**МІНІСТРЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ПОЛТАВСЬКИЙ ПОЛІТЕХНІКА ІМЕНІ ЮРІЯ КОНДРАТЮКА»**

НАВЧАЛЬНИЙ НАУКОВИЙ ІНСТИТУТ

ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І РОБОТОТЕХНІКИ

КАФЕДРА КОМП’ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

**Лабораторна робота 5**

з дисципліни «Програмне забезпечення комп'ютерних систем»

за темою «Процес розробки та тестування програмного забезпечення»

**Виконав:**

Студент 501-ТК

Коренівський Олег Григорович

**Перевірив:**

професор

Фесенко Тетяна Григорівна

**Полтава**

**2024Вступна частина**

У даній лабораторній роботі описується процес моделювання етапів розробки та тестування програмного забезпечення з використанням методологій IDEF0 (SADT) та DFD. Метою роботи є побудова функціональної моделі, що включає опис основних етапів розробки ПЗ, їх взаємозв'язків, визначення вхідних та вихідних даних, механізмів, обмежень, а також деталізація процесів за допомогою декомпозицій у нотації DFD із врахуванням сховищ даних і зв’язків між ними.

# **Мета роботи**

Дослідити та побудувати функціональні діаграми для процесу "Розробка та тестування програмного забезпечення" у нотації IDEF0 і DFD. Визначити мету, точку зору, межі моделі, а також виконати декомпозицію основних процесів.

# **Завдання до роботи**

1. Побудувати контекстну діаграму A-0 у нотації IDEF0.
2. Виконати декомпозицію процесу A0 на 4 підпроцеси.
3. Побудувати декомпозиції кожного з 4 підпроцесів у нотації DFD, використовуючи 3-4 підпроцеси та одне сховище даних.
4. Оформити звіт відповідно до вимог.

# **1. Опис моделі**

## **1.1. Контекстна діаграма A-0**

Контекстна діаграма відображає весь процес розробки та тестування програмного забезпечення як єдиний блок. Вхідними даними є вимоги до програмного забезпечення, які обробляються у процесі розробки. Контрольні елементи включають технічне завдання та стандарти. Ресурси для виконання процесу представлені інструментами розробки та командою розробників. Вихідним результатом є протестоване програмне забезпечення.

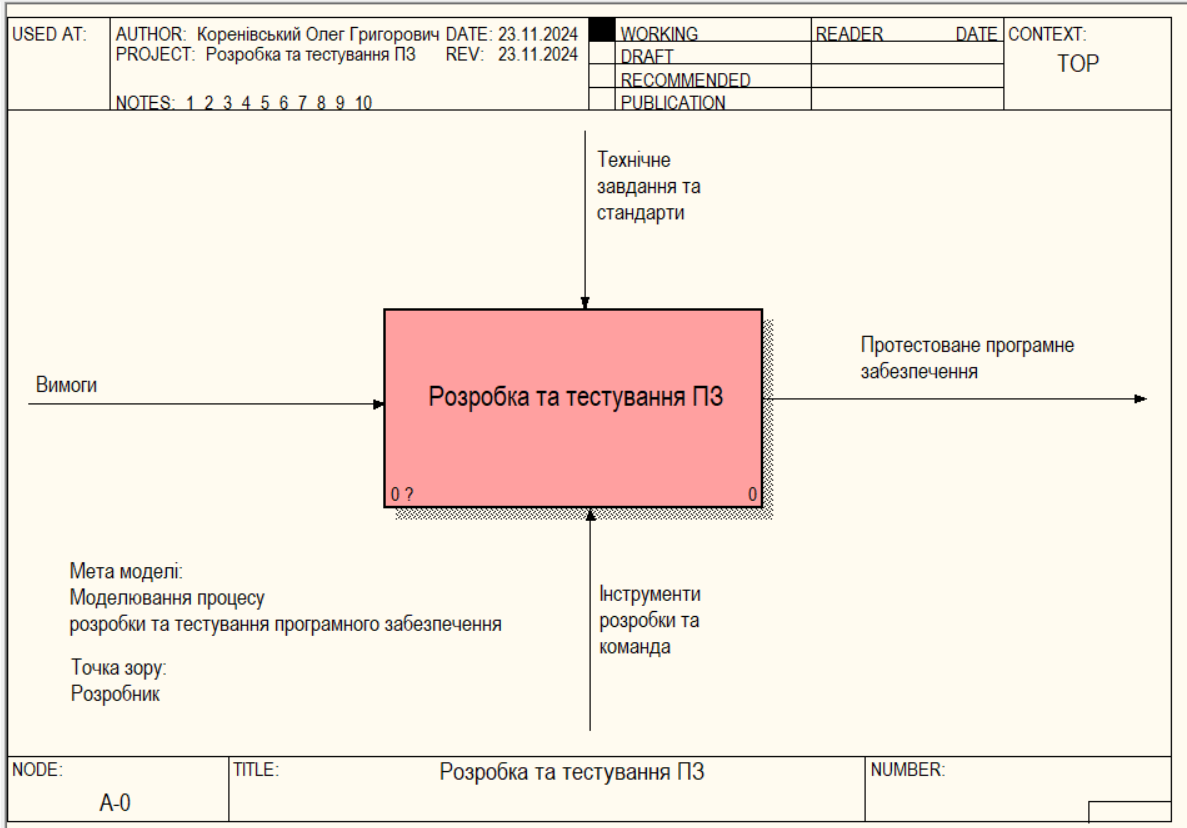


Рисунок 1 – Контекстна діаграма процесу розробки та тестування програмного забезпечення (Зроблено в програмі Process Modeler)

Таблиця 1.1 – Глосарій термінів та ключових слів процесу «Розробка та тестування програмного забезпечення»

|  |  |  |
| --- | --- | --- |
| **Ім’я стрілки (Arrow Name)** | **Визначення стрілки (Arrow Definition)** | **Тип стрілки (Arrow Type)** |
| Вимоги | Запит на створення або оновлення програмного забезпечення. | Вхідний потік даних (Input) |
| Технічне завдання та стандарти | Документація, яка регламентує процес розробки та тестування ПЗ. | Керуючий потік (Control) |
| Протестоване ПЗ | Програмний продукт, який пройшов перевірку та готовий до впровадження. | Вихідний потік даних (Output) |
| Інструменти розробки та команда | Ресурси, включаючи програмні засоби й спеціалістів, які виконують завдання. | Механізм (Mechanism) |

## **1.2. Декомпозиція A0**

Процес розробки та тестування програмного забезпечення розділений на такі підпроцеси:

1. **Аналіз вимог:** Оцінка вимог до програмного забезпечення, визначення технічних задач.
2. **Розробка коду:** Створення коду на основі технічного завдання.
3. **Тестування:** Перевірка функціональності та якості коду.
4. **Реліз:** Підготовка програмного забезпечення до впровадження та випуску.

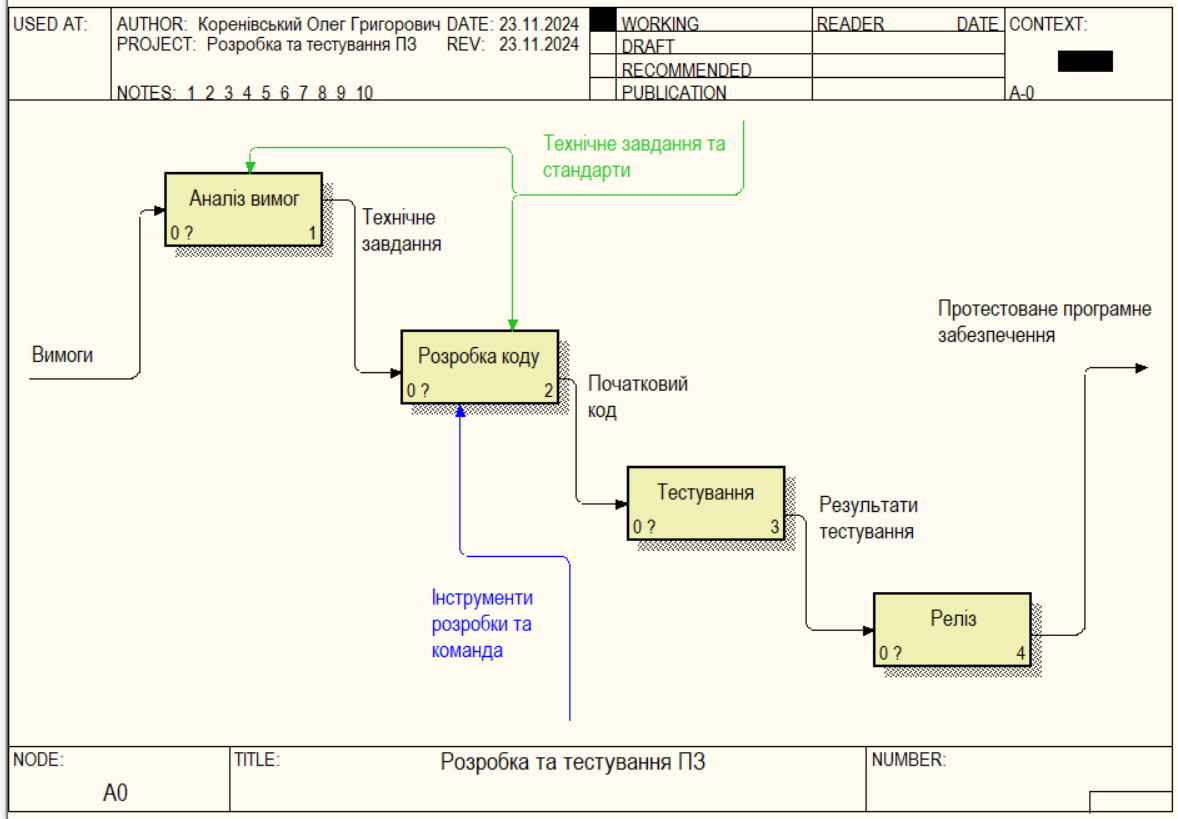


Рисунок 2 – Декомпозиція контекстна діаграма процесу «Розробки та тестування програмного забезпечення»

Глосарій етапів робіт діаграм декомпозицій процесів ремонту розробки та тестування програмного забезпечення IDEF0 (таблиця 1.2).

Таблиця 1.2 – Глосарій етапів робіт діаграм декомпозицій процесів

у стандарті IDEF0

|  |  |  |
| --- | --- | --- |
| ***Ім’я роботи***  ***(Activity Name)*** | ***Визначення (Definition)*** | ***Призначення елементу*** |
| Аналіз вимог | Визначення та уточнення вимог замовника, їх обробка та створення технічного завдання. | Процес |
| Розробка коду | Створення програмного коду на основі технічного завдання, його інтеграція та тестування. | Процес |
| Тестування | Перевірка працездатності системи та відповідності її вимогам. | Процес |
| Реліз | Завершення розробки ПЗ, його впровадження та підготовка до експлуатації. | Процес |
| Вимоги | Початкові дані, що визначають, яким має бути програмне забезпечення. | Вхідний потік даних (Input) |
| Технічне завдання та стандарти | Документовані правила та вимоги, які керують процесом розробки. | Керуючий потік (Control) |
| Технічне завдання | Результат процесу "Аналіз вимог", який стає основою для розробки коду. | Потік даних між процесами |
| Початковий код | Код, створений під час розробки, який передається на тестування. | Потік даних між процесами |
| Результати тестування | Звіт про якість програмного забезпечення після перевірки, переданий для релізу. | Потік даних між процесами |
| Протестоване ПЗ | Програмний продукт, готовий до впровадження після завершення тестування. | Вихідний потік даних (Output) |
| Інструменти розробки та команда | Фахівці та програмно-апаратне забезпечення, що використовуються для створення та впровадження ПЗ. | Механізм (Mechanism) |

**1.3. Декомпозиція процесу "Аналіз вимог" A1**

На цьому етапі відбувається визначення потреб замовника, збір і аналіз інформації, а також формування документації, яка регламентує вимоги до програмного забезпечення.

* **Підпроцеси:**
  + - 1. **Збір вимог:** отримання інформації від замовника або користувачів, запис даних.
      2. **Аналіз вимог:** визначення повноти, точності й суперечностей у вимогах.
      3. **Формування технічного завдання:** оформлення аналізованих вимог у вигляді ТЗ.
* **Сховище:** "База вимог" (зберігає зібрані та аналізовані дані).

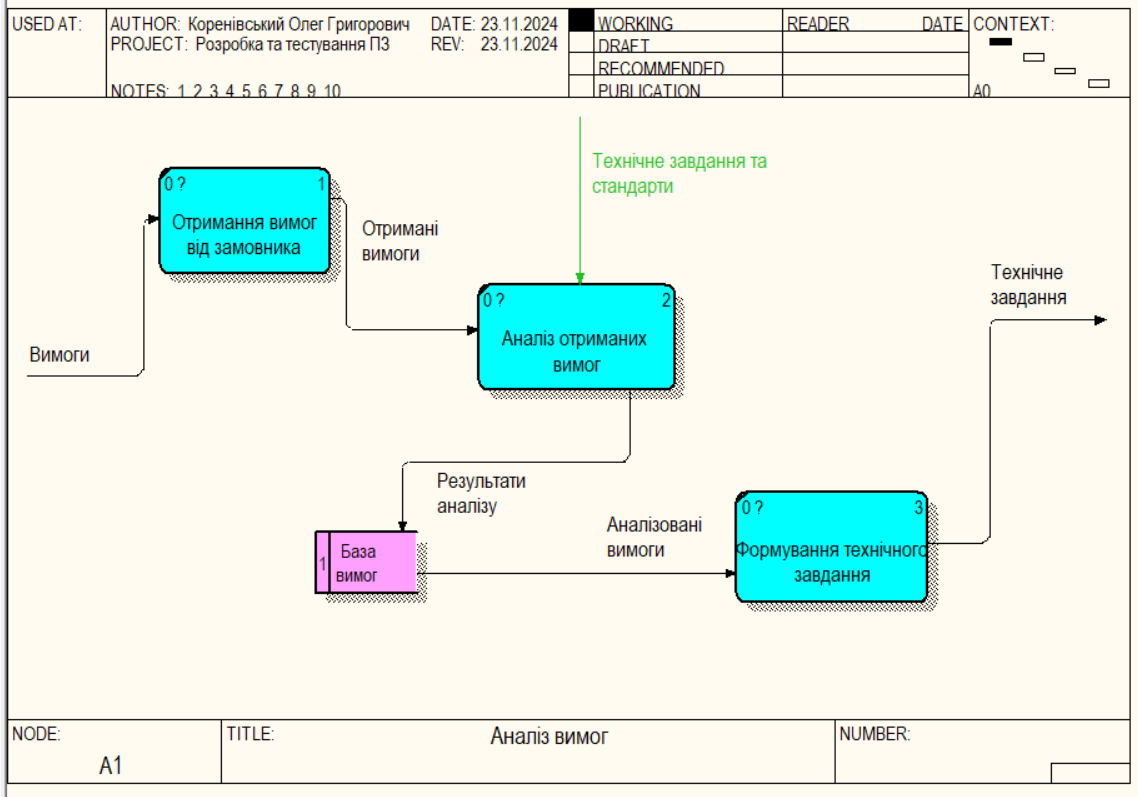


Рисунок 3 – Декомпозиція процесу «Аналіз вимог»

Таблиця 1.3 – Глосарій термінів та ключових слів діаграми "Аналіз вимог", що розроблені у нотації DFD

|  |  |  |
| --- | --- | --- |
| ***Ключові слова*** | ***Визначення*** | ***Призначення елементу*** |
| Збір вимог | Отримання початкових даних про потреби клієнта. | Процес |
| Аналіз вимог | Вивчення та уточнення зібраних вимог, виявлення суперечностей. | Процес |
| Формування технічного завдання | Перетворення проаналізованих вимог у документ (ТЗ). | Процес |
| База вимог | Сховище даних, де зберігаються зібрані та оброблені вимоги. | Сховище |
| Вимоги | Вхідні дані, надані клієнтом. | Вхідний потік даних (Input) |
| Отримані вимоги | Вимоги, які були зібрані на етапі збору, передані на аналіз. | Потік даних між процесами |
| Результати аналізу | Вимоги, оформлені у структуровану форму після аналізу. | Потік даних між процесами |
| Аналізовані вимоги | Дані, витягнуті зі сховища для формування ТЗ. | Потік даних між процесами |
| Технічне завдання (ТЗ) | Фінальний документ, що описує вимоги до системи. | Вихідний потік даних (Output) |

## **1.4. Декомпозиція процесу "Розробка коду" A2**

Цей етап охоплює створення архітектури програмного забезпечення, написання програмного коду та його інтеграцію у працюючий продукт.

* **Підпроцеси:**
  1. **Розробка архітектури:** створення структури системи, визначення її компонентів.
  2. **Написання модулів:** програмування окремих функціональних частин ПЗ.
  3. **Інтеграція модулів:** об’єднання розроблених модулів у єдину систему.
  4. **Розробка інтерфейсу:** створення зручного користувацького інтерфейсу.
* **Сховище:** "Кодова база" (зберігає написані модулі).

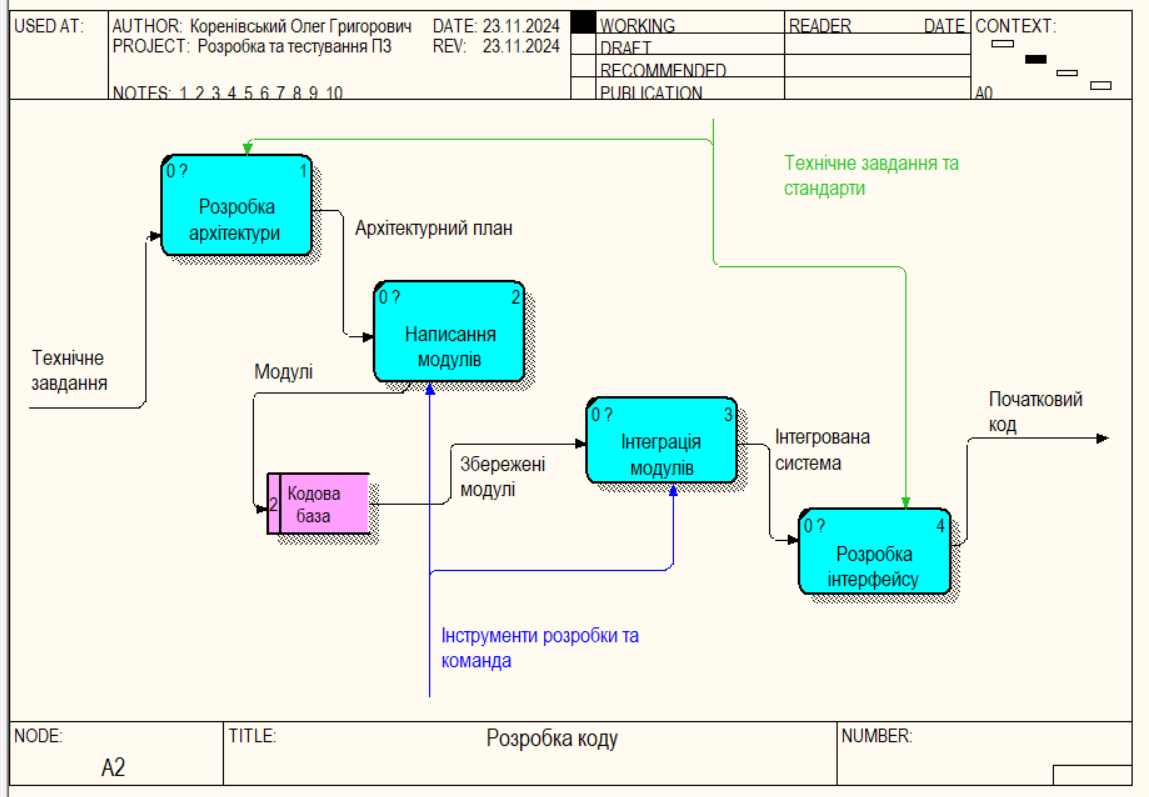


Рисунок 4 – Декомпозиція процесу «Розробка коду»

Таблиця 1.4 – Глосарій термінів та ключових слів діаграми " Розробка коду", що розроблені у нотації DFD

|  |  |  |
| --- | --- | --- |
| ***Ключові слова*** | ***Визначення*** | ***Призначення елементу*** |
| Розробка архітектури | Планування структури програмного забезпечення. | Процес |
| Написання модулів | Програмування окремих функціональних частин системи. | Процес |
| Інтеграція модулів | Об’єднання окремих частин у працюючу систему. | Процес |
| Розробка інтерфейсу | Створення зручного графічного інтерфейсу для користувачів. | Процес |
| Кодова база | Сховище, яке містить створені модулі. | Сховище |
| Технічне завдання | Документ, який визначає функціональні вимоги. | Вхідний потік даних (Input) |
| Архітектурний план | Структурований опис майбутньої системи. | Потік даних між процесами |
| Модулі | Окремі компоненти системи, які зберігаються у базі. | Потік даних між процесами |
| Інтегрована система | Система, що об’єднує всі компоненти в одну програму. | Потік даних між процесами |
| Збережені модулі | Модулі, записані у сховище (кодову базу) для подальшої інтеграції. | Потік даних між процесами |
| Початковий код | Фінальний результат розробки. | Вихідний потік даних (Output) |

## **1.5. Декомпозиція процесу "****Тестування" A3**

Тестування забезпечує перевірку якості створеного програмного забезпечення, включаючи окремі модулі та їх взаємодію в системі.

* **Підпроцеси:**

1. **Розробка тестових сценаріїв:** підготовка набору тестів для перевірки ПЗ.
2. **Виконання модульного тестування:** перевірка роботи окремих модулів.
3. **Виконання інтеграційного тестування:** перевірка коректної роботи модулів у системі.

* **Сховище:** "Результати тестування" (зберігає звіти про виконані тести).

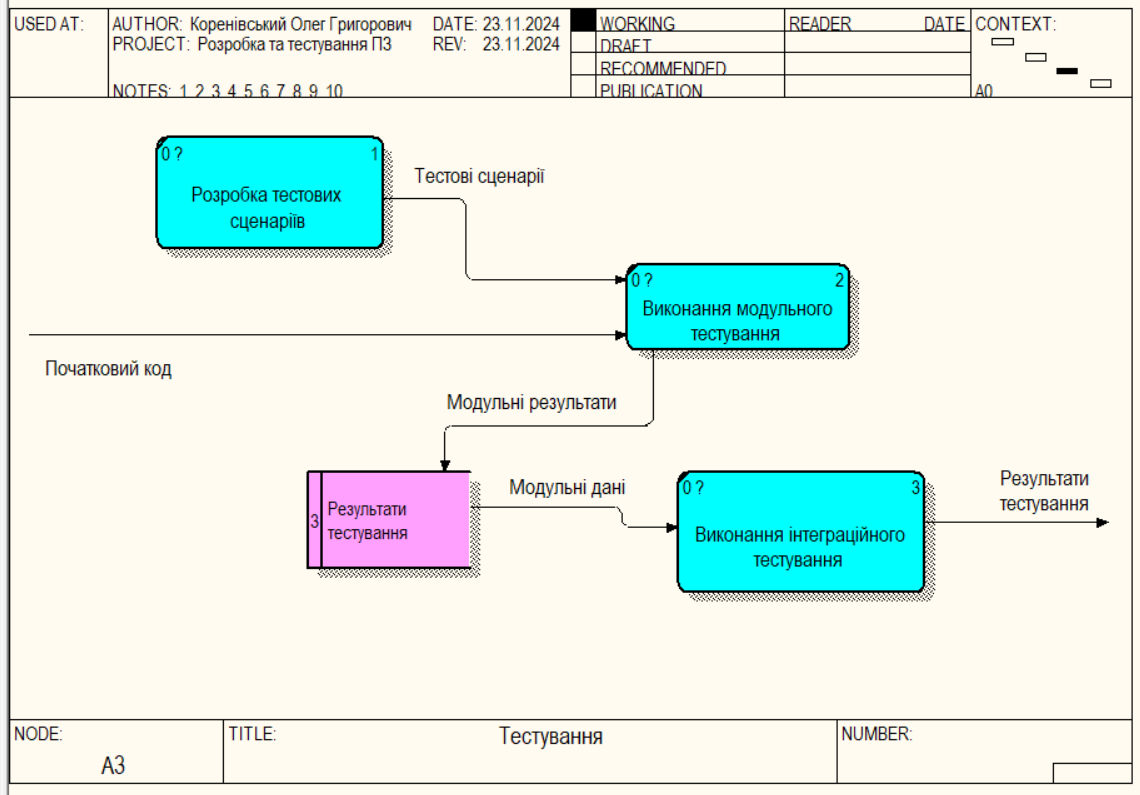


Рисунок 5 – Декомпозиція процесу «Тестування»

Таблиця 1.5 – Глосарій термінів та ключових слів діаграми " Тестування", що розроблені у нотації DFD

|  |  |  |
| --- | --- | --- |
| ***Ключові слова*** | ***Визначення*** | ***Призначення елементу*** |
| Розробка тестових сценаріїв | Підготовка сценаріїв для перевірки функціональності. | Процес |
| Модульне тестування | Тестування окремих частин системи. | Процес |
| Інтеграційне тестування | Перевірка взаємодії компонентів системи. | Процес |
| Результати тестування | Сховище, де зберігаються звіти про виконані тести. | Сховище |
| Початковий код | Код, який необхідно перевірити на коректність. | Вхідний потік даних (Input) |
| Тестові сценарії | Набір перевірок, які використовуються під час тестування. | Керуючий потік (Control) |
| Модульні результати | Результати тестування окремих модулів. | Потік даних між процесами |
| Модульні дані | Дані з результатів модульного тестування, необхідні для інтеграційного тестування. | Потік даних між процесами |
| Результати інтеграційного тестування | Вихідні дані після перевірки взаємодії всіх модулів. | Вихідний потік даних (Output) |

## **1.6. Декомпозиція процесу** **"****Реліз" A4**

Завершальний етап роботи над ПЗ, який включає підготовку продукту до випуску, створення документації та перевірку після впровадження.

* **Підпроцеси:**

1. **Підготовка релізу:** узгодження та створення готової версії ПЗ.
2. **Документування ПЗ:** створення супровідної документації.
3. **Впровадження ПЗ:** передача ПЗ у використання.
4. **Перевірка роботи після впровадження:** тестування продукту у робочому середовищі.

* **Сховище:** "Архів релізів" (зберігає завершені релізи та документацію).

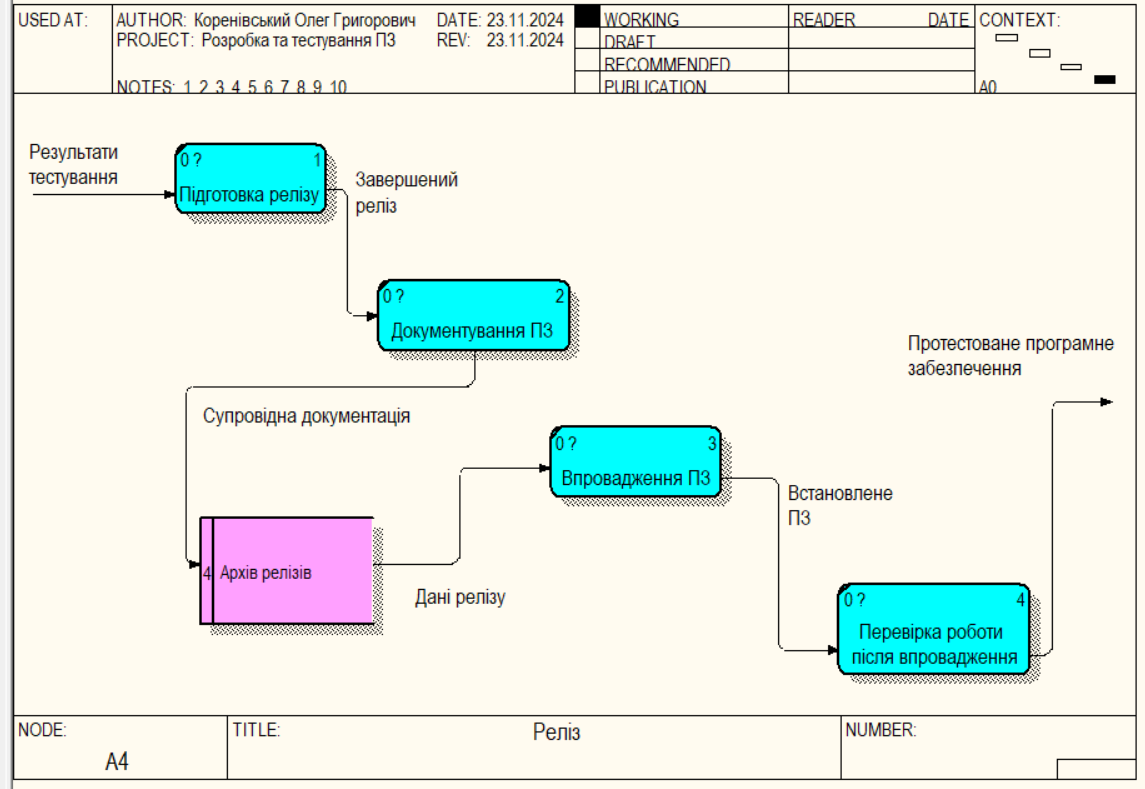


Рисунок 6 – Декомпозиція процесу «Реліз»

Таблиця 1.4 – Глосарій термінів та ключових слів діаграми " Реліз", що розроблені у нотації DFD

|  |  |  |
| --- | --- | --- |
| ***Ключові слова*** | ***Визначення*** | ***Призначення елементу*** |
| Підготовка релізу | Формування фінальної версії продукту для впровадження. | Процес |
| Документування ПЗ | Створення супровідної документації. | Процес |
| Впровадження ПЗ | Встановлення готового продукту в робоче середовище. | Процес |
| Перевірка роботи після впровадження | Контроль працездатності продукту після встановлення. | Процес |
| Архів релізів | Сховище даних, яке містить усі завершені версії програмного забезпечення. | Сховище |
| Протестоване ПЗ | Готовий до впровадження програмний продукт. | Вхідний потік даних (Input) |
| Супровідна документація | Інструкції для користувачів і адміністраторів. | Потік даних між процесами |
| Дані релізу | Фінальні матеріали для впровадження. | Потік даних між процесами |
| Встановлене ПЗ | Програмне забезпечення, успішно впроваджене в середовище користувачів. | Потік даних між процесами |
| Готовий продукт | Завершений програмний продукт, готовий до експлуатації. | Вихідний потік даних (Output) |

## **1.7.** **Діаграма дерева вузлів**

Діаграма дерева вузлів відображає ієрархію процесу розробки та тестування програмного забезпечення.

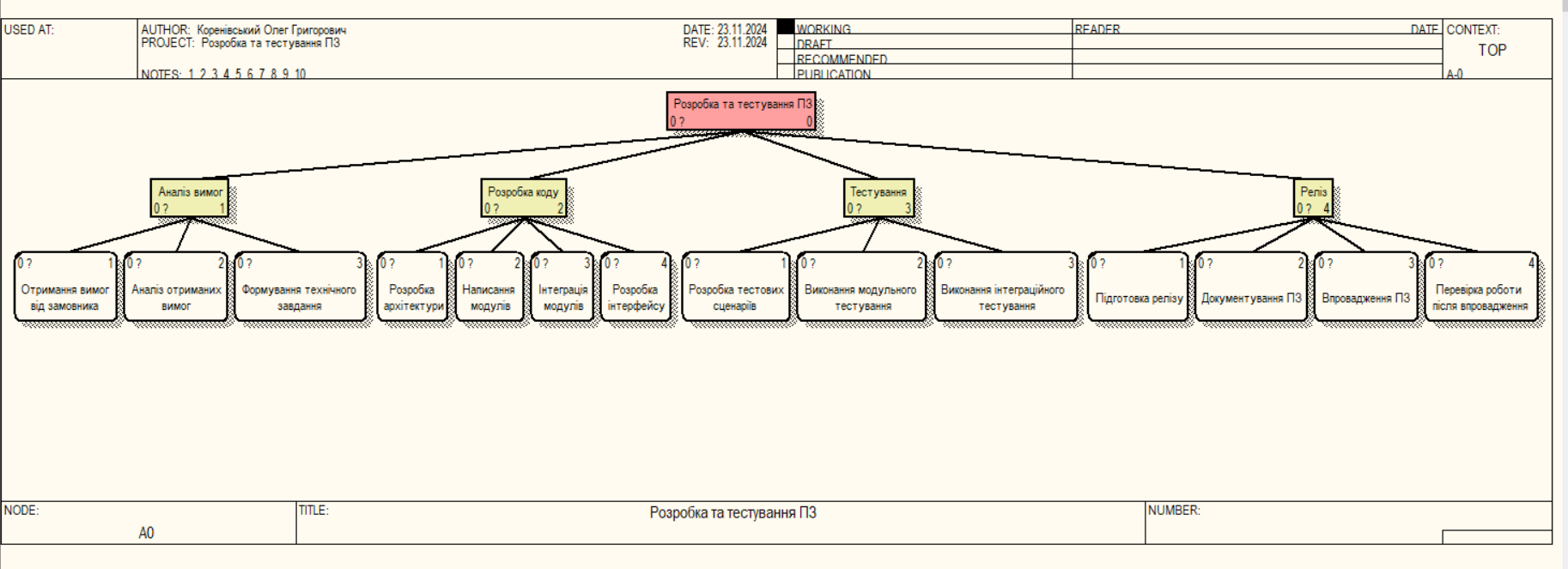


Рисунок 7 – Діаграма дерева вузлів процесу «Розробки та тестування програмного забезпечення»

**Властивості моделі**

**Основні властивості моделі та її контекст**

Властивості розділу "General" відображають основну інформацію про модель, зокрема її назву, автора, назву проєкту та часову рамку (AS-IS). Ці дані формують основу для ідентифікації моделі та її контексту.

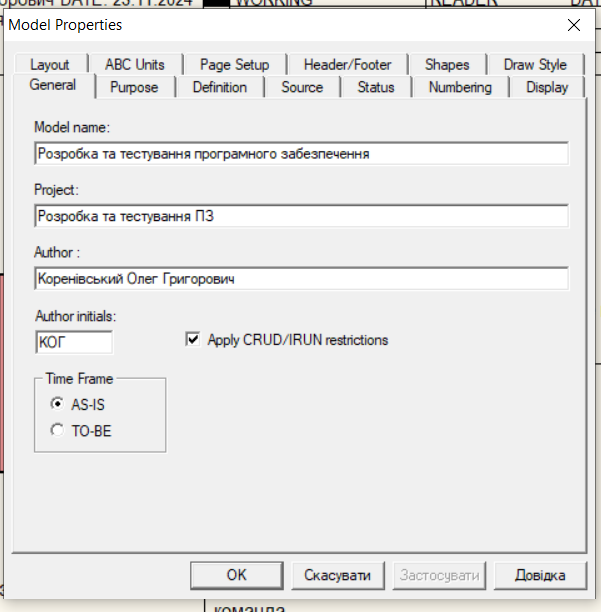


Рисунок 8 – Загальні властивості

**Мета моделі та точка зору на процес розробки**

У властивостях "Purpose" визначається мета моделі, яка полягає у формалізації процесу розробки та тестування програмного забезпечення. Також вказано точку зору (Viewpoint) — вона відображає підхід розробників, що акцентує увагу на технічних аспектах реалізації..

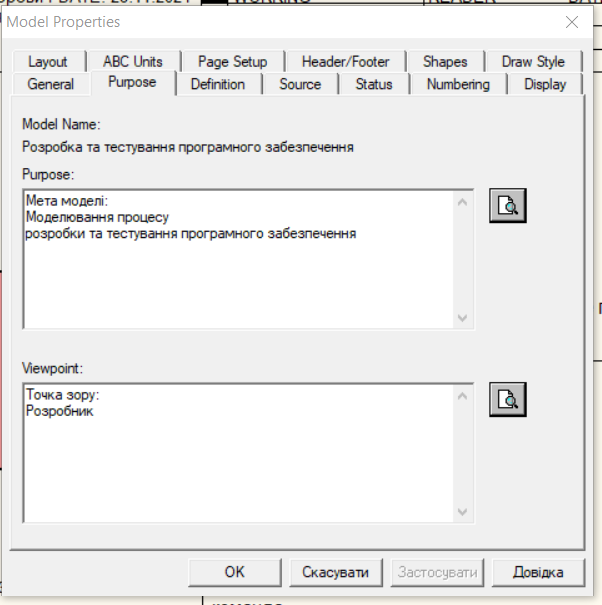


Рисунок 9 – Властивості «Purpose»

**Опис структури моделі та взаємозв’язків між процесами**

На рисунку наведено властивості "Definition", які описують загальну структуру моделі. Тут вказано ключові процеси, такі як аналіз вимог, розробка коду, тестування та реліз. Також зазначено взаємозв’язки між процесами через потоки даних, механізми й обмеження. Цей розділ допомагає зрозуміти основну логіку моделі та її відповідність цілям проєкту. Властивості спрямовані на деталізацію кожного етапу для забезпечення прозорості моделювання.

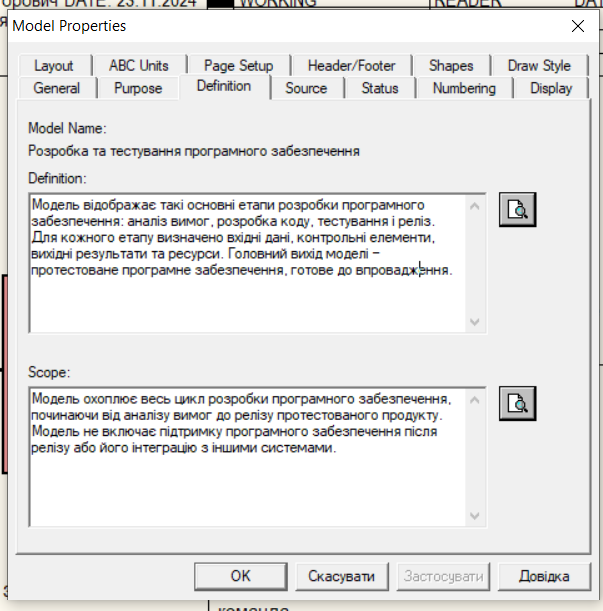


Рисунок 10 – Властивості «Definition»

**Опис одиниць роботи та їх ролі в процесі розробки**

На рисунку зображено властивості "Unit of Work", які деталізують основні одиниці роботи в моделі. Включено такі етапи: аналіз вимог, розробка коду, тестування та реліз. Для кожного етапу вказано їхній взаємозв’язок через потоки даних та сховища, що забезпечує послідовність ієрархії процесів. Властивості визначають, як кожна одиниця роботи інтегрується у загальну структуру моделі, формуючи чіткий і логічний робочий процес. Це важливо для візуалізації залежностей між процесами.

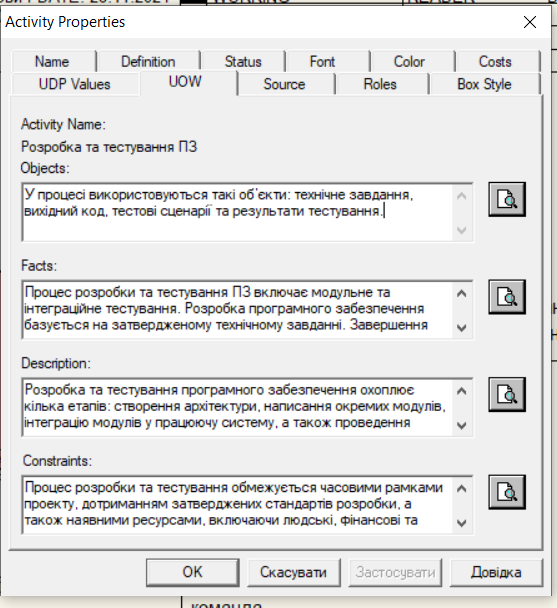


Рисунок 11 – Властивості «UOW»

# **Відповіді на контрольні питання**

**1. Для чого необхідні IDEF3-моделі?**

IDEF3-моделі необхідні для опису логіки взаємодії інформаційних потоків, відносин між процесами обробки інформації та об'єктів, що є частиною цих процесів. Вони використовуються для моделювання бізнес-процесів, аналізу завершеності процедур обробки інформації та документування сценаріїв виконання функцій.

**2. Які основні поняття використовуються при створенні функціональної діаграми IDEF3?**

Основні поняття:

* **Одиниці роботи (Unit of Work, UOW)** — центральні компоненти моделі, які зображуються прямокутниками і відображають процеси дії.
* **Стрілки (зв'язки):**
  + Старша стрілка (Precedence).
  + Стрілка відношення (Relational Link).
  + Стрілка потоку об'єктів (Object Flow).
* **Перехрестя (Junction)** — злиття (Fan-in) або розгалуження (Fan-out) стрілок.
* **Об'єкти посилань** (типи: OBJECT, GOTO, UOB, NOTE, ELAB).

**3. Для чого в діаграмах IDEF3 використовуються перехрестя?**

Перехрестя використовуються для злиття (Fan-in) або розгалуження (Fan-out) стрілок, а також для відображення безлічі подій, які можуть або повинні бути завершені перед початком наступної роботи. Вони не можуть одночасно використовуватись для обох функцій.

**4. Перерахуйте основні об'єкти IDEF3, їх опис і призначення.**

* **Одиниці роботи (UOW):** Відображають процеси дії.
* **Стрілки:** Показують зв'язки між одиницями роботи та іншими елементами.
* **Перехрестя:** Логіка злиття та розгалуження процесів.
* **Об'єкти посилань:** Для додаткової інформації та взаємозв'язків між елементами (типи: OBJECT, GOTO, UOB, NOTE, ELAB).

**5. Чим відрізняються синхронні перехрестя від асинхронних?**

* **Синхронні (Synchronous):** Всі попередні процеси мають завершитись одночасно; наступні запускаються одночасно.
* **Асинхронні (Asynchronous):** Попередні процеси можуть завершуватись або розпочинатись незалежно.

**6. Що таке посилання?**

Посилання — це об'єкти, які виражають ідеї, концепції або дані, що не можна напряму пов'язати зі стрілками, перехрестями або роботами. Приклади: GOTO (циклічний перехід), NOTE (документування), UOB (множинне використання роботи).

**7. Чому перехрестя «Виключаюче Або» не може бути синхронним?**

«Виключаюче Або» (XOR) вказує, що лише один попередній процес завершений або один наступний запускається, тому синхронність (одночасність) тут не може бути досягнута за своєю природою.

**8. При виконанні яких проектів краще всього використовувати DFD-методологію?**

DFD-методологію краще використовувати для опису існуючих потоків даних в організації, визначення сховищ даних, аналізу необхідних даних, створення моделей структури даних (ERD) та виділення бізнес-процесів організаці.

**Висновок**

У ході виконання лабораторної роботи було побудовано функціональну модель процесу розробки та тестування програмного забезпечення за методологією IDEF0. Модель відобразила основні етапи цього процесу: аналіз вимог, розробка коду, тестування та реліз.

Було створено контекстну діаграму, яка визначає взаємозв’язок системи із зовнішнім середовищем, і виконано декомпозицію процесу, що дозволило детально описати кожен етап. Крім того, побудовано декомпозиції у нотації DFD, які дозволяють розкрити підпроцеси кожного з етапів, зокрема їх вхідні та вихідні дані, сховища інформації та взаємозв’язки.

Діаграма дерева вузлів допомогла візуалізувати ієрархію моделі та забезпечити її структурованість. Завдяки цьому вдалося краще зрозуміти процеси, що лежать в основі розробки програмного забезпечення, та їх взаємозв’язки.

У результаті виконання роботи вдалося досягти поставленої мети – формалізувати процеси розробки ПЗ, визначити вхідні, вихідні дані, ресурси та обмеження кожного етапу. Модель є зручною для подальшого використання в управлінні розробкою ПЗ і може бути базою для оптимізації процесів.

Ця робота сприяла кращому розумінню методології IDEF0, яка дозволяє формалізувати складні процеси, роблячи їх наочними, логічними та структурованими. Додаткове застосування нотації DFD розширило можливості моделювання. Отримані знання можуть бути використані як у навчанні, так і на практиці в проєктуванні інформаційних систем і управлінні проєктами.