування.

Код виду	Назва виду ПЗ	Коефіцієнт **q** відповідно до складу та вмісту розробленого ПЗ
1	2	3
3.0	ПЗ методо-орієнтованих розрахунків	3.6. ПЗ оптимізаційних розрахунків (забезпечують рішення різного класу задач оптимального планування та управління виробництвом); 3.7. ПЗ статистичного аналізу та прогнозування (для прогнозування техніко-економічних показників, попиту, тощо); 3.8. ПЗ мережевого планування та управління; 3.9. ПЗ загальної математики; 3.10. ПЗ імітаційного моделювання.
4.0	ПЗ організації обчислювального процесу	4.5. Автоматизація процесу введення наборів даних, при забезпечені їх надійного та систематизованого збереження; 4.6. Підвищення продуктивності ПЕОМ і користувачів ПЗ; 4.7. Формування і видача звітів про роботу ПЕОМ; 4.8. Оперативний контроль системи та ресурсів.
5.0	ПЗ функціонального призначення	Для автоматизації обробки економічних даних виділяється ПЗ функціонального призначення. 5.6. ПЗ оперативного управління основним виробництвом; 5.7. ПЗ управління технічною підготовкою виробництва; 5.8. ПЗ бухгалтерського обліку та управління фінансами; 5.9. ПЗ управління кадрами; 5.10. ПЗ, які не ввійшли ні в один із перерахованих вище видів ПЗ.

Коефіцієнт **С** враховує новизну та складність програмного рішення.

Програмні продукти за рівнем новизни можна віднести до однієї з 4-х груп:

- Група А – розробка принципово нових завдань.
- Група Б - розробка оригінальних програм.
- Група В - розробка програм з використанням типових рішень.
- Група Г - разова типова задача.

За ступенем складності програмні продукти можуть бути віднесені до однієї з 3-х груп:

1–алгоритми оптимізації та моделювання системи

2 – завдання обліку, звітності та статистики;

3 – стандартні алгоритми.

Коефіцієнт **С** визначається з табл. 4.2 на перетині показників складності та новизни.

Таблиця. 4.2

Коефіцієнт С, що визначає новизну та складність програми

Мова програмування	**Група складності**	**Рівень новизни**			
		А	**Б**	**В**	**Г**
Високого рівня	1	1,38	1,26	1,15	0,69
	2	1,3	1,19	1,08	0,65
	3	1,2	1,1	1,00	0,60
Низького рівня	1	1,58	1,45	1,32	0,79
	2	1,49	1,37	1,24	0,74
	3	1,38	1,26	1,15	0,69

Тепер, виходячи з формули (1) можна визначити умовне число команд Q.

Трудомісткість розробки програмного продукту (t) визначається за формулою (2), люд.-міс.

$$t=3,6*(\eta_{т.р.к})^{1,2}, \qquad (2)$$

де $\eta_{т.р.к}$ - число **тисяч** команд програмного коду.

Загальна тривалість розробки ПП (Т) обчислюється за формулою (3), міс.:

$$T=2,5*t^{0,32}, \quad (3)$$

Середня кількість виконавців ($PL_{вик}$) розраховується виходячи з трудомісткості та тривалості розробки ПП за формулою (4), люд.:

$$PL_{вик}=t/T, \quad (4)$$

Продуктивність праці групи розробників ПП ($П_р$), вихідних команд/люд.-міс, визначається за формулою (5):

$$П_р=1000*\eta_{т.р.к}/t \quad (5)$$

5. Порядок контролю і приймання

Для створення якісного програмного продукту, який би відповідав очікуванням замовника, доцільно проводити парні консультації та ревізії вимог до програми між представниками різних команд.

Уточнення вимог потрібно здійснювати на кожному етапі розробки. Будь-які зміни ретельно фіксувати у програмній документації.

Демонструвати проміжні рішення у вигляді діаграм, схем, макетів та прототипів викладачеві, як представнику стейкхолдерів.

Перевірка розробленого програмного рішення виконується стороною замовника відповідно до специфікації вимог до програмного продукту.

Об’єкти та методи тестування правильності виконання програмою заявлених функцій подаються у розділі 4 курсової роботи – «Програма і методика випробувань».

Перевірка роботи користувача з системою здійснюється за попередньо розробленими сценаріями виконання користувачами функцій, заявлених у специфікації.

ДОДАТОК Є.
Діаграма Ганта в Excel

Що таке діаграма Ганта?

Діаграма Ганта названа в честь Генрі Ганта, американського інженера і консультанта з менеджменту, який придумав таку діаграму в 1910 році. Діаграма Ганта в Excel представляє проекти або завдання у вигляді каскаду горизонтальних лінійчатих графіків. Діаграма Ганта показує розкладену на частини структуру проекту (дату початку і закінчення, різні зв'язки між завданнями в рамках проекту) і таким чином допомагає контролювати виконання завдань у часі і відповідно до намічених орієнтирів.

Як створити діаграму Ганта в Excel 2010 і 2013

На жаль, Microsoft Excel не пропонує вбудованого шаблону діаграми Ганта. Однак, можна швидко створити її самостійно, використовуючи функціонал лінійчатої діаграми і трохи форматування.

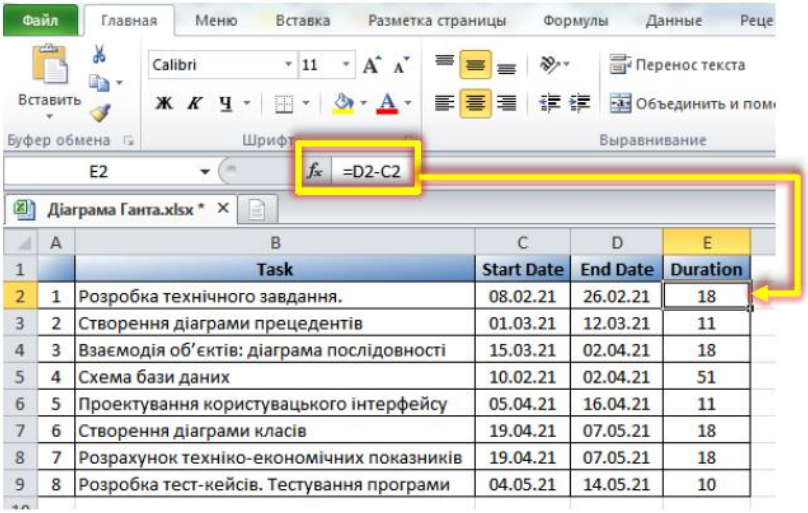
Уважно виконайте наступні кроки. У нашому прикладі ми створюємо діаграму Ганта в Excel 2010, але те ж саме можна зробити в інших версіях Excel.

Крок 1. Створіть таблицю проекту

Насамперед, введемо дані проекту на лист Excel. Запишіть кожну задачу в окремому рядку і побудуйте структурний план проекту, вказавши **дату початку** (Start date), **завершення** (End

date) і **тривалість** (Duration), тобто кількість днів, яке потрібно для завершення завдання.

Порада: Для створення діаграми Ганта необхідними є тільки стовпці Start date і Duration. Однак, якщо створити також стовпець End date, то обчислити тривалість завдання можна за допомогою простої формули, як видно на малюнку нижче:

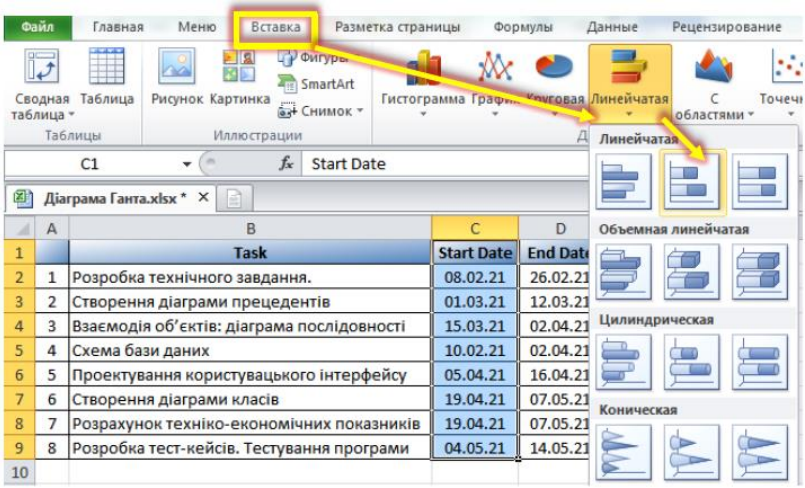


	A	B	C	D	E
1		Task	Start Date	End Date	Duration
2	1	Розробка технічного завдання.	08.02.21	26.02.21	18
3	2	Створення діаграми прецедентів	01.03.21	12.03.21	11
4	3	Взаємодія об'єктів: діаграма послідовності	15.03.21	02.04.21	18
5	4	Схема бази даних	10.02.21	02.04.21	51
6	5	Проектування користувацького інтерфейсу	05.04.21	16.04.21	11
7	6	Створення діаграми класів	19.04.21	07.05.21	18
8	7	Розрахунок техніко-економічних показників	19.04.21	07.05.21	18
9	8	Розробка тест-кейсів. Тестування програми	04.05.21	14.05.21	10

Крок 2. Побудуйте звичайну лінійчату діаграму Excel на базі даних стовпця "Start date"

Почніть побудову діаграми Ганта в Excel зі створення звичайної лінійчатої діаграми з накопиченням:

- Виділіть діапазон **Start Date** разом з заголовком стовпця, в нашому прикладі це C1: C9. Потрібно виділити тільки комірки з даними, а не весь стовпець аркуша.
- На вкладці **Вставка** (Insert) у розділі **Діаграми** (Charts) натисніть **Вставити лінійчату діаграму** (Bar).
- У меню, в групі **Лінійчата** (2-D Bar) натисніть **Лінійчата з накопиченням** (Stacked Bar).

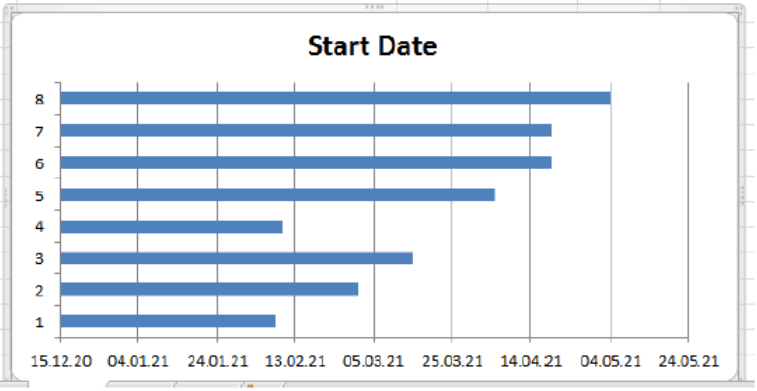


	A	B	C	D
1		Task	Start Date	End Date
2	1	Розробка технічного завдання.	08.02.21	26.02.21
3	2	Створення діаграми прецедентів	01.03.21	12.03.21
4	3	Взаємодія об'єктів: діаграма послідовності	15.03.21	02.04.21
5	4	Схема бази даних	10.02.21	02.04.21
6	5	Проектування користувацького інтерфейсу	05.04.21	16.04.21
7	6	Створення діаграми класів	19.04.21	07.05.21
8	7	Розрахунок техніко-економічних показників	19.04.21	07.05.21
9	8	Розробка тест-кейсів. Тестування програми	04.05.21	14.05.21
10				

В результаті на аркуші повинна з'явиться ось така діаграма:

Діаграма Ганта.xlsx *

	A	B	C	D	E
1		Task	Start Date	End Date	Duration
2	1	Розробка технічного завдання.	08.02.21	26.02.21	18
3	2	Створення діаграми прецедентів	01.03.21	12.03.21	11
4	3	Взаємодія об'єктів: діаграма послідовності	15.03.21	02.04.21	18
5	4	Схема бази даних	10.02.21	02.04.21	51
6	5	Проектування користувацького інтерфейсу	05.04.21	16.04.21	11
7	6	Створення діаграми класів	19.04.21	07.05.21	18
8	7	Розрахунок техніко-економічних показників	19.04.21	07.05.21	18
9	8	Розробка тест-кейсів. Тестування програми	04.05.21	14.05.21	10

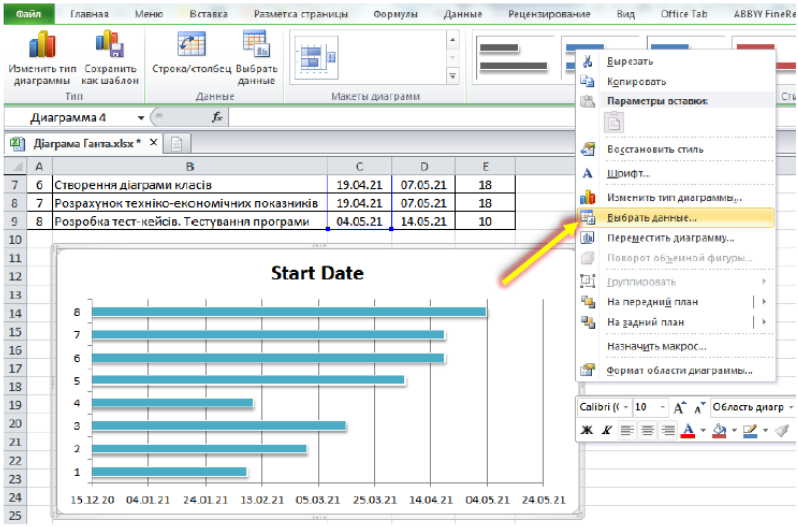


Зауваження: В деяких інших інструкціях по створенню діаграм Ганта пропонується спочатку створити порожню лінійчату діаграму, а потім наповнити її даними, як ми це зробимо на наступному кроці. Показаний метод краще, оскільки Microsoft Excel автоматично додасть один ряд даних і таким чином ми зекономимо трохи часу.

Крок 3. Додайте до діаграми дані про тривалість

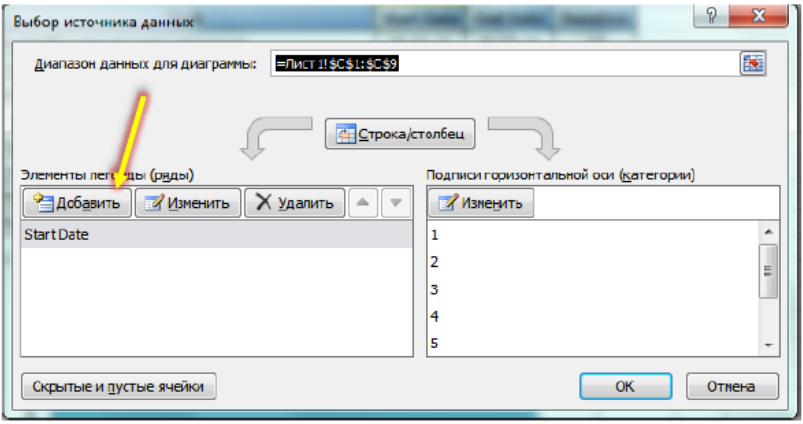
Далі до нашої майбутньої діаграмі Ганта потрібно додати ще один ряд даних.

1) Клацніть правою кнопкою миші в будь-якому місці діаграми і в контекстному меню натисніть **Вибрати дані** (Select Data).



Відкриється діалогове вікно **Вибір джерела даних** (Select Data Source). Як видно на малюнку нижче, дані стовпця **Start Date** вже додані в поле **Елементи легенди (ряди)** (Legend Entries (Series). Тепер сюди ж потрібно додати дані стовпця **Duration**.

2) Натисніть кнопку **Додати** (Add), щоб вибрати додаткові дані (Duration), які потрібно відобразити на діаграмі Ганта.

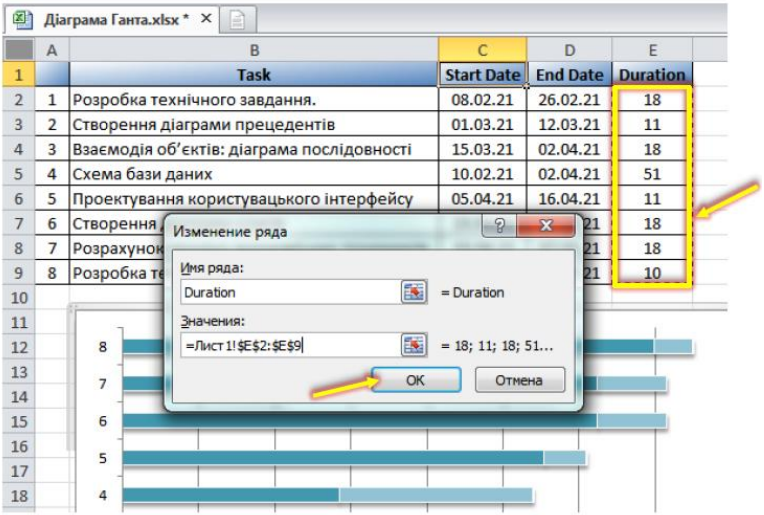


3) У вікні **Зміна ряду** (Edit series) зробіть ось що:

- У полі **Ім'я ряду** (Series name) введіть "Duration" або будь-яке інше ім'я за бажанням. Або можна поставити курсор в це поле і потім клацнути на заголовку відповідного стовпця в таблиці.
- Натисніть іконку вибору діапазону поруч з полем **Значення** (Series values).

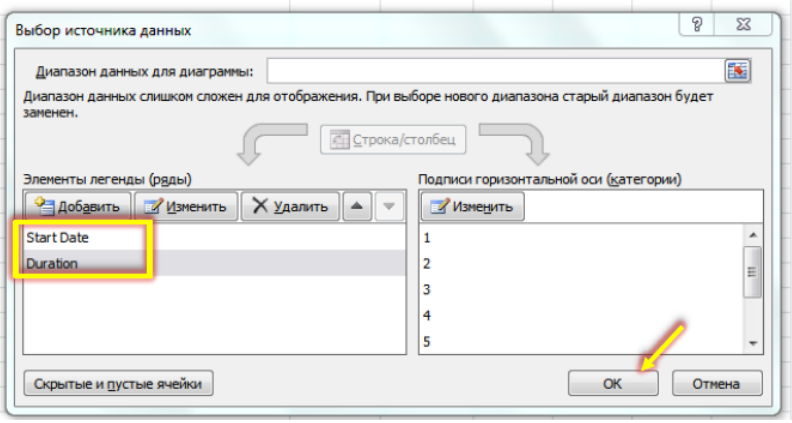
4) Діалогове вікно **Зміна ряду** (Edit series) зменшиться. Виділіть дані в стовпці Duration, клікнувши по першій клітинці (в нашому випадку це **E2**) і простягнувши мишею вниз до останньої

клітинки з даними (**E9**). Перевірте, що **не виділили** випадково заголовок або якусь вільну позицію.

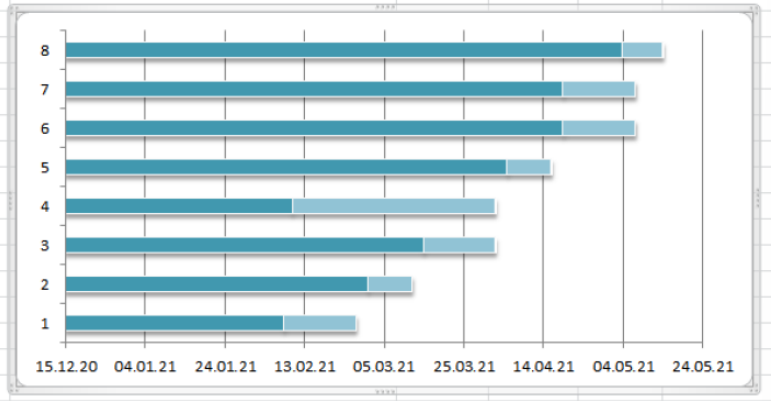


5) Натисніть ще раз іконку вибору діапазону. Діалогове вікно **Зміна ряду** (Edit series) буде знову розгорнуто і з'являться поля **Ім'я ряду** (Series name) і **Значення** (Series values). Натисніть ОК.

6) Ми знову повернемося до вікна **Вибір джерела даних** (Select Data Source). Тепер в полі **Елементи легенди (ряди)** (Legend Entries (Series) ми бачимо ряд Start Date і ряд Duration. Просто натисніть **ОК**, і дані будуть додані до діаграми.



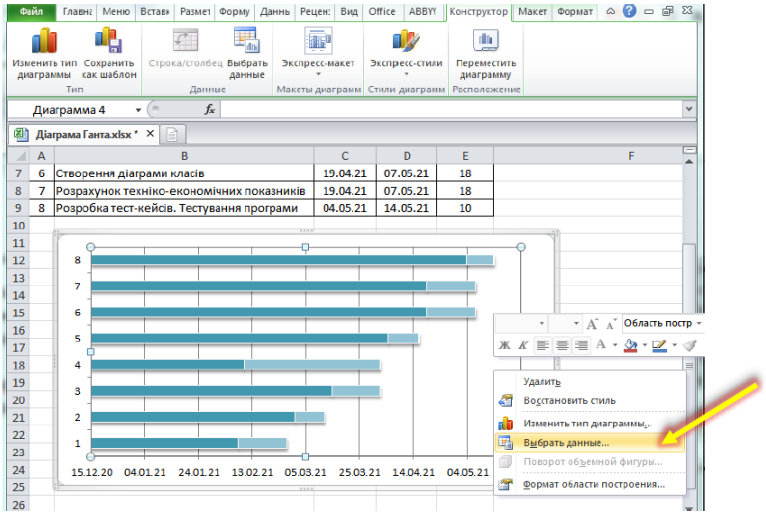
Діаграма повинна прийняти **приблизно** ось такий вигляд:



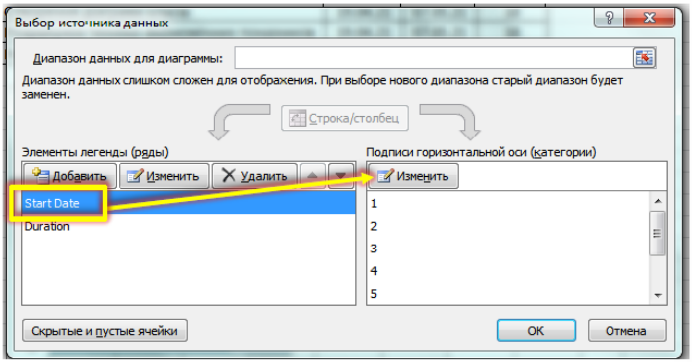
Крок 4. Додайте опис завдань до діаграми Ганта

Тепер потрібно в лівій частині діаграми замість чисел показати список завдань.

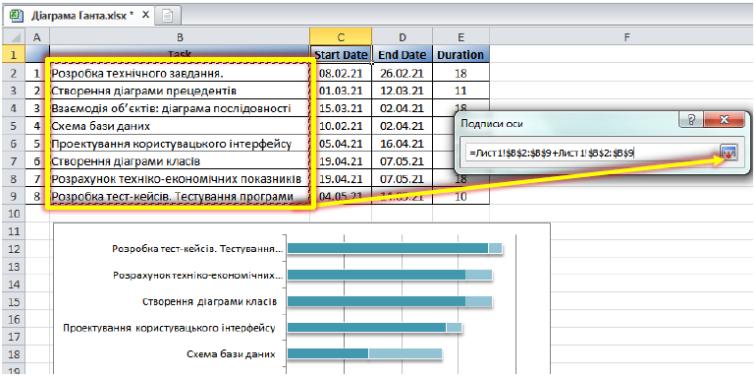
1) Клацніть правою кнопкою миші в будь-якому місці області побудови діаграми (область з синіми і блакитними смугами) і в меню натисніть **Вибрати дані** (Select Data), щоб знову з'явилося діалогове вікно **Вибір джерела даних** (Select Data Source).



2) У лівій області діалогового вікна виділіть **Start Date** і натисніть кнопку **Змінити** (Edit) у правій області вікна під назвою **Підписи горизонтальній осі (категорії)** (Horizontal (Category) Axis Labels).



3) Відкриється маленьке діалогове вікно **Підписи осі** (Axis Labels). Тепер потрібно виділити завдання так само, як на попередньому кроці вибирали дані про тривалість завдань (стовпець Durations) - натискаємо іконку вибору діапазону, потім клацаємо на першій задачі в таблиці і простягаємо виділення мишею вниз до останнього завдання. Пам'ятайте, що заголовок стовпчика не повинен виявитися виділеним. Зробивши це, ще раз натисніть по іконці вибору діапазону, щоб з'явилося діалогове вікно.

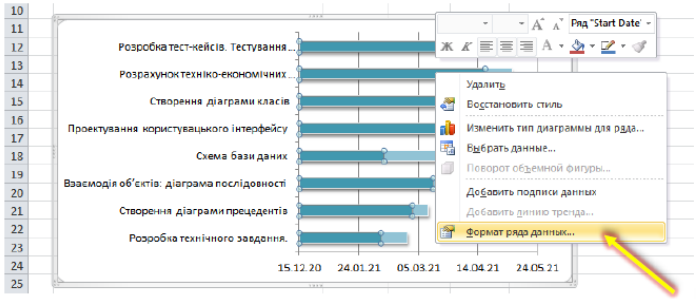


4) Двічі натисніть **ОК**, щоб закрити всі діалогові вікна.

Крок 5. Перетворюємо лінійчату діаграму в діаграму Ганта

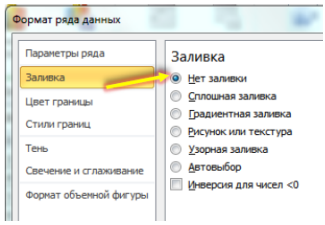
На даному етапі наша діаграма все ще є лінійчатої діаграмою з накопиченням. Щоб вона набула вигляду діаграми Ганта, потрібно правильно її оформити. Наше завдання - видалити сині лінії, щоб видимими залишилися тільки блакитні частини графіків, які відповідають завданням проекту. Технічно, ми не будемо видаляти сині лінії, а просто зробимо їх прозорими, а значить - невидимими.

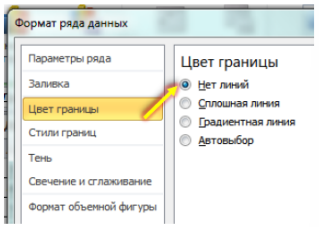
1) Клацніть на будь-якій синій лінії на діаграмі Ганта, при цьому всі вони будуть виділені. Натисніть на виділеній частині правою кнопкою миші і в контекстному меню оберіть **Формат ряду даних** (Format Data Series).



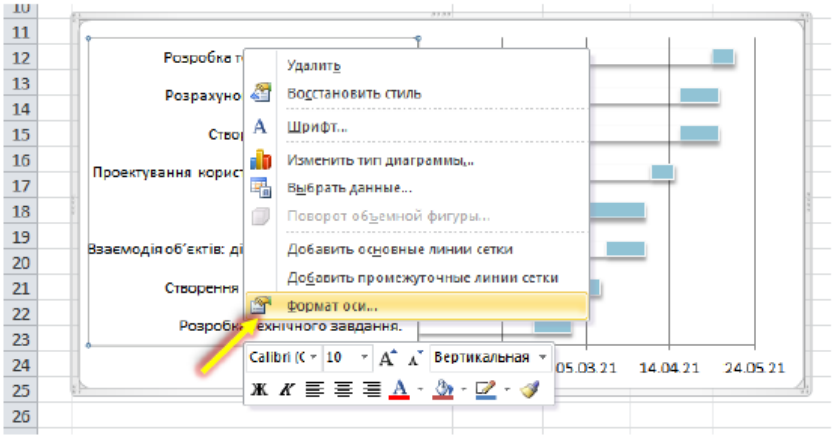
2) У діалоговому вікні зробіть наступне:

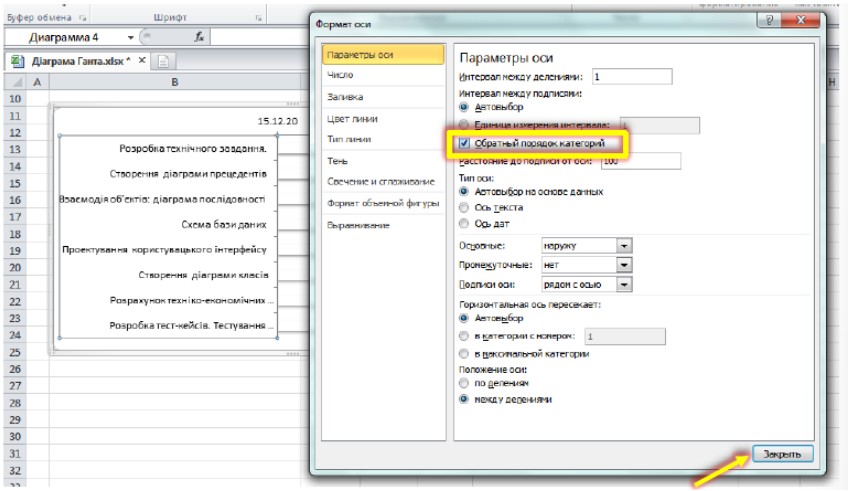
- У розділі **Заливка** (Fill) виберіть **Немає заливки** (No Fill).
- У розділі **Границя** (Border Color) виберіть **Немає ліній** (No Line).





3) Завдання на діаграмі Ганта, яку ми побудували в Excel, розташовані в зворотному порядку. Зараз ми це виправимо. Клацніть на списку завдань в лівій частині діаграми Ганта, щоб виділити вісь категорій. Відкриється діалогове вікно **Формат осі** (Format Axis). У розділі **Параметри осі** (Axis Options) позначте галочкою опцію **Зворотний порядок категорій** (Categories in reverse order), потім закрийте вікно, щоб зберегти зроблені зміни.





В результаті щойно зроблених змін:

- Завдання на діаграмі Ганта розташовані в правильному порядку.
- Дати на горизонтальній осі перемістилися з нижньої у верхню частину діаграми.

Діаграма стає схожою на звичайну діаграму Ганта. Наприклад, наша діаграма Ганта тепер виглядає ось так:

Крок 6. Налаштовуємо дизайн діаграми Ганта в Excel

Діаграма Ганта вже набуває потрібної форми, але можна додати ще кілька штрихів, щоб зробити її дійсно стильною.

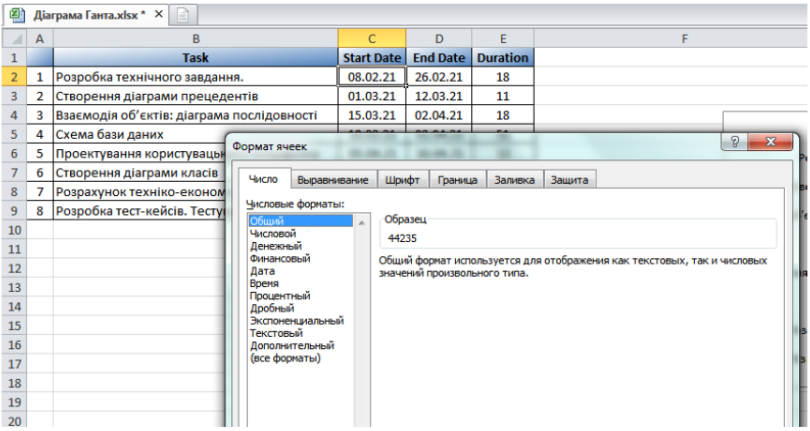
1. Прибираємо порожній простір в лівій частині діаграми Ганта

При побудові діаграми Ганта на початку графіка ми вставляли сині смуги, що показують початкову дату. Тепер можна прибрати і перемістити смуги завдань вліво, ближче до вертикальної осі.

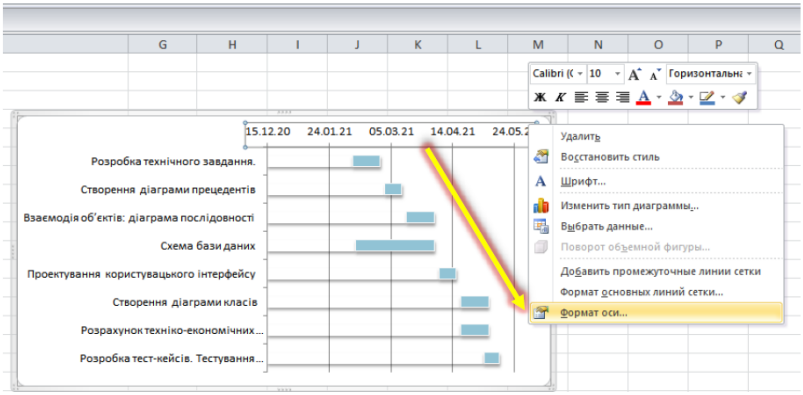
- Клацніть правою кнопкою миші на першому значенні стовпчика **Start Date** в таблиці з вихідними даними, в контекстному меню виберіть **Формат комірок**> **Число**> **Загальний** (Format Cells> Number> General). Запам'ятайте число, яке побачите в полі Зразок (Sample) - це числове значення цієї дати. У нашому випадку це число 44235. Як відомо, Excel зберігає дати у вигляді

чисел, рівних кількості днів від 1 січня 1900 до цієї дати (де 1 січня 1900 року = 1). Жодних змін

тут робити не потрібно, просто натисніть **Cancel**.



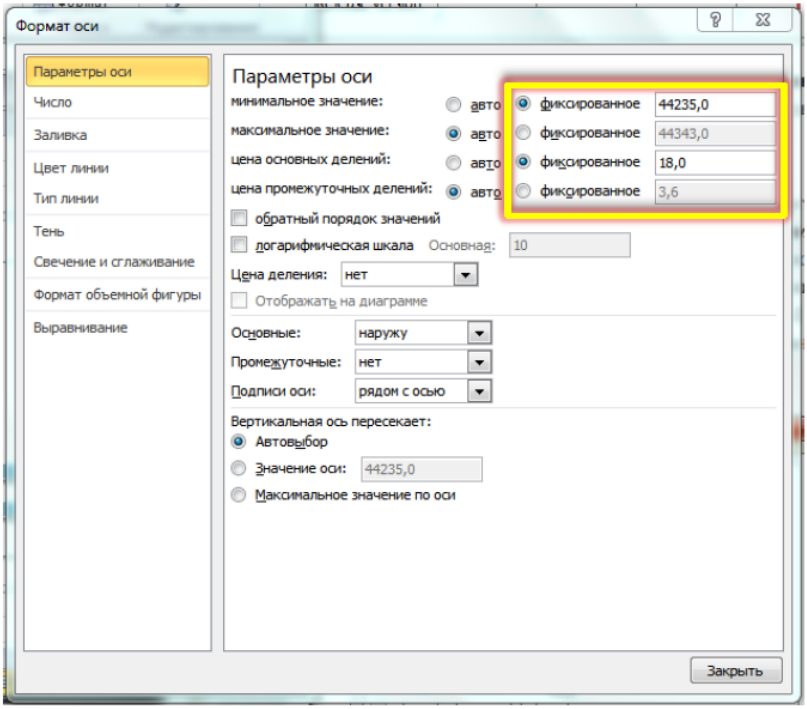
- На діаграмі Ганта клікніть на будь-яку дату над графіком. Одне клацання мишею виділить всі дати, після цього клікніть по ним правою кнопкою миші і в контекстному меню оберіть **Формат осі** (Format Axis).



- У меню **Параметри осі** (Axis Options) змініть опцію **Мінімум** (Minimum) на **Число** (Fixed) і введіть число, яке запам'ятали на попередньому кроці.

2. Налаштовуємо кількість дат на осі діаграми Ганта

Тут же, в діалоговому вікні **Формат осі** (Format Axis), яке відкрили на попередньому кроці, змініть параметри **Основні ділення** (Major unit) і **Проміжні ділення** (Minor unit) на **Число** (Fixed) і введіть потрібні значення інтервалів на осі. Зазвичай, чим коротші тимчасові рамки завдань в проекті, тим менший крок поділів потрібен на осі часу. Наприклад, якщо потрібно показати кожну другу дату, то введіть 2 для параметра **Основні ділення** (Major unit).

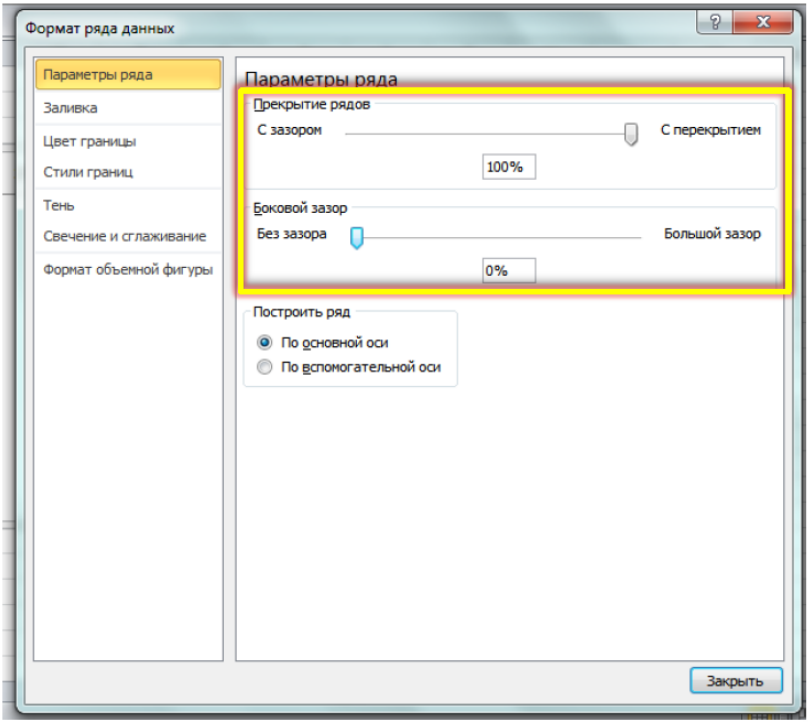


Порада: Пограйте настройками параметрів, поки не отримаєте потрібний результат. Не бійтеся зробити щось неправильно, завжди можна повернутися до налаштувань за замовчуванням, встановивши для параметрів значення **Автоматично** (Auto) в Excel 2010 і 2007 або натиснувши **Скидання** (Reset) в Excel 2013.

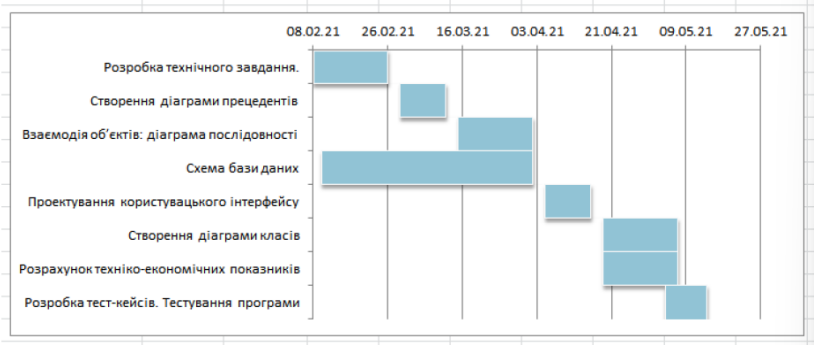
3. Видаляємо зайве порожнє місце між смугами

Розмістіть смуги завдань на графіку більш компактно, і діаграма Ганта стане виглядати ще краще.

- Виділіть блакитні смуги графіків, клацнувши на одній з них лівою кнопкою миші, потім клацніть на ній правою кнопкою миші і в меню оберіть **Формат ряду даних** (Format Data Series).
- У діалоговому вікні **Формат ряду даних** (Format Data Series) встановіть для параметра **Перекриття рядів** (Series Overlap) значення 100% (повзунок посуньте до упору вправо), а для параметра **Бічний зазор** (Gap Width) значення 0% або майже 0% (повзунок до упору або майже до упору вліво).



І ось результат наших зусиль - проста, але цілком акуратна діаграма Ганта в Excel:



Пам'ятайте, що створена таким способом діаграма Excel дуже близька до справжньої діаграми Ганта, і при цьому зберігає всі зручності діаграм Excel:

- Діаграма Ганта в Excel буде змінювати розмір при додаванні або видаленні завдань.
- Змініть початкову дату завдання (Start date) або її тривалість (Duration), і графік відразу ж автоматично відобразить зроблені зміни.
- Створену в Excel діаграму Ганта можна зберегти як картинку або перетворити в формат HTML і опублікувати в інтернеті.

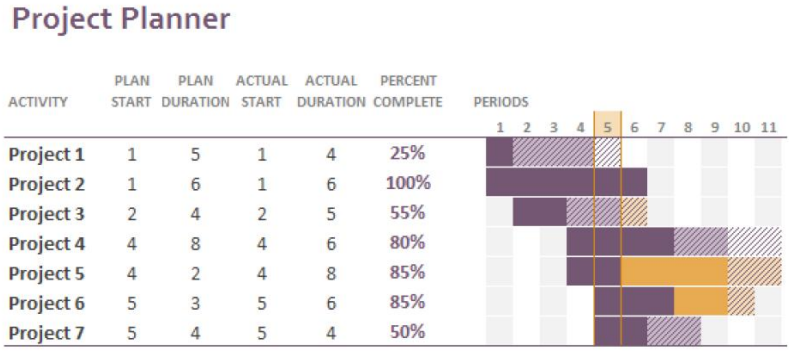
ПОРАДИ:

Налаштуйте оформлення діаграми Ганта, змінюючи параметри заливки, границі, тіні і навіть використовуючи 3D ефекти. Всі ці параметри доступні в діалоговому вікні **Формат ряду даних** (Format Data Series). Правою кнопкою миші по смузі графіка в області побудови діаграми і в контекстному меню натисніть **Формат ряду даних** (Format Data Series).

Шаблон діаграми Ганта для Excel 2013 від Microsoft

Цей шаблон діаграми Ганта для Excel називається **Планувальник проекту** (Gantt Project Planner). Він призначений для відстеження виконання проекту за різними показниками, таким як Plan Start і Actual Start, Plan Duration і Actual Duration, а також Відсоток завершення (Percent Complete).

В Excel 2013 цей шаблон доступний на вкладці **Файл** (File) у вікні **Створити** (New). Якщо в цьому розділі шаблон відсутній, то його можна завантажити з веб-сайту Microsoft. Для використання цього шаблону не потрібно ніяких додаткових знань - клікніть по ньому і приступайте до роботи.



ДОДАТОК Ж.

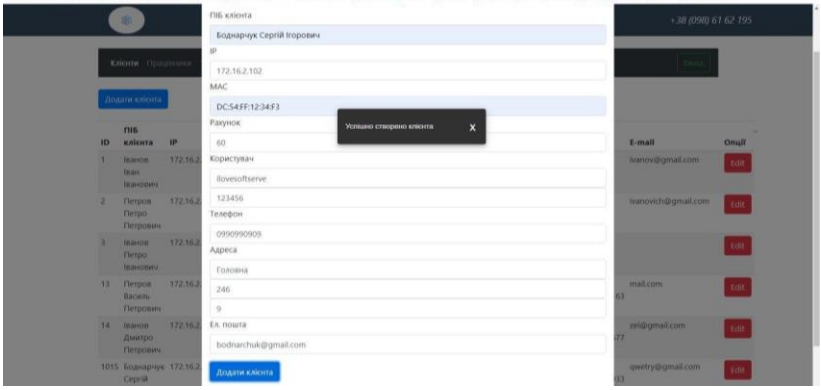
Приклади тест-кейсів

Тест-кейс №1

Назва: Додавання нового клієнта
Функціональна вимога, що перевіряється: Зберігання інформації про користувачів провайдера;
Пріоритет: 3/3
Автор: Волошин Андрій
Загальний опис тест-кейса:

1. Відкрити будь-який сучасний браузер
2. Перейти за адресою www.korononet.ua/admin
3. Ввести у поле «Логін»: «admin»
4. Ввести у поле «Пароль»: «admin»
5. Натиснути кнопку «УВІЙТИ»
6. Натиснути кнопку «Додати клієнта»
7. Ввести у поле «ПІБ клієнта»: «Боднарчук Сергій Ігорович»
8. Ввести у поле «IP»: «172.16.2.102»
9. Ввести у поле «MAC»: «DC:54:FF:12:34:F3»
10. Ввести у поле «Рахунок»: «60»
11. Ввести у поле «Логін»: «ilovesoftserve»
12. Ввести у поле «Пароль»: «123456»
13. Ввести у поле «Телефон»: «0990990909»
14. Ввести у поле «Вулиця»: «Головна»
15. Ввести у поле «Будинок»: «246»
16. Ввести у поле «Квартира»: «9»
17. Ввести у поле «Ел. пошта»: «bodnarchuk@gmail.com»
18. Натиснути кнопку «Додати клієнта»

Результат перевірки: З’явиться повідомлення «Успішно створено клієнта»



Позитивний сценарій:

Дія	***Очікуваний результат***
1. Відкрити будь-який сучасний браузер 2. Перейти за адресою www.korononet.ua/admin	• *Завантажилася головна веб-сторінка сайту* • *Всі поля вводу за замовчуванням порожні*
3. Ввести у поле «Логін»: «admin» *4.* Ввести у поле «Пароль»: «admin»	• *Поля заповнені*
5. Натиснути кнопку «УВІЙТИ»	• *Перехід на адмін-панель*
6. Натиснути кнопку «Додати клієнта»	• *З'являється модальне вікно*
7. Ввести у поле «ПІБ клієнта»: «Боднарчук Сергій Ігорович» 8. Ввести у поле «IP»: «172.16.2.102» 9. Ввести у поле «MAC»: «DC:54:FF:12:34:F3» 10. Ввести у поле «Рахунок»: «60» 11. Ввести у поле «Логін»: «ilovesoftserve» 12. Ввести у поле «Пароль»: «123456» 13. Ввести у поле «Телефон»: «0990990909» 14. Ввести у поле «Вулиця»: «Головна» 15. Ввести у поле «Будинок»: «246» 16. Ввести у поле «Квартира»: «9»	• *Поля заповнені*

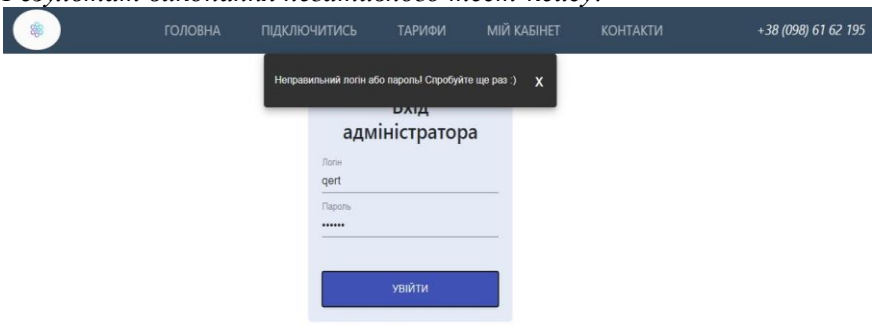
17. Ввести у поле «Ел. пошта»: «bodnarchuk@gmail.com»	
18. Натиснути кнопку «Додати клієнта»	• *З'явилось повідомлення «Успішно створено клієнта»* • *Дані про нового користувача збереглись у базу даних*

Негативний сценарій:

Дія	***Очікуваний результат***
1. Відкрити будь-який сучасний браузер 2. Перейти за адресою www.korononet.ua/admin	• *Відповідна веб-сторінка не завантажилась*
3. Ввести у поле «Логін»: «noname» 4. Ввести у поле «Пароль»: «nopass»	• *Поля заповнені*
5. Натиснути кнопку «УВІЙТИ»	• *З'явилось повідомлення «Неправильний логін або пароль! Спробуйте ще раз :)»*
6. Натиснути кнопку «Додати клієнта»	• *З'являється модальне вікно*
7. Ввести у поле «ПІБ клієнта»: «Боднарчук Сергій Ігорович» 8. Ввести у поле «IP»: «172.16.2.102» 9. Ввести у поле «MAC»: «F3:77:00:88:00:F3» 10. Ввести у поле «Рахунок»: «0» 11. Ввести у поле «Логін»: «ilovesoftserve» 12. Ввести у поле «Пароль»: «123456» 13. Ввести у поле «Телефон»: «tel0fo$%n» 14. Ввести у поле «Вулиця»: «Головна» 15. Ввести у поле «Будинок»: «ten» 16. Ввести у поле «Квартира»: «six»	• *Поля заповнені*

17. Ввести у поле «Ел. пошта»: «bodnarchuk@gmail.com»	
18. Натиснути кнопку «Додати клієнта»	• *З'явилось повідомлення «Дана IP адреса уже використовується»* • *З'явилось повідомлення «Даний логін уже використовується»* • *З'явилось повідомлення «Введено некоректні дані»* • *Дані про нового користувача не збереглись у базу даних*

Результат виконання негативного тест-кейсу:



ДОДАТОК З.

Форма для оцінювання
виконання курсової роботи
з дисципліни «Інженерія програмного забезпечення»
Пояснювальна записка

Виконавці _____

1. Чи присутні всі виконавці роботи?
   - Так (1 бал)
   - Ні (0 балів)
2. Чи складено і подано графік виконання робіт і розподіл обов'язків?
   - Так (1 бал)
   - Ні (0 балів)
3. Оцініть виконання робіт, запланованих за графіком (технічне завдання, діаграма прецедентів, діаграма послідовності):
   - Повністю виконано всі завдання (3 бали)
   - Виконано більшість завдань (2 бали)
   - Виконано меншість завдань (1 бал)
   - Не виконано жадного завдання (0 балів).
4. Чи визначилися виконавці з програмними засобами виконання програми?
   - Так (1 бал)
   - Ні (0 балів)
5. Чи розроблено і наведено схему бази даних?
   - Так (2 бали)
   - Ні (0 балів)
6. Чи продемонстровано інтерфейс майбутньої програми?
   - Наведено фрагменти готового інтерфейсу (2 бали)
   - Створено прототипи інтерфейсу (1 бал)
   - Не наведено (0 балів).

Коментарі/побажання:

Всього: ____ /10 балів

ДОДАТОК І.

Форма для оцінювання програмної розробки,
виконаної в рамках курсової роботи
з дисципліни «Інженерія програмного забезпечення»
Проект розробленого ПЗ. Функціональна частина

Тема розробки: _____
Виконавці розробки: _____
Розробка використана засобами:
Мова програмування _____, СУБД _____

1. Реалізація функцій, зазначених у завданні?
   ☐ реалізовано меншість функцій (2 бали)
   ☐ реалізовано половину функцій (3 бали)
   ☐реалізовано більшість функцій (4 бали)
   ☐ повністю реалізовані (5 балів)

2. Оцініть зручність та зрозумілість інтерфейсу:
   ☐ дуже зручний та зрозумілий (5 балів)
   ☐ вивчення інтерфейсу вимагає деякого часу, виникали затримки при виконанні завдання (3 бали)
   ☐ ускладнення з орієнтацією та навігацією по програмі (2 бали).

3. Оцініть дизайн інтерфейсу програми:
   ☐ послідовний та лаконічний (5 балів)
   ☐ перевантажений елементами (4 бали)
   ☐ невідповідність у оформленні та реалізації дій (3 бали)
   ☐ викликає напруження при тривалій роботі (2 бали).

4. Чи реалізовано багатокористувацький доступ з визначеними правами користувачів:
   ☐ Так (5 балів)
   ☐ Частково (3 бали)
   ☐ Ні (1 бал)
5. Чи реалізовано взаємодію з базою даних:
   ☐ Так (5 балів) ☐Ні (0 балів)

6. Чи виконувалася перевірка щодо застосування програми на різних платформах/ у різних браузерах?
   ☐ Так (5 бали) (вкажіть ОС/браузери, під якими працює програма:_____
   ☐ Ні (2 балів)
7. Оцініть якість написання програмного коду:
   ☐ кожна форма/сторінка має свою змістовну назву, код легко читати, використано послідовну систему позначень (іменувань) програмних елементів, змінено їх назви за замовчуванням, використовуються коментарі. Оригінальність коду (5 балів).
   ☐ у коді наявна деяка неузгодженість у позначенні програмих елементів, для деяких не змінено стандарні назви, рідко зустрічаються коментарі (3 бали).
   ☐ більшість коду згенеровано автоматично, відсутні коментарі та умовні позначення елементів (1 бал) .
8. Оцініть швидкодію та зворотній зв'язок з користувачем:
   ☐ операції виконуються швидко і безпомилково. В програмі реалізовано засоби підтримки користувача (5 балів).
   ☐ при роботі з програмою виникли незначні помилки, проте реалізовані програмні засоби допомагали їх виправити (3 бали).
   ☐ робота з програмою викликала значні помилки, які важко було виправити без безпосереднього втручання у код програми (1 бал).

   Всього: _____/ 20 балів
   Перевірили:_____

ДОДАТОК К.

Форма для оцінювання програмної розробки,
виконаної в рамках курсової роботи
з дисципліни «Інженерія програмного забезпечення»
Захист курсових робіт

1. Дизайн та зручність інтерфейсу :

□	□	□	□	□	□
0	1	2	3	4	5

2. Взаємодія з користувачем:

□	□	□	□	□	□
0	1	2	3	4	5

3. Подання результатів обробки:

□	□	□	□	□	□
0	1	2	3	4	5

4. Реалізація функцій відповідно до ТЗ:

□	□	□	□	□	□
0	1	2	3	4	5

5. Тестування розробленого програмного продукту :

□	□	□	□	□	□
0	1	2	3	4	5

6. Опис логічної структури програми (діаграми, схеми):

□	□	□	□	□	□
0	1	2	3	4	5

7. Оформлення програмної документації

□	□	□	□	□	□
0	1	2	3	4	5

8. Орієнтація у програмі та програмному коді:

□	□	□	□	□	□
0	1	2	3	4	5

Внесок кожного учасника у спільну розробку (у %):

_____ _____ _____

Всього: _____(з 40 балів)

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

DFD – *data flow diagrams,* діаграма потоків даних

IDEF0 – *Function Modeling,* методологія функціонального моделювання і графічного опису процесів

NoSQL – *non (relational) SQL, not only SQL*, нереляційні БД

QA – *Quality Assurance*, забезпечення якості

QC – *Quality Control*, контроль якості

SDLC – *Software Development Life Cycle,*

SRS – *Software Requirements Specification,* Специфікація вимог до програмного забезпечення

UCD – *Use Case Diagram,* діаграма варіантів використання або прецедентів

UI – *User Interface,* інтерфейс користувача

UML – *Unified Modeling Language,* Уніфікована мова моделювання

UX– *User Experience*, користувацький досвід

АСУ – автоматизована система управління

БД – база даних

ЕОМ – електронно-обчислювальна машина

ОС – операційна система

ПЕОМ – персональна електронно-обчислювальна машина

ПЗ – програмне забезпечення

ПК – персональний комп’ютер

ПП – програмний продукт

СУБД – система управління базами даних

СПИСОК ДЖЕРЕЛ ЛІТЕРАТУРИ

1. Інженерія програмного забезпечення: конспект лекцій / уклад.: Танасюк Ю В. Вацек Д. О. – Чернівці : Чернівецький національний університет імені Юрія Федьковича, 2022. 204 с. (електронне видання).
2. 3. Farley D. Modern software engineering: doing what works to build better software faster. Addison-Wesley Professional, 2021. 256 p.
4. Мартін Р. Чиста архітектура. – Харків: Фабула, 2019. 368 с.
5. Мартін Р. Чистий Agile. Назад до основ. – Харків: Фабула, 2021. 224 с.
6. Ousterhout J. A philosophy of software design. Yaknyam Press, 2018. 190 p.
7. Thomas D. The pragmatic programmer: your journey to mastery, 20th anniversary edition. Addison-Wesley, 2019. 352 p.
8. Richards M. Fundamentals of software architecture: an engineering approach. O'Reilly Media, 2020. 419 p.
9. Mohan G. Full stack testing. A practical guide for delivering high quality software. O'Reilly Media, 2022. 406 p.
10. Левус Є.В. Вступ до інженерії програмного забезпечення: навч. посібник / Левус Є.В, Мельник Н.Б. – Львів: Видавництво Львівської політехніки, 2017. 280 с.
11. Sommerville I. Software Engineering Global Edition. Pearson Education, 2015. 624 p.
12. Sullivan S. Designing for wearables: effective UX for current and future devices, 1st edition. O'Reilly Media, 2017. 194 p.
13. Fowler M. UML distilled: A brief guide to the standard object modeling language, 3rd edition. Addison-Wesley Professional, 2016. 208 p.
14. Booch G. Unified modeling language user guide. Addison-Wesley Professional, 2017. 504 p.

15. Rumpe B. Modeling with UML: language, concepts, methods, 1st edition. Springer, 2016. 295 p.
16. Авраменко А.С. Тестування програмного забезпечення: навч. посібник / Авраменко А.С., Авраменко В.С., Косенюк Г.В . – Черкаси : ЧНУ імені Богдана Хмельницького, 2017. 284 с.
17. Agile manifesto: Understanding Agile values and Principles. URL : https://www.softwaretestinghelp.com/agile-manifesto
18. Scrum Guides. URL : https://scrumguides.org/
19. Design patterns. URL : https://refactoring.guru/design-patterns
20. The clean architecture – Beginner's guide. URL : https://betterprogramming.pub/the-clean-architecture-beginners-guide-e4b7058c1165
21. Agile Software Development Metrics and KPIs that Help Optimize Product Delivery. URL : https://www.altexsoft.com/blog/business/agile-software-development-metrics-and-kpis-that-help-optimize-product-delivery/
22. Agile in a nutshell. URL : http://www.agilenutshell.com/

ЗМІСТ

Навчальне видання

ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Методичні вказівки до лабораторних робіт

Укладачі Танасюк Юлія Володимирівна,
Одайська Христина Савеліївна

Відповідальний за випуск Г. І. Воробець

Літературний редактор О. В. Колодій

Підписано до друку 27.11.2022. Формат 60 х 84/16
Ум. друк. арк. 8.
Обл.-вид. арк. 8,6. Тираж 100