Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____
(повна назва)
Кафедра _____Інформаційних управляючих систем_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____

__Дослідження моделей і методів управління ІТ-проєктом рефакторингу бази даних Інтернет-провайдера
(тема)

Виконав:
студент 2 курсу, групи __УПГІТм-21-1__
_____Дмитро МІРОШНИЧЕНКО_____
(власне ім'я, прізвище)

Спеціальність___122 Комп'ютерні науки___
(код і повна назва спеціальності)
Тип програми _____освітньо-наукова_____
(освітньо-професійна або освітньо-наукова)
Освітня програма Управління проєктами в галузі інформаційних технологій_____
(повна назва освітньої програми)

Керівник проф. каф. ІУС Максим ЄВЛАНОВ
(посада, власне ім'я, прізвище)

Допускається до захисту
Зав. кафедри __  _____ Костянтин ПЕТРОВ
(підпис) (власне ім'я, прізвище)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
Кафедра Інформаційних управляючих систем
Рівень вищої освіти другий (магістерський)
Спеціальність 122 Комп'ютерні науки
(код і повна назва)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Управління проєктами в галузі інформаційних технологій
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри ___
(підпис)
«03» квітня 2023 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Мірошниченку Дмитру Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження моделей і методів управління ІТ-проєктом рефакторингу бази даних Інтернет-провайдера
затверджена наказом університету від 03 квітня 2023 р. № 319 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 16.05.2023 р.
3. Вихідні дані до роботи: науково-технічні публікації; джерела інтернету; науково-технічна література, що стосуються теми кваліфікаційної роботи
4. Перелік питань, що потрібно опрацювати в роботі: аналіз існуючих моделей і методів управління ІТ-проєктом рефакторингу бази даних Інтернет-провайдера, розробка моделі і методу управління ІТ-проєктом рефакторингу бази даних Інтернет-провайдера, адаптація моделі оцінювання трудовитрат на проведення рефакторингу бази даних Інтернет-провайдера, практична апробація результатів магістерської роботи.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Проведення аналізу актуальності дослідження моделей і методів управління ІТ-проєктом рефакторингу бази даних Інтернет-провайдера	03.04.2023	виконано
2	Проведення аналізу існуючих рішень рефакторингу бази даних; моделей і методів управління ІТ-проєктом рефакторингу бази даних Інтернет-провайдера	07.04.2023	виконано
3	Обґрунтування мети створення методу та моделі для визначення та проведення рефакторингу для підвищення гнучкості бази даних Інтернет-провайдера	11.04.2023	виконано
4	Розробка кількісної оцінки складності опису актуальної бази даних до проведення рефакторингу та опису змін, що вносяться для подальшої оцінки складності рефакторингу, що проводиться	14.04.2023	виконано
5	Адаптація моделі оцінювання трудовитрат на проведення рефакторингу бази даних Інтернет-провайдера	19.04.2023	виконано
6	Практична апробація результатів магістерської роботи	27.04.2023	виконано
7	Оформлення пояснювальної записки та графічного матеріалу	03.05.2023	виконано
8	Підготовка презентації	04.05.2023	виконано
9	Подання студентом роботи для перевірки на плагіат	08.05.2023	виконано
10	Надання роботи на підпис науковому керівнику	10.05.2023	виконано
11	Попередній захист роботи	11.05.2023	виконано
12	Надання роботи на рецензію	12.05.2023	виконано
13	Надання роботи на підпис завідувачу кафедри	15.05.2023	виконано
14	Захист кваліфікаційної роботи	17.05.2023	виконано

Дата видачі завдання _03_ __квітня___ 2023 р.

Студент _____
(підпис)

Керівник роботи _____ проф. каф. ІУС Максим ЄВЛАНОВ
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи: 104 стор., 24 рис., 1 табл., 26 джерел, 1 додаток.

БАЗА ДАНИХ, ІНТЕРНЕТ-ПРОВАЙДЕР, ІТ-ПРОЄКТ, КІЛЬКІСНА ОЦІНКА, РЕФАКТОРИНГ, ТЕОРЕТИКО-МНОЖИННИЙ АПАРАТ ОПИСУ ERD, COCOMO2, PRINCE2, RFC, SQL, USER STORY.

Метою даної роботи є дослідження моделей і методів управління ІТ-проєктом рефакторингу бази даних Інтернет-провайдера задля підтримки гнучкого створення замовлень.

Об’єктом дослідження в рамках магістерської кваліфікаційної роботи є процес рефакторингу бази даних інформаційно-облікової системи фірми Інтернет-провайдера.

Предметом дослідження являються методи аналізу та вирішення задачі рефакторингу бази даних Інтернет-провайдера.

Теоретичними результатами дослідження є опис подальшого розвитку моделі оцінювання трудовитрат COCOMO2 для ІТ-проєктів рефакторингу баз даних.

Практичним результатом є апробація отриманих наукових результатів під час ініціації ІТ-проєкту рефакторингу бази даних Інтернет-провайдера.

Новизна дослідження полягає в розробці моделі оцінювання складності проведення рефакторингу бази даних та отриманні подальшого розвитку моделі оцінювання трудовитрат COCOMO2 для ІТ-проєктів рефакторингу баз даних.

ABSTRACT

The explanatory note to the qualification work: 104 p., 24 fig., 1 tabl., 26 sources, 1 appendix.

COCOMO2, DATABASE, INTERNET PROVIDER, IT-PROJECT, PRINCE2, QUANTITATIVE EVALUATION, REFACTORING, RFC, SQL, THEORETICAL MULTIPLE ERD DESCRIPTION APPARATUS, USER STORY.

The purpose of this work is to investigate models and methods of managing the IT project of refactoring the Internet provider's database to support the flexible creation of orders.

The object of research within the framework of the master's qualification work is the process of refactoring the database of the information and accounting system of the Internet provider.

The subject of the research is methods of analysis and solution of the issue of refactoring the Internet provider's database.

The theoretical results of the research are a description of the further development of the COCOMO2 labor cost estimation model for database refactoring IT projects.

The practical results are the approbation of the obtained scientific results during the initiation of the IT project of refactoring the Internet provider's database.

The novelty of the research consists in the development of a model for assessing the complexity of database refactoring and obtaining the further development of the COCOMO2 labor cost assessment model for database refactoring IT projects.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних;

ЖЦ – життєвий цикл;

ІС – інформаційна система;

ПЗ – програмне забезпечення;

COCOMO – Constructive Cost Model;

DDL – Data Definition Language;

ERD – Entity – Relation Diagram;

PMBOK – Project Management Body of Knowledge’s;

PRINCE2 – PRojects IN Controlled Enviroments;

RFC – Request for Changes.

ВСТУП

Провайдер – організація, яка надає послуги доступу до глобальної мережі Інтернет. Клієнтом провайдера може бути як фізична, так і юридична особа (фірма, компанія тощо). Провайдер надає доступ до сервісів, перенаправляючи інтернет-трафік клієнтів через власні сервери. Основною метою роботи провайдерів є надання стабільного доступу до сервісів та забезпечення захисту й цілісності даних. У сучасному світі фірми Інтернет-провайдера представлені IT-проєктами, процеси яких повністю автоматизовані.

Найчастіше користувачі систем Інтернет-провайдерів не задоволені тим, що вони не мають можливості гнучкого самостійного налаштування послуг тому, що їх налаштування вже визначені. Послуги повинні бути різноманітними та підлаштовуватися під змінення ринку. Багато систем не готові до таких змін, вони потребують рефакторингу. Головним обмеженням переходу з старого підходу на новий є спосіб зберігання даних, а саме структура-схема чи тип бази даних (БД).

Рефакторинг БД – це зміна схеми БД з метою покращення дизайну БД і збереження як інформації, так і семантики поведінки. Зокрема покращення дизайну БД має на меті створення умов до легкого розширення функціоналу IT-проєкту. Це процес реструктуризації існуючої схеми БД з метою покращення її дизайну, продуктивності або зручності обслуговування, зберігаючи її функціональність і цілісність даних. Рефакторинг є усталеною концепцією в розробці програмного забезпечення (ПЗ), і принципи, якими керується рефакторинг ПЗ, можна також застосувати до рефакторингу БД.

Мета рефакторингу БД – зробити схему БД більш гнучкою, масштабованою та придатною для обслуговування з часом, мінімізуючи ризик появи помилок або втрати даних. Рефакторинг може включати ряд дій, від

невеликих змін до окремих таблиць або стовпців до більш масштабних змін у загальній архітектурі схеми.

Рефакторинг БД буде корисним Інтернет-провайдеру тим, що дозволить покращити гнучкість надання послуг без припинення повсякденної діяльності. У свою чергу, перехід до нового підходу взаємодії з тарифними планами та пакетами дозволить не тільки підвищити конкурентоспроможність, а й зайняти нову нішу на ринку першим, залучити нових клієнтів та розширити спектр послуг, що надаються.

Після проведення рефакторингу у подальшому розвитку проєкту очікується підвищення ремонтопридатності БД, тобто рефакторинг допоможе зробити схему даних більш гнучкою, модульною та зручною у супроводі з плином часу. Це спростить додавання нових функцій, виправлення помилок і адаптацію до бізнес-вимог, що змінюються, без внесення помилок або порушення нормальної роботи.

Таким чином, інвестиції в проєкт рефакторингу БД можуть принести низку переваг, які можуть покращити результати бізнесу та знизити ризики.

Об’єктом дослідження в рамках магістерської кваліфікаційної роботи є процес рефакторингу БД інформаційно-облікової системи фірми Інтернет-провайдера.

Предметом дослідження являються методи аналізу та вирішення задачі рефакторингу БД Інтернет-провайдера.

Метою даної роботи є дослідження моделей і методів управління ІТ-проєктом рефакторингу БД Інтернет-провайдера задля підтримки гнучкого створення замовлень.

Магістерська кваліфікаційної робота виконана згідно з методичними вказівками 2019 року щодо розробки та оформлення магістерської кваліфікаційної роботи за спеціальністю 122 Комп’ютерні науки та освітньою програмою «Управління проектами в галузі інформаційних технологій» [1].

1 АНАЛІЗ ІСНУЮЧИХ МОДЕЛЕЙ І МЕТОДІВ УПРАВЛІННЯ ІТ-ПРОЄКТОМ РЕФАКТОРИНГУ БАЗИ ДАНИХ ІНТЕРНЕТ-ПРОВАЙДЕРА

1.1 Аналіз актуальності дослідження моделей і методів управління ІТ-проєктом рефакторингу бази даних Інтернет-провайдера

Провайдер – організація, яка надає послуги доступу до глобальної мережі Інтернет. Клієнтом провайдера може бути як фізична, так і юридична особа (фірма, компанія тощо). Провайдер надає доступ до сервісів, перенаправляючи інтернет-трафік клієнтів через власні сервери. Основною метою роботи провайдерів є надання стабільного доступу до сервісів та забезпечення захисту й цілісності даних. У сучасному світі фірми Інтернет-провайдера представлені ІТ-проєктами, процеси яких повністю автоматизовані [2].

Найчастіше користувачі систем Інтернет-провайдерів не задоволені тим, що вони не мають можливості гнучкого самостійного налаштування послуг тому, що їх налаштування вже визначені. Послуги повинні бути різноманітними та підлаштовуватися під змінення ринку. Багато систем не готові до таких змін, вони потребують рефакторингу. Головним обмеженням переходу з старого підходу на новий є спосіб зберігання даних, а саме структура-схема чи тип БД [2].

Рефакторинг БД – це зміна схеми БД з метою покращення дизайну БД і збереження як інформації, так і семантики поведінки. Зокрема покращення дизайну БД має на меті створення умов до легкого розширення функціоналу ІТ-проєкту. Це процес реструктуризації існуючої схеми БД з метою покращення її дизайну, продуктивності або зручності обслуговування, зберігаючи її функціональність і цілісність даних. Рефакторинг є усталеною концепцією в розробці ПЗ, і принципи, якими керується рефакторинг ПЗ, можна також застосувати до рефакторингу БД [2].

Мета рефакторингу БД – зробити схему БД більш гнучкою, масштабованою та придатною для обслуговування з часом, мінімізуючи ризик появи помилок або втрати даних. Рефакторинг може включати ряд дій, від невеликих змін до окремих таблиць або стовпців до більш масштабних змін у загальній архітектурі схеми [3].

Деякі типові приклади рефакторингу БД включають [3]:

– видалення зайвих або невикористаних таблиць, стовпців або індексів;

– розбиття великих таблиць на менші для підвищення продуктивності або спрощення запитів;

– об’єднання кількох таблиць в одну для зменшення складності;

– перейменування таблиць або стовпців для покращення чіткості чи послідовності;

– зміна типів даних або обмежень для кращого відображення моделі даних або бізнес-правил;

– додавання або зміна індексів для оптимізації продуктивності запитів;

– нормалізація або денормалізація схеми для покращення цілісності даних або зменшення надмірності.

Рефакторинг БД може бути складним і трудомістким процесом, особливо у великих або критично важливих системах. Важливо ретельно планувати та виконувати рефакторинг, а також ретельно тестувати отримані зміни, щоб переконатися, що вони не створюють нових проблем і не порушують існуючі функції.

Деякі найкращі методи успішного рефакторингу БД включають [3]:

– створення резервної копії або знімка БД перед внесенням будь-яких змін;

– детальне документування поточної схеми та будь-яких запропонованих змін;

– тестування змін у невиробничому середовищі перед застосуванням їх до живої системи;

– зведення до мінімуму обсягу та частоти змін, щоб уникнути порушення нормальної роботи;

– співпраця із зацікавленими сторонами, розробниками та адміністраторами БД, щоб переконатися, що рефакторинг відповідає бізнес-цілям і технічним вимогам.

Ефективність рефакторингу БД буде вимірюватися в мінімізації змін та у виборі найкращого шляху реструктуризації БД з метою підтримки розширення функцій системи. Процес рефакторингу буде представлений у вигляді алгоритму. Алгоритм буде аналізувати дані, які потрібні для розширення функціоналу, та на основі цього аналізу буде чи видавати інформацію про використання існуючих таблиць та їх атрибутів, чи запропоновувати шлях реструктуризації БД, враховуючи можливість зміни типу БД на іншу найбільш підходящу.

Рефакторинг БД буде корисним Інтернет-провайдеру тим, що дозволить покращити гнучкість надання послуг без припинення повсякденної діяльності. У свою чергу, перехід до нового підходу взаємодії з тарифними планами та пакетами дозволить не тільки підвищити конкурентоспроможність, а й зайняти нову нішу на ринку першим, залучити нових клієнтів та розширити спектр послуг, що надаються.

Після проведення рефакторингу у подальшому розвитку проєкту очікується підвищення ремонтопридатності БД, тобто рефакторинг допоможе зробити схему даних більш гнучкою, модульною та зручною у супроводі з плином часу. Це спростить додавання нових функцій, виправлення помилок і адаптацію до бізнес-вимог, що змінюються, без внесення помилок або порушення нормальної роботи.

Таким чином, інвестиції в проєкт рефакторингу БД можуть принести низку переваг, які можуть покращити результати бізнесу та знизити ризики.

1.2 Аналіз відмінностей та особливостей управління ІТ-проєктом рефакторингу бази даних від інших типів ІТ-проєктів

Проєкти рефакторингу БД відрізняються від інших типів проєктів кількома характеристиками [4]:

– фокус: основна мета проєкту рефакторингу БД полягає в покращенні дизайну, продуктивності та зручності обслуговування існуючої схеми БД, зберігаючи при цьому функціональність і цілісність даних. З іншого боку, інші проєкти можуть мати інші цілі, такі як розробка нових функцій, виправлення помилок або розгортання нових програм;

– складність: проєкти рефакторингу БД можуть бути складнішими за інші проєкти, оскільки вони передбачають внесення змін до існуючої системи, яка вже розгорнута та може використовуватися багатьма користувачами. Це вимагає ретельного планування, тестування та координації, щоб уникнути переривання нормальної роботи або появи нових проблем;

– співпраця: проєкти рефакторингу БД вимагають тісної співпраці між зацікавленими сторонами, розробниками та адміністраторами БД, щоб забезпечити відповідність змін бізнес-цілям, технічним вимогам і цілісності даних. Інші проєкти також можуть вимагати співпраці, але рівень координації та спілкування, необхідний для проєктів рефакторингу БД, може бути вищим;

– ризики: проєкти рефакторингу БД можуть мати більший ризик, ніж інші проєкти, через можливість втрати даних або помилок. Важливо мати стратегії резервного копіювання та відновлення, ретельно перевіряти зміни перед розгортанням їх у системі, що працює;

– тривалість: виконання проєктів рефакторингу БД може тривати довше, ніж інші проєкти, через складність змін і необхідність ретельного планування та тестування кожного кроку. Це може вимагати довгострокових зобов’язань від зацікавлених сторін і може потребувати додаткових ресурсів для успішного виконання [5].

Незважаючи на згадані вище відмінності в управлінні проєкту рефакторингу з іншими проєктами, вони мають схожості, а саме однакові стадії життєвого циклу (ЖЦ) з точки зору управління. Відповідно до стандарту ISO/IEC/IEEE 15288:2015 всі системи мають однакові стадії ЖЦ, вони зображені на рис. 1.1 [6].

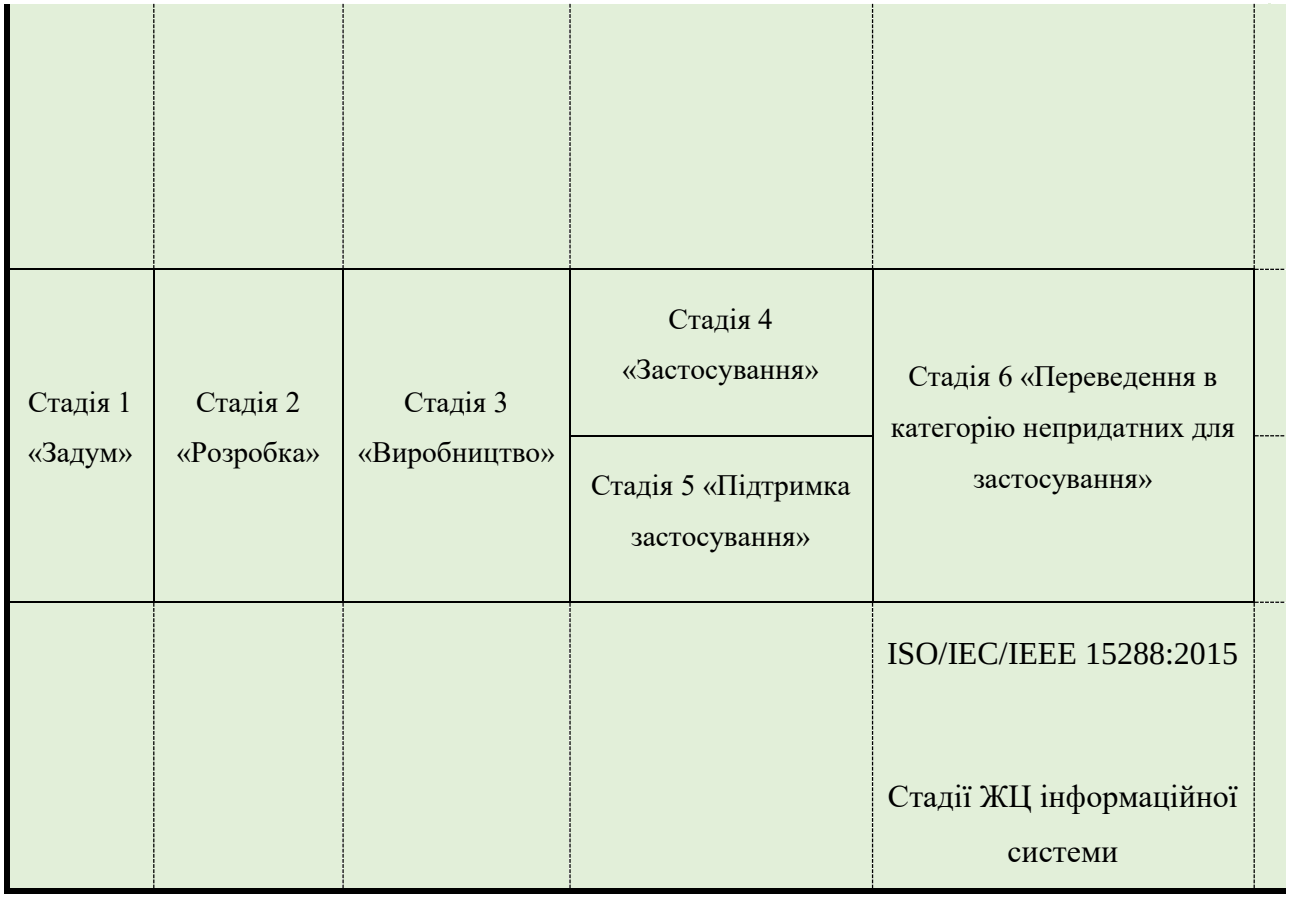

Рисунок 1.1 – Життєвий цикл системи за стандартом ISO/IEC 15288:2002

Проєкт рефакторингу відрізняється від інших проєктів саме тим, що він прив’язаний до стадій «Застосування» та «Підтримка застосування» ЖЦ існуючої системи, а система прив’язана до усіх стадій ЖЦ.

Це виражається тим, що проєкт рефакторингу проводиться в рамках покращення та оптимізації ресурсів експлуатованої інформаційної системи (ІС).

Виходячи з вище перерахованих відмінностей та однакових-подібних рис, хоча проєкти рефакторингу БД мають невелику схожість з точки зору

стадій ЖЦ за стандартом ISO/IEC/IEEE 15288:2015 з іншими проєктами, вони вимагають додаткового планування, тестування та координації, щоб переконатися, що зміни вносяться безпечно та ефективно, зберігаючи цілісність даних і зводячи до мінімуму порушення нормальної роботи.

Було розглянуто верхній рівень ІС з виділенням відмінностей та подібностей проєкту рефакторингу з іншими проєктами. Для пошуку наступних подібностей та відмінностей буде розглядатися більш низький рівень, а саме ЖЦ проєктів по методології PRINCE2 [7].

PRINCE2 (PRojects IN Controlled Enviroments) – це методологія проектного менеджменту, у центрі впливу якого є управлінські аспекти проєктів.

Ця структура управління проєктом є лінійною та базується на процесах, зосереджена на просуванні ініціатив через заздалегідь визначені етапи. PRINCE2 також містить основні принципи управління проєктами, як-от окреслення обсягу та бюджету вашого проєкту, що робить його хорошим варіантом для новачків [7].

Особливості PRINCE2 [8]:

– планування, що ґрунтується на продуктовому підході;
– поділ проєкту на керовані та контрольовані стадії;
– гнучкість стосовно масштабів проєкту;
– певна організована стадія команди управління проєктом.

За допомогою методології PRINCE2 можна провести підрозділ ЖЦ системи на етапи, з її допомогою можна також виділити схожості та відмінності управління проєктом рефакторингу та інших проєктів. Етапи ЖЦ систем методології PRINCE2 подано на рис. 1.2 [8].

Рисунок 1.2 – Життєвий цикл IT-проєкту за методологією PRINCE2

Етапи ЖЦ проєкту рефакторингу є ітераціями ЖЦ іншого проєкту. Рефакторинг систем майже завжди відбувається на п’ятому етапі ЖЦ. Проєкт рефакторингу проходить всі етапи ЖЦ, але є другою, третьою або N-ою ітерацією ЖЦ існуючого IT-проєкту в залежності від того, який раз він проводиться.

З точки зору IT-проєкту планування – це вирішення завдань, пов’язаних із предметною областю, бізнесом та діяльністю Інтернет-провайдера (зберігання та обробка замовлень, взаємозв’язок із клієнтом, ведення обліку послуг тощо). А з погляду проєкту рефакторингу, планування – це знаходження проблем, пов’язаних із системою зберігання даних, схемою БД тощо.

Проектування IT-проєкту обмежено зовнішнім середовищем (все, що впливає на IT-проєкт, але не є його складовою, наприклад, поточні умови ринку та інше), існуючими процесами фірми Інтернет-провайдера. Всі внутрішні та зовнішні обмеження батьківського IT-проєкту будуть враховані в задачі управління проєктом рефакторингу, але в якості головного обмеження

проєкту рефакторингу слід розглядати існуючий ІТ-проєкт. Тобто існуючий ІТ-проєкт, «батьківський» – це той, для якого здійснюється поліпшення, а його важливість обумовлена тим, що рефакторинг нерозривно пов’язаний з об’єктом поліпшення, і бажаним результатом.

З погляду ІТ-проєкту, під стадією «Використання» мають на увазі експлуатацію фінальної версії кінцевими користувачами. А з погляду проєкту рефакторингу БД Інтернет-провайдера, під стадією «Використання» мається на увазі введення у в експлуатацію командою розробників результатів проведеного рефакторингу до існуючої системи.

Стадії «Експлуатація» та «Підтримка» у рамках проєкту рефакторингу БД Інтернет-провайдера представляють собою ітераційні (N-і) фази батьківського проєкту та повинні розглядатися з точки зору його ЖЦ.

Проаналізувавши стадії ЖЦ ІТ-проєкту та проєкту рефакторингу БД Інтернет-провайдера за методологією PRINCE2, було визначено, що проєкт рефакторингу має відмінності на кожній стадії ЖЦ порівняно з ІТ-проєктом для якого застосовується рефакторинг [8].

Підводячи підсумок аналізу стадій ЖЦ систем за стандартом ISO/IEC/IEEE 15288:2015 та стадій ЖЦ ІТ-проєкту за методологією PRINCE2 можна виділити наступні відмінності:

– проєкт рефакторингу прив’язаний до стадій «Застосування» та «Підтримка застосування» ЖЦ існуючої системи, а система прив’язана до усіх стадій ЖЦ системи за стандартом ISO/IEC/IEEE 15288:2015;

– за методологією PRINCE2 проєкт рефакторингу має відмінності на кожній стадії ЖЦ з порівнянням інших ІТ-проєктів.

Так як проєкт рефакторингу проходить на стадіях ЖЦ «Застосування» та «Підтримка застосування», що проходять паралельно, можна виділити тісний зв’язок між стадіями ЖЦ системи та проєктом рефакторингу, тим самим на час розробки та введення в експлуатацію проєкту рефакторингу існуюча система не повинна припиняти свою дію.

1.3 Аналіз існуючих моделей і методів управління ІТ-проєктом рефакторингу бази даних Інтернет-провайдера

Існує декілька моделей та методів управління ІТ-проєктом рефакторингу БД.

Перша з них це методика рефакторингу БД (Database Refactoring Methodology) розроблена Скоттом Амблером та його колегами. Вона являє собою набір процедур та практик, які допомагають безпечно та ефективно проводити рефакторинг БД у рамках проєкту розробки ПЗ.

Вона може бути використана для різних цілей, наприклад, для оптимізації продуктивності БД, спрощення їх структури, поліпшення їх масштабованості тощо. Також вона дозволяє зменшувати витрати на їх розвиток та підтримку, покращувати якість та надійність БД, а також підвищувати зручність роботи з ними для кінцевих користувачів.

Методика рефакторингу БД полягає в тих принципах, як і методика рефакторингу коду. Вона пропонує підхід до рефакторингу, який дозволяє поступово покращувати якість БД та знижувати ризики, пов’язані зі зміною БД [9].

Методика рефакторингу БД складається з наступних основних етапів [9]:

– аналіз БД: у межах цього етапу визначаються слабкі місця у БД, які потребують рефакторингу. Аналіз може проводитись як вручну, так і за допомогою спеціальних інструментів;

– планування рефакторингу: на цьому етапі визначається послідовність та тривалість рефакторингу, а також оцінюються ризики та витрати на рефакторинг;

– рефакторинг БД: на цьому етапі відбуваються зміни у структурі БД з метою покращення її якості. Рефакторинг може включати зміну схеми БД, перейменування об’єктів, зміна типів даних і т. д.;

– тестування та впровадження: на цьому етапі перевіряється працездатність БД після рефакторингу, а також вносяться зміни до програм, які використовують БД;

– документування змін: у цьому процесі документуються всі зміни, внесені до БД, щоб забезпечити подальшу підтримку та розвиток.

Методика рефакторингу БД надає докладний набір процедур та інструментів, необхідних проведення рефакторингу БД у рамках проєкту розробки ПЗ. Вона дозволяє проводити рефакторинг без ризику втрати даних та забезпечує високий рівень захисту цілісності даних у процесі зміни БД.

Вона пропонує використовувати інструменти автоматизації, які можуть прискорити процес рефакторингу і зменшити ризики, пов’язані зі зміною БД.

Переваги використання методики рефакторингу БД включають [9]:

– поліпшення якості БД;

– зниження ризиків, пов’язаних із зміною БД;

– прискорення процесу розробки;

– збільшення гнучкості системи;

– поліпшення розуміння структури БД.

Одним із недоліків методики рефакторингу БД є те, що вона може бути надто трудомісткою та складною для невеликих проєктів або невеликих змін у БД. Це пов’язано з тим, що методика включає безліч кроків, починаючи з аналізу БД і закінчуючи тестуванням і впровадженням змін, які можуть вимагати значного часу і зусиль.

Ще один недолік полягає в тому, що методика не завжди може бути легко застосована у випадках, коли програма має високий ступінь залежності від БД. У таких випадках зміна БД може вимагати внесення змін до додатка, що може спричинити значні витрати на розробку та тестування.

Крім того, методика рефакторингу БД може вимагати використання спеціалізованих інструментів, таких як інструменти для автоматичного тестування БД або інструменти міграції даних. Це може бути додатковим витратним фактором.

Нарешті, недоліком методики може бути те, що вона може бути використана тільки для поліпшення якості БД. Якщо потрібно змінити бізнес-логіку або функціональність програми, то може знадобитися використання інших методів та підходів до розробки ПЗ.

Модель управління якістю даних (Data Quality Management Model) – це модель, розроблена The Data Warehousing Institute (TDWI). Вона надає докладний план дій для поліпшення якості даних і включає процеси, пов’язані з рефакторингом БД [10].

Модель управління якістю даних визначає стандарти та методології, необхідні для оцінки та управління якістю даних в організації. Вона складається з наступних етапів, що описують процес управління якістю даних від початку до кінця [10]:

– визначення вимог щодо якості даних – визначення критеріїв якості даних та створення механізму для оцінки відповідності даних цим критеріям;

– аналіз якості даних – оцінка якості даних з використанням певних критеріїв якості та визначення причин помилок у даних;

– очищення даних – видалення або коригування некоректних чи неповних даних з метою покращення якості даних;

– управління якістю даних – встановлення процедур та механізмів для безперервного контролю якості даних та забезпечення їх відповідності встановленим критеріям якості;

– моніторинг якості даних – оцінка та контроль якості даних для забезпечення їх відповідності правилам та стандартам.

Ця модель допомагає забезпечувати точність, повноту та достовірність даних ІТ-проєкту, що дозволяє приймати більш обґрунтовані рішення та підвищувати ефективність бізнес-процесів ІТ-проєкту.

До переваг моделі управління якістю даних можна віднести наступне [10]:

– поліпшення розуміння структури БД;

– поліпшення якості БД;

– зниження ризиків, пов’язаних зі зміною БД;

– збільшення гнучкості системи для введення нового функціоналу.

Методика зміни БД (Database Change Management) – це методика, яка описує процеси та практики, необхідні для управління змінами у БД. У межах цієї методики рефакторинг БД являється одним із процесів, необхідних для ефективної зміни БД [11].

Основними перевагами методики зміни БД є підвищення ефективності управління змінами, зменшення ризику виникнення помилок та спрощення процесу розгортання змін у виробничому середовищі [11].

Одним із інструментів, який може бути використаний для реалізації методики зміни БД є система контролю версій, така як Git або SVN. Вони дозволяють контролювати зміни в коді БД, відстежувати їх і легко повертатися до попередніх версій за потреби. Виходячи з наведено аналізу можна виділити основні переваги кожної моделі та методу:

– методика рефакторингу БД Скотта Амблера являє собою набір процедур та практик, що допомагають ефективно та безпечно проводити рефакторинг БД у рамках проєкту розробки ПЗ;

– модель управління якістю даних надає докладний план дій для поліпшення якості даних;

– методика зміни БД описує процеси та практики, необхідні для управління змінами у БД.

Підводячи підсумок, перелічені методи та моделі можуть використовуватися у поєднанні один з одним, щоб створити найбільш ефективну модель управління проєктом рефакторингу БД. Нова модель буде розпланована по етапах, матиме план поліпшення якості даних та за допомогою процесів та практик проведе зміну БД у відмовостійкій та надійній манері.

1.4 Аналіз існуючих рішень рефакторингу бази даних

Існує декілька програмних рішень рефакторингу БД, які можуть допомогти автоматизувати та спростити процес рефакторингу схеми БД.

Redgate SQL Refactor – це комерційний інструмент, який інтегрується з Microsoft SQL Server Management Studio, щоб допомогти автоматизувати та спростити процес рефакторингу коду SQL. Він надає низку функцій, таких, як перейменування об’єктів, форматування коду, генерування сценаріїв SQL, переміщення стовпців, тощо [12].

Одна з найпростіших і найчастіше використовуваних функцій рефакторингу БД – це функція перейменування. Зміна назви об’єкта коду, таблиці чи стовпця може бути трудомістким і навіть складним завданням через усі залежності, які можуть існувати у БД Інтернет-провайдера.

У всьому коді та обмеженнях розробники повинні бути впевнені, що знають про всі можливі побічні ефекти однієї, здавалося б, простої зміни імені. Внесення цих змін вручну може зайняти від декількох хвилин до декількох годин.

Для пошуку та виявлення усіх залежностей можна скористатися побудовою діаграми залежностей об’єктів. Інструмент SQL Dependency Tracker від Redgate інтегрується з середою розробки та надає детальну діаграму залежностей для будь-якого вибраного об’єкта.

Наприклад, у SSMS (SQL Server Management Studio) Object Explorer можна відкрити за допомогою натискання правою кнопкою миші по об’єкту у БД та виберу «Переглянути діаграму залежностей для…» [12].

Діаграма залежностей для об’єкта відображає сам об’єкт та інші об’єкти, які мають з ним зв’язок. Діаграма залежностей зображена на рис. 1.3.

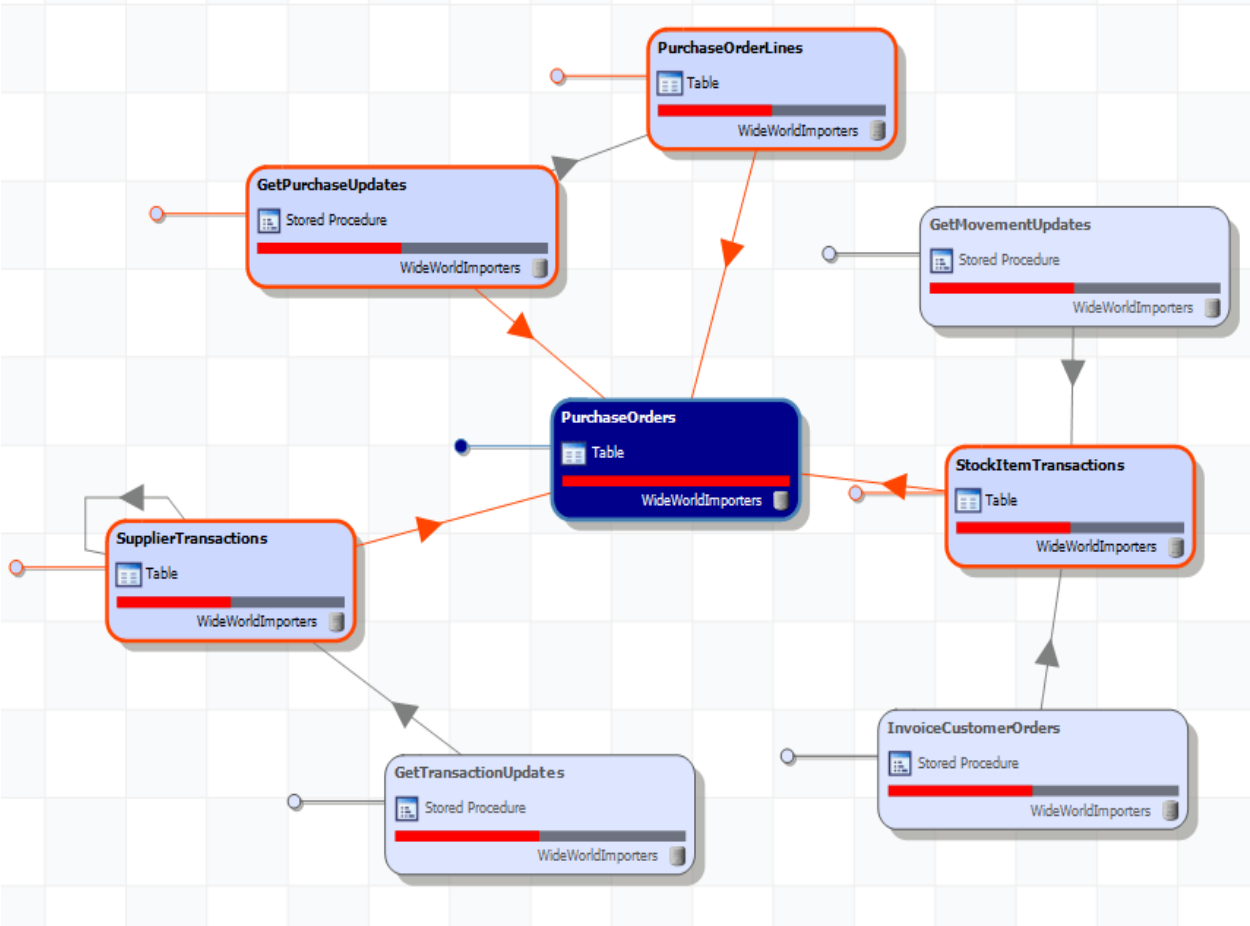

Рисунок 1.3 – Діаграма залежностей

З цієї діаграми видно масштабність завдання, яке стоїть перед розробником, якщо йому потрібно буде змінити ім’я вручну. На щастя, він може використовувати функцію Smart Rename SQL Prompt, яка автоматично змінює майже всі посилання на перейменований об’єкт у поточній БД. Динамічні посилання SQL не оброблятимуться, тому ця функція не скасовує потребу в надійних планах тестування [12].

Приклад використання функції розумного перейменування зображено на рис. 1.4.

Також інструмент Redgate SQL Refactor має набір інших переваг, а саме [12]:

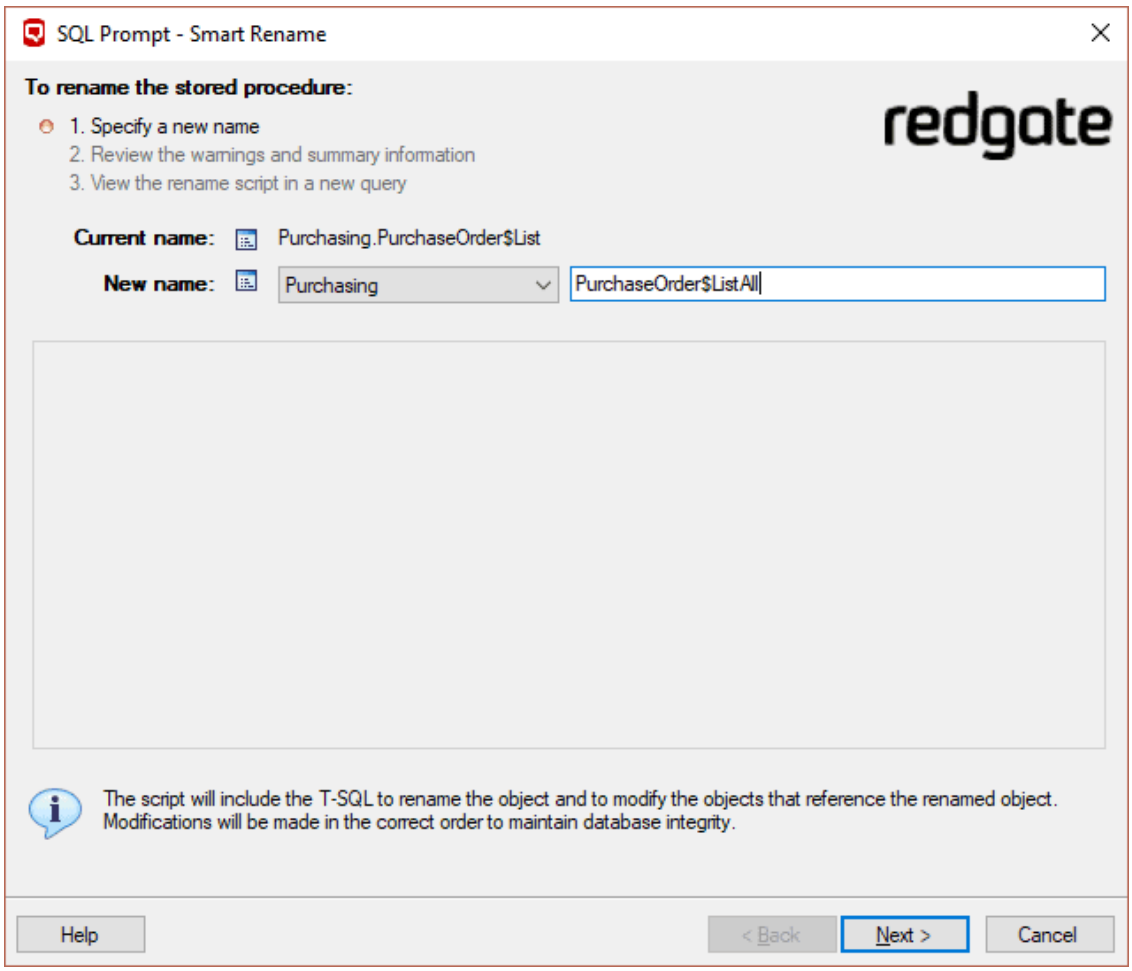

Рисунок 1.4 – Функція «Розумне перейменування»

– покращення читабельності: SQL Refactor автоматично форматує SQL-код, щоб зробити його більш читабельним. Це допомагає спростити налагодження та полегшити супровід коду;

– спрощення написання коду: SQL Refactor надає більше 200 правил перейменування, перенесення, об’єднання та інших змін коду, що дозволяє швидко та легко оптимізувати SQL-запити та зробити їх ефективнішими;

– підтримка багатьох редакторів: SQL Refactor підтримує безліч редакторів, включаючи SQL Server Management Studio, Visual Studio, Eclipse, IntelliJ IDEA та інші;

– швидкість роботи: SQL Refactor працює дуже швидко, що дозволяє значно скоротити час на розробку та оптимізацію SQL-коду;

– поліпшення безпеки: SQL Refactor надає можливість видаляти конфіденційну інформацію з SQL-коду, таку як імена користувачів та паролі, що допомагає покращити безпеку БД;

– поліпшення продуктивності: SQL Refactor надає можливість автоматично переписувати SQL-запити, щоб зробити їх більш оптимальними та покращити продуктивність БД;

– спрощення командної роботи: SQL Refactor спрощує спільну роботу над SQL-кодом, дозволяючи команді розробників використовувати єдині правила написання коду та автоматично перевіряти його відповідність цим правилам;

– підтримка різних БД: SQL Refactor підтримує різні типи БД, включаючи SQL Server, Oracle, MySQL, PostgreSQL та інші.

Наступний інструмент, який розглядається це додаток DBmaestro. DBmaestro – це комерційний інструмент автоматизації випуску БД, який допомагає автоматизувати та керувати змінами схеми БД у багатьох середовищах. Він підтримує ряд БД, включаючи Oracle, SQL Server і MySQL (рисунок 1.5) [13].

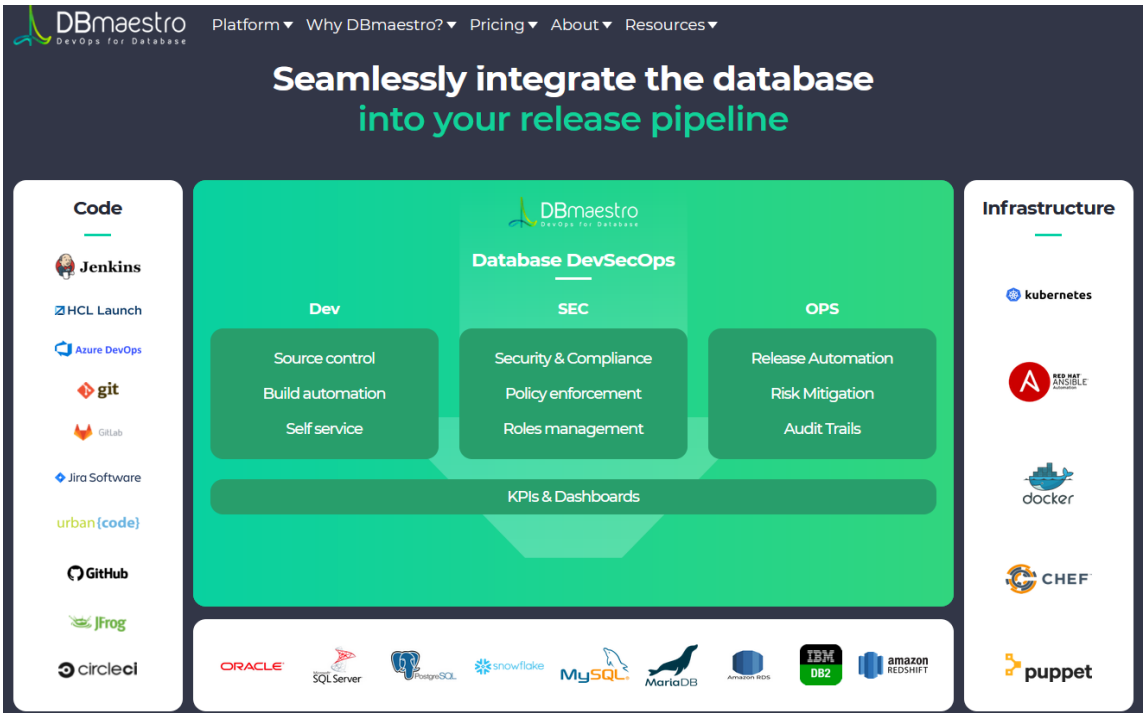

Рисунок 1.5 – Перелік сервісів з якими інтегрується додаток DBmaestro

Додаток DBmaestro має наступні переваги [13]:

– централізоване управління: DBmaestro надає централізоване управління для змін у БД, дозволяючи команді розробників та адміністраторів працювати з одним і тим самим джерелом правди;

– поліпшення якості: DBmaestro допомагає покращити якість БД, забезпечуючи контроль версій та автоматичну перевірку змін, а також забезпечуючи відповідність БД стандартам безпеки;

– прискорення процесу розгортання: DBmaestro автоматизує процес розгортання БД, що дозволяє скоротити час на цей процес та зменшити ризик помилок при розгортанні;

– управління релізами: DBmaestro забезпечує управління релізами, що дозволяє командам розробників та адміністраторів контролювати та керувати змінами, пов’язаними з кожним релізом;

– легкість використання: DBmaestro надає простий і інтуїтивно зрозумілий інтерфейс користувача, що дозволяє швидко і легко використовувати це ПЗ;

– інтеграція з іншими інструментами: DBmaestro інтегрується з іншими інструментами, що використовуються в процесі розробки та управління БД, включаючи Microsoft Visual Studio, Jenkins, Git та інші;

– масштабованість: DBmaestro масштабується для підтримки як малих, так і великих команд розробників та адміністраторів БД.

Ці переваги роблять DBmaestro корисним інструментом для управління змінами в БД та автоматизації процесу розгортання БД, дозволяючи командам розробників та адміністраторів прискорити та покращити процес розробки та управління БД (рисунок 1.6).

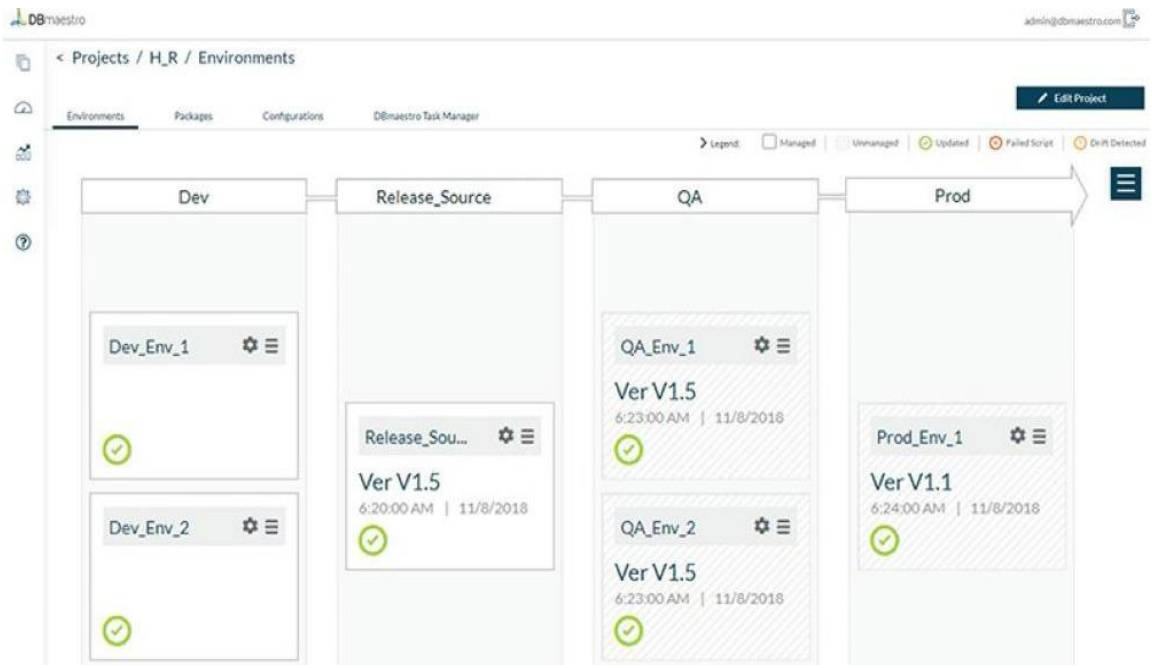

Рисунок 1.6 – Приклад екрану використання додатку DBmaestro

Останнім інструментом, що розглядається є додаток ApexSQL Refactor. ApexSQL Refactor – це комерційний інструмент, який інтегрується з Microsoft SQL Server Management Studio, щоб допомогти автоматизувати та спростити процес рефакторингу коду SQL. Він надає низку функцій, таких як перейменування об’єктів, форматування коду, аналіз залежностей та рефакторингу SQL коду. Він дозволяє швидко та легко переписати складний SQL код у більш читаний та зручний формат, покращуючи ефективність та продуктивність роботи з БД (рисунок 1.7) [14].

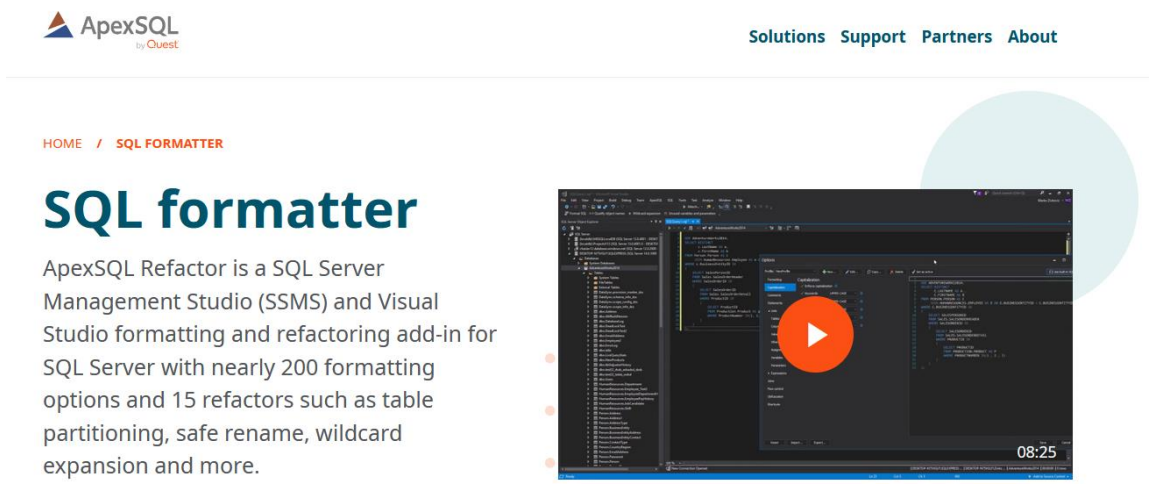

Рисунок 1.7 – Сайт ApexSQL з прикладом головного екрану програми

Нижче наведено переваги додатку ApexSQL Refactor [14]:

– підвищення читабельності SQL коду: надає функції форматування SQL коду, що робить його більш зрозумілим та читабельним для розробників;

– підвищення продуктивності: ApexSQL Refactor надає функції рефакторингу SQL коду, що дозволяє оптимізувати код для підвищення продуктивності та прискорення роботи з БД;

– автоматичне форматування: ApexSQL Refactor надає можливість автоматичного форматування SQL-коду за допомогою стандартних шаблонів форматування, що дозволяє скоротити час на ручне форматування коду;

– підтримка різних БД: ApexSQL Refactor підтримує роботу з різними БД, включаючи MS SQL Server, Oracle, MySQL та інші;

– інтеграція з іншими інструментами: інтегрується з іншими інструментами, які використовуються в процесі розробки та керування БД, включаючи MS Visual Studio та SQL Server Management Studio;

– легкість використання: ApexSQL Refactor надає простий і інтуїтивно зрозумілий інтерфейс користувача, що дозволяє швидко і легко використовувати це ПЗ;

– масштабованість: ApexSQL Refactor масштабується для підтримки як малих, так і великих команд розробників та адміністраторів БД.

Перелічені переваги роблять його корисним інструментом для форматування та рефакторингу SQL коду, дозволяючи розробникам та адміністраторам БД підвищити ефективність та продуктивність роботи з БД.

Отже, кожен з перелічених інструментів може допомогти оптимізувати й автоматизувати процес рефакторингу існуючої схеми БД, але важливо ретельно оцінити кожне рішення, щоб переконатися, що воно відповідає вашим конкретним потребам і вимогам. Також важливо чітко розуміти принципи рефакторингу БД і найкращі практики, а також чіткий план тестування та розгортання змін у живій системі.

Було проведено порівняльний аналіз існуючих рішень рефакторингу БД, результати наведено у вигляді таблиці 1.1.

Таблиця 1.1 – Порівняння існуючих рішень рефакторингу БД

Критерії порівняння рішень	Redgate SQL Refactor	DBmaestro	ApexSQL Refactor
функціональність	автоматичне форматування коду, перейменування об’єктів, усунення дублікатів, розширена перевірка синтаксису та інші	управління змінами в БД, керування версіями, автоматизовані сценарії для розгортання, моніторинг змін, керування збірками тощо	автоматичне форматування коду, перейменування об’єктів БД, забезпечення сумісності з різними стандартами написання SQL-коду
тип рішення	розширення	веб-застосунок	розширення
інтерфейс	простий та інтуїтивно зрозумілий	простий та інтуїтивно зрозумілий	простий та інтуїтивно зрозумілий
СУБД, що підтримуються	SQL Server, Azure SQL Database, Amazon RDS, PostgreSQL, Oracle	SQL Server, Oracle, MySQL, PostgreSQL	SQL Server, Azure SQL Database, Amazon RDS, PostgreSQL, Oracle
робота з командним рядком	не підтримується	підтримується	не підтримується
автоматизація завдань	вузька спрямованість на автоматичне форматування коду та перейменування об’єктів	широкий набір можливостей для автоматизації завдань	вузька спрямованість на автоматичне форматування коду та перейменування об’єктів
функції та можливості	автоматичне форматування коду, перейменування об’єктів, розширена перевірка синтаксису, усунення дублікатів	контроль версій, відстеження змін, створення скриптів, керування збірками тощо.	автоматичне форматування коду, перейменування об’єктів БД, створення скриптів

Отже, можна виділити, що всі три програми мають простий та інтуїтивно зрозумілий інтерфейс, а також підтримку роботи з декількома типами БД. Redgate SQL Refactor і ApexSQL Refactor забезпечують автоматичне форматування коду.

DBmaestro надає ширший набір функцій та можливостей, таких як керування версіями, моніторинг змін та керування збірками, в той час як Redgate SQL Refactor та ApexSQL Refactor зосереджені на автоматичному форматуванні коду та перейменуванні об’єктів БД. DBmaestro також підтримує роботу з командним рядком, що робить його більш гнучким для автоматизації завдань.

1.5 Обґрунтування мети створення методу та моделі для визначення та проведення рефакторингу для підвищення гнучкості бази даних Інтернет-провайдера

Результати аналізу сучасного стану проблеми гнучкого створення замовлень в ІТ-проєкті фірми Інтернет-провайдера дозволяють зробити наступні висновки.

По-перше, проведення рефакторингу у подальшому розвитку проєкту буде сприяти підвищенню ремонтопридатності БД та спрощенню її структури, тобто рефакторинг допоможе зробити схему даних більш гнучкою, модульною та зручною у супроводі з плином часу. Це спростить додавання нових функцій, виправлення помилок і адаптацію до бізнес-вимог, що змінюються, без внесення помилок або порушення нормальної роботи. Таким чином, інвестиції в проєкт рефакторингу БД принесуть низку переваг, які покращать результати бізнесу та знизять ризики у подальшому розвитку ІТ-проєкту.

По-друге, виходячи з аналізу існуючих моделей і методів було визначено, що для проведення рефакторингу розглянуті рішення треба використовувати у поєднанні один з одним. Це надасть змогу створити найбільш ефективну модель управління проєктом рефакторингу БД і рішення, яке б допомогло розпланувати етапи проведення рефакторингу немає. Новий метод дозволить провести аналіз існуючого стану, на основі якого будуть виявленні кроки для рефакторингу, які будуть основою для планування по етапах. Крім того, цей метод матиме план поліпшення якості даних та за допомогою процесів та практик проведе зміну БД у відмовостійкій та надійній манері.

По-третє, аналогічно з існуючими методами проведення рефакторингу не існує рішення, яке б покривало усі необхідні кейси. І найкраще рішення – це використовувати ці додатки у поєднанні. Після проведення аналізу існуючих рішень рефакторингу БД можна виділити, що вони допомагають тільки у процесі рефакторингу, а саме: вони являють собою допоміжні інструменти для його проведення. Їм бракує одного – відсутність виявлення кроків, які треба зробити, щоб з’ясувати чи потрібен рефакторинг, якщо так, то отримати з поточного стану бажане та як це перевірити.

1.6 Висновки за результатами проведеного аналізу та постановка задачі дослідження моделей і методів управління ІТ-проєктом рефакторингу бази даних Інтернет-провайдера

Об’єктом дослідження в рамках магістерської кваліфікаційної роботи є процес рефакторингу БД Інтернет-провайдера, що представляє собою процес управління проєктом інтегрований контроль змін (4.6 Perform Integrated Change Control), згідно з Project Management Body of Knowledge’s (PMBOK) [15].

Предметом дослідження являються методи аналізу та вирішення задачі рефакторингу БД Інтернет-провайдера.

Метою даної роботи є дослідження моделей і методів управління ІТ-проєктом рефакторингу БД Інтернет-провайдера задля підтримки гнучкого створення замовлень.

Для досягнення мети, необхідно досліджувати наступні питання:

– проаналізувати відмінності та особливості управління ІТ-проєктом рефакторингу від інших типів проєктів;

– проаналізувати існуючі моделі і методи управління ІТ-проєктом рефакторингу БД Інтернет-провайдера;

– проаналізувати існуючі рішення рефакторингу БД;

– сформувати вимоги до виконання роботи з оцінювання доцільності проведення рефакторингу БД;

– розробити модель оцінювання складності проведення рефакторингу БД фірми Інтернет-провайдера;

– вдосконалити модель оцінювання трудовитрат та витрат часу на виконання ІТ-проєкту рефакторингу БД;

– застосувати модель на практиці задля відображення його переваг над іншими методами.

У рамках магістерської кваліфікаційної роботи необхідно дослідити процеси та механізми рефакторингу БД Інтернет-провайдера задля підвищення гнучкості БД.

Вимоги до цієї моделі – зрозумілість, яка виражається в тому, що вона повинна бути зрозумілою навіть для людей, які мають небагато досвіду та експертизи з рефакторингу БД. Для цього береться структура у форматі UserStory існуючої та бажаної системи. Необхідно розробити вдосконалити модель для пошуку кількісної оцінки проведення рефакторингу БД.

Після отримання результатів необхідно розглянути структури та поради щодо рефакторингу БД.

2 РОЗРОБКА МОДЕЛІ ОЦІНЮВАННЯ ДОЦІЛЬНОСТІ ПРОВЕДЕННЯ РЕФАКТОРИНГУ БАЗИ ДАНИХ ІНТЕРНЕТ-ПРОВАЙДЕРА

2.1 Опис моделі і методики управління ІТ-проєктом рефакторингу бази даних Інтернет-провайдера

Незважаючи на те, що рефакторинг БД як різновид ІТ-проєктів виник вже достатньо давно, рівень формальності опису рефакторингу залишається дуже низьким. Основну увагу дослідники зосереджують на створенні прикладних методик проведення рефакторингу БД. Як приклад такої методики розглянемо запропоновану Скоттом Амблером та його колегами у [4] методику рефакторингу БД. Вона являє собою набір процедур та практик, які допомагають безпечно та ефективно проводити рефакторинг БД у рамках проєкту розробки ПЗ. Ця методика пропонує підхід до рефакторингу, який дозволяє поступово покращувати якість БД та знижувати ризики, пов'язані зі зміною БД [4].

Ця методика заснована на принципах Agile-розробки та передбачає постійний аналіз та покращення БД у ході її ЖЦ. Рефакторинг БД у цій методиці передбачає поступове поліпшення структури та функціональності БД, без переривання її роботи та з мінімальним впливом на додатки, які використовують цю БД [4].

Методика рефакторингу БД Скотта Амблера може бути використана для різних цілей, наприклад, для оптимізації продуктивності БД, спрощення їх структури, поліпшення їх масштабованості тощо. Вона також дозволяє покращувати якість та надійність БД, зменшувати витрати на їх підтримку та розвиток, а також підвищувати зручність роботи з ними для кінцевих користувачів.

Методика рефакторингу БД представлена С. Амблером як процес, наведений на рис. 2.1 [16].

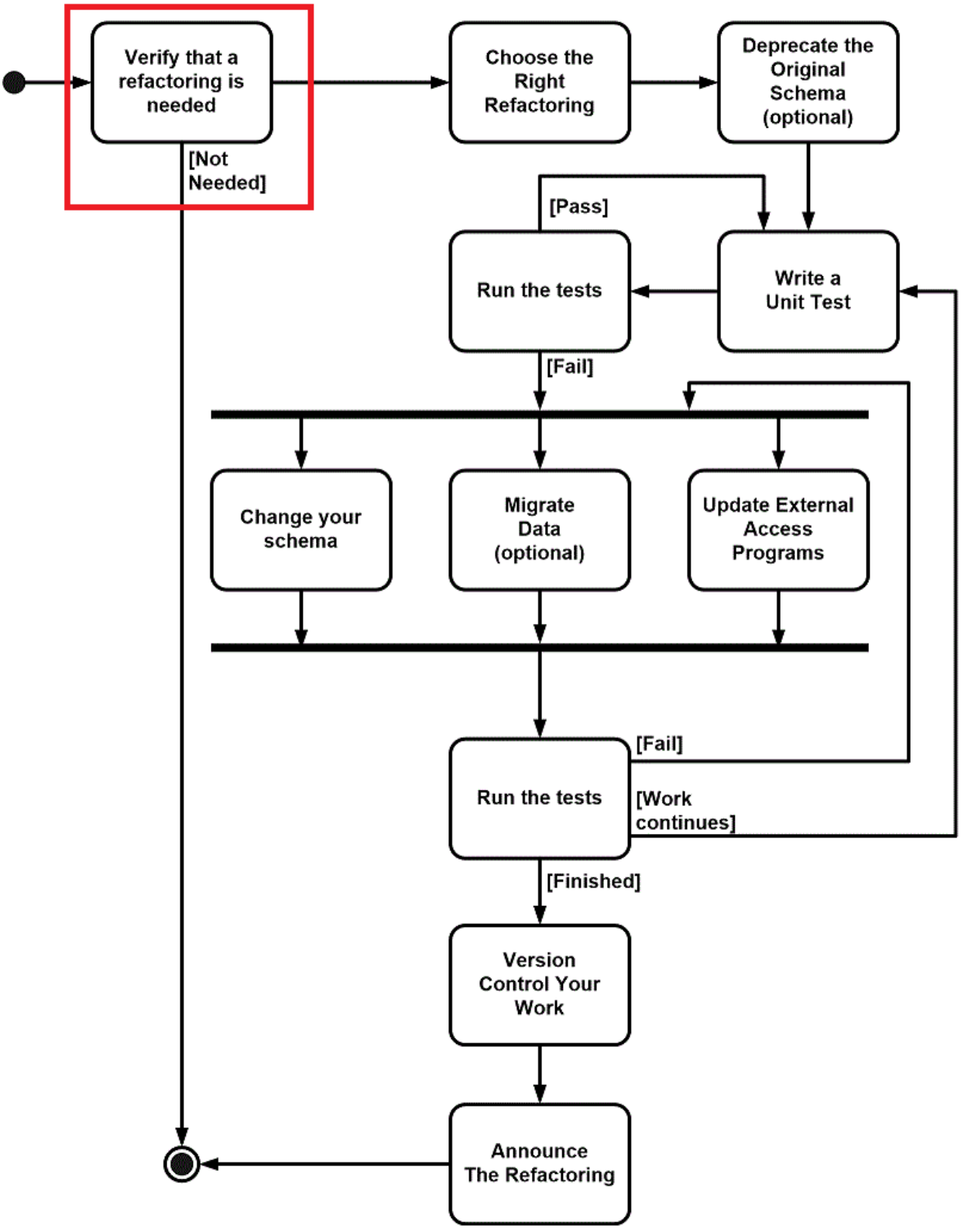

Рисунок 2.1 – Схема проведення процесу рефакторингу БД за методикою С. Амблера

З методики проведення процесу рефакторингу БД запропонованої С. Амблером можна виділити лише загальні рекомендації. Ця методика позбавлена конкретних порад щодо питання потрібен рефакторинг взагалі чи ні. Для того, щоб зрозуміти чи потрібен рефакторинг треба звертатися до експертів. В цій роботі пропонується розробити практичні поради щодо розуміння чи потрібен рефакторинг БД. Необхідно розробити сценарій за допомогою якого буде дана відповідь на це запитання.

Рефакторинг ІТ-продукту можна представити як комбінацію наступних видів операцій: додавання нової функції ІС, редагування існуючої функції ІС та видалення функції ІС, яка більше не використовуватиметься.

Незалежно від операції, що проводиться, тобто додавання чи зміна – бізнес-процес проведення рефакторингу діє за єдиним сценарієм. Цей сценарій використовується для підрахунку складності змін, що вносяться.

Сценарій виконання бізнес-процесу оцінки доцільності проведення рефакторингу складається з наступних кроків:

– 1 крок: необхідно створити запит на зміну, в якій прописується сутність та її атрибути;

– 2 крок: необхідно розрізнити, запит на додавання чи редагування. Якщо додавання – потрібно вибрати сутність із бібліотеки, де вони зберігаються та додавати опис кожної сутності з її доменом та атрибутами. Якщо змінюємо сутність – потрібно дістати опис існуючої функції, що змінюється;

– 3 крок: підрахувати кількість введених сутностей і атрибутів;

– 4 крок: перевірити чи збігаються ці сутності або атрибути з існуючими. При функції додавання описується лише бажаний стан. При функції зміни ми вже маємо опис функції;

– 5 крок: підрахувати кількість атрибутів, опис яких відсутній у вихідній БД. Це буде кількість внесених змін;

– 6 крок: отримання підсумкової кількісної оцінки змін. Чим ближче вона до одиниці, тим доцільніше розглядати процес реінжинірингу.

Сценарій представлено у вигляді діаграми потоків даних на рис. 2.2.

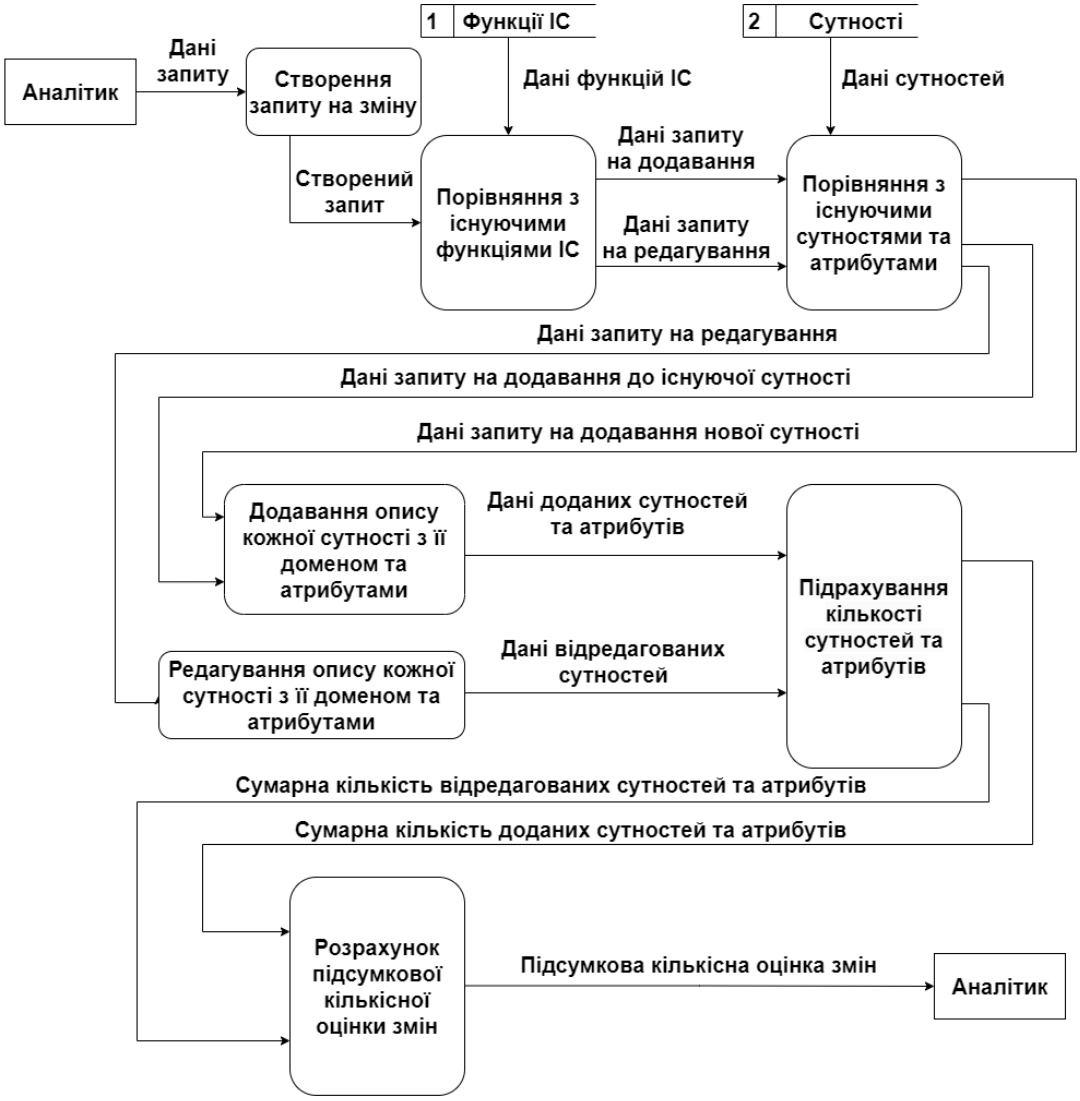

Рисунок 2.2 – Сценарій бізнес-процесу оцінки доцільності проведення рефакторингу

2.2 Розробка кількісної оцінки складності опису бази даних до проведення рефакторингу для подальшої оцінки складності рефакторингу, що проводиться

Результатом цього пункту буде формула кількісної оцінки складності проведення рефакторингу. Вона буде використовуватися для подальшого

прийняття рішення про доцільність проведення рефакторингу або реінжинірингу БД.

Складність проведення рефакторингу – це кількісна оцінка обсягу змін, внесених в опис БД, порівняно з обсягом опису БД до проведення рефакторингу.

Для того, щоб ці обсяги можна було порівнювати, необхідно використовувати однаковий математичний апарат і для опису обсягу змін, що вносяться, і для опису БД до проведення рефакторингу. Таким математичним апаратом пропонується вважати теоретико-множинний опис Entity – Relation Diagram (ERD), запропонований у статті [17].

Даний апарат обраний тому, що опис ERD є найпоширенішою формою представлення проектованої чи змінюваної БД, яка практично не залежить від конкретної моделі даних. Цей апарат є універсальним.

Будь-яку БД можна описати наступним кортежем [17]:

$$ERD_{DB} =< E, R, D >, \tag{2.1}$$

де $E = \{E_i\}$ – множина сутностей ERD, $i = 1, t$, t – кількість сутностей в БД, доцільність проведення рефакторингу якої оцінюється;

$R = \{R_i\}$ – множина зв'язків, визначених на сутності множини E та елементах цих сутностей, $i = 1, n$, $n -$ кількість зв'язків в БД, доцільність проведення рефакторингу якої оцінюється;

$D = \{D_l\}$ – множина доменів, визначених у ERD, $l = 1, p$, p – кількість доменів в БД, доцільність проведення рефакторингу якої оцінюється.

Кожна сутність із множини $\{E_i\}$ може бути також описана кортежем виду [17]:

$$E_i =< n_{E_i}, Tit_{E_i}, B_{E_i} >, \tag{2.2}$$

де n_{E_i} – ім’я сутності E_i;

Tit_{E_i} – заголовок сутності E_i;

$B_{E_i} = \{e_{E_i}^k\}, k = 1,2, \ldots$ – тіло сутності E_i;

$e_{E_i}^k$ – k-ий екземпляр сутності E_i.

Заголовок Tit_{E_i} сутності E_i в свою чергу може бути описаний кортежем наступного виду [17]:

$$Tit_{E_i} =< atr_{E_i}^j > \subseteq Atr_E, \tag{2.3}$$

де Atr_E – множина атрибутів, які використовуються для формування заголовків усієї множини сутностей E;

$atr_{E_i}^j$ – j-ий атрибут, який використовується для опису заголовку сутності E_i.

Кожен атрибут $atr_{E_i}^j$ сутності E_i може бути описаний кортежем виду [17]:

$$atr_{E_i}^j =< n_{E_i}^j, D_{E_i}^j >, \tag{2.4}$$

де $n_{E_i}^j$ – ім’я атрибуту $atr_{E_i}^j$;

$D_{E_i}^j$ – домен атрибуту $atr_{E_i}^j$, причому $D_{E_i}^j \in D$.

Зв’язком $R_i \in R$ в ERD прийнято називати іменовану значну асоціацію між двома сутностями чи сутності із собою. У загальному випадку зв’язок $R_i \in R$ між сутностями E_m та E_n може бути описаний кортежем виду [17]:

$$R_i =< n_{R_i}, Atr_{E_m}^{R_i}, Atr_{E_n}^{R_i}, Pow_{E_m}^{R_i}, Pow_{E_n}^{R_i}, S_{E_m}^{R_i}, S_{E_n}^{R_i} >, \quad (2.5)$$

де n_{R_i} – ім'я зв'язку R_i;

$Atr_{E_m}^{R_i} \subseteq Atr_{E_m}$ – підмножина атрибутів сутності E_m, що беруть участь в утворенні зв'язку R_i;

$Atr_{E_n}^{R_i} \subseteq Atr_{E_n}$ – підмножина атрибутів сутності E_n, що беруть участь в утворенні зв'язку R_i;

$Pow_{E_m}^{R_i}$ – потужність зв'язку R_i для сутності E_m (кількість екземплярів $e_{E_m}^k \in B_{E_m}$, що беруть участь в утворенні зв'язку R_i);

$Pow_{E_n}^{R_i}$ – потужність зв'язку R_i для сущности E_n (кількість екземплярів $e_{E_n}^k \in B_{E_n}$, що беруть участь в утворенні зв'язку R_i);

$S_{E_m}^{R_i}$ – ступінь участі сутності E_m у зв'язку R_i, яка визначає обов'язковість участі екземплярів $e_{E_m}^k \in B_{E_m}$ у зв'язку R_i;

$S_{E_n}^{R_i}$ – ступінь участі сутності E_n у зв'язку R_i, яка визначає обов'язковість участі екземплярів $e_{E_n}^k \in B_{E_n}$ у зв'язку R_i.

Аналізуючи множину зв'язків $R = \{R_i\}$ за ступенем обов'язковості участі сутностей у освіті даних зв'язків, можна розглянути цю множину як сукупність підмножин [17]:

$$R = R^{00} \cup R^{10} \cup R^{11}, \quad (2.6)$$

де R^{00} – підмножина зв'язків, необов'язкових з боку сутностей E_m и E_n;

R^{10} – підмножина зв'язків, обов'язкових з боку сутності E_n і необов'язкових з боку сутності E_m;

R^{11} – підмножина зв'язків, обов'язкових з боку сутностей E_m и E_n.

Тоді умови належності зв'язку R_i до одного з розглянутих вище підмножин, виділених за ступенем обов'язковості участі сутностей освіти даного зв'язку; визначаються так [17]:

$$R_i \in R^{00} \text{ при умові } \begin{cases} S_{E_m}^{R_i} = 0; \\ S_{E_n}^{R_i} = 0; \end{cases} ; R_i \in R^{10} \text{ при умові } \begin{cases} S_{E_m}^{R_i} = 1; \\ S_{E_n}^{R_i} = 0; \end{cases} \quad (2.7)$$

$$R_i \in R^{11} \text{ при умові } \begin{cases} S_{E_m}^{R_i} = 1; \\ S_{E_n}^{R_i} = 1; \end{cases} \quad (2.8)$$

Причому

$$S_{E_k}^{R_i} = \begin{cases} 1 \text{ якщо } \exists e_{E_k}^i \in B_{E_k}; \\ 0 \text{ в іншому випадку.} \end{cases} \quad (2.10)$$

Аналізуючи найпоширеніші типи зв'язків, що утворюють множину $R = \{R_i\}$, виділені за кількістю сутностей, що беруть участь в освіті екземплярів; дану множину можна описати так [17]:

$$R = R^{om} \cup R^{mm}, \quad (2.11)$$

де R^{om} – підмножина зв'язків типу «один – до багатьох»;

R^{mm} – підмножина зв'язків типу «багато – до багатьох». Зв'язок типу «один – одного» у разі розглядається як окремий випадок зв'язку типу «один – до багатьох».

Тоді умови належності зв'язку R_i до одного з розглянутих вище підмножин, виділених за кількістю сутностей, що беруть участь в освіті екземплярів; визначаються так [17]:

$$R_i \in R^{om} \text{ при умові } \begin{cases} Pow_{E_m}^{R_i} = 1; \\ Pow_{E_n}^{R_i} = k, k = 1, p. \end{cases} \quad (2.12)$$

$$R_i \in R^{mm} \text{ при умові } \begin{cases} Pow_{E_m}^{R_i} = k, k = 1, p; \\ Pow_{E_n}^{R_i} = l, l = 1, p. \end{cases} \quad (2.13)$$

Для ERD у нотації IDEF1X немає можливості позначати ступінь участі сутності з боку «багато», і, при переході від логічної моделі даних до фізичної, підмножина зв'язків R^{mm} відображається в підмножину зв'язків R^{om}.

Щоб отримати детальний опис зв'язків на кшталт «один – до багатьох», введемо поняття первинного ключа PK_{E_i} для сутності E_i. Такий ключ описуватиметься виразом [17]:

$$PK_{E_i} = Atr_{E_i}^{PK} \subseteq Atr_{E_i}, \quad (2.14)$$

де $Atr_{E_i}^{PK} =< atr_{E_i}^{jPK} >$ – підмножина атрибутів, що утворюють первинний ключ для сутності E_i. Елементи даного підмножини виділяються за такою умовою [17]:

$$\begin{cases} val_{E_i}^{kPK} = \{val_{E_i}^{kjPK}\} \subseteq < val_{E_i}^{kj} >; val_{E_i}^{kjPK} \neq \emptyset; r(Atr_{E_i}^{PK}) \to min; \\ val_{E_i}^{kPK} \neq val_{E_i}^{lPK} \text{ якщо } k \neq l; \\ val_{E_i}^{kPK} = val_{E_i}^{lPK} \text{ якщо } k = l. \end{cases} \quad (2.15)$$

де $val_{E_i}^{kPK}$ – значення первинного ключа для екземпляра сутності$e_{E_i}^{k}$;

$val_{E_i}^{kjPK}$ – значення j-го атрибуту, який бере участь в утворенні первинного ключа, що є присутнім в екземплярі сутності $e_{E_i}^k$;

$r\left(Atr_{E_i}^{PK}\right)$ – функція, що визначає потужність підмножини атрибутів $Atr_{E_i}^{PK}$, утворюють первинний ключ сутності E_i.

За аналогією з визначенням первинного ключа введемо поняття зовнішнього ключа FK_{E_m} для якоїсь сутності E_m. Такий ключ можна описати виразом [17]:

$$FK_{E_m} = Atr_{E_m}^{FK} \subseteq Atr_{E_m}, \tag{2.16}$$

де $Atr_{E_m}^{FK} =< atr_{E_m}^{jFK} >$ – підмножина атрибутів, що утворюють первинний ключ для сутності E_m. Елементи даного підмножини виділяються за такою умовою [17]:

$$\begin{cases} \exists E_n, \text{причому для } e_{E_n}^k \in B_{E_n} \exists e_{E_m}^l \in B_{E_m}; \\ val_{E_n}^{kPK} = val_{E_m}^{lFK}; val_{E_n}^{kPK} = \{val_{E_n}^{kjPK}\}; val_{E_m}^{lFK} = \{val_{E_m}^{ljFK}\}; \\ val_{E_m}^{ljFK} \in D^j; val_{E_n}^{kjPK} \in D^j; D^j \in D, j = 1, t; \end{cases} \tag{2.17}$$

де $val_{E_n}^{kPK}$ – значення первинного ключа для екземпляра сутності$e_{E_n}^k$;

$val_{E_m}^{lFK}$ – значення вторинного ключа для екземпляра сутності $e_{E_m}^l$;

$val_{E_n}^{kjPK}$ – значення j-го атрибуту, який бере участь в утворенні первинного ключа, що є присутнім в екземплярі сутності $e_{E_n}^k$;

$val_{E_m}^{ljFK}$ – значення j-го атрибуту, який бере участь в утворенні зовнішнього ключа, що є присутнім в екземплярі сутності $e_{E_m}^l$;

D^j – домен атрибутів $atr_{E_n}^{jPK}$ та $atr_{E_m}^{jFK}$.

Тоді формулу (2.7) для зв'язку з підмножини R^{om} можна перетворити так [17]:

$$R_i^{om} =< n_{R_i}, FK_{E_m}^{R_i}, PK_{E_n}^{R_i}, Pow_{E_m}^{R_i}, Pow_{E_n}^{R_i}, S_{E_m}^{R_i}, S_{E_n}^{R_i} >, \quad (2.18)$$

при умові

$$\begin{cases} Pow_{E_m}^{R_i} \geq 1; \\ Pow_{E_n}^{R_i} = 1. \end{cases} \quad (2.19)$$

Зв'язок R_i^{om} буде обов'язковим з боку сутності E_n, якщо виконується умова $S_{E_n}^{R_i} = 1$. Зв'язок R_i^{om} буде ідентифікуючою, якщо вірна умова [17]:

$$\begin{cases} FK_{E_m}^{R_i} \subseteq PK_{E_m}^{R_i}; \\ S_{E_n}^{R_i} = 1. \end{cases} \quad (2.20)$$

При цьому сутність E_m є слабкою сутністю (її первинний ключ може бути визначений без участі екземпляра сутності E_n), а E_n – сильною сутністю (за умови, що вона не є слабкою сутністю в іншому зв'язку).

Використання запропонованого математичного апарату дозволяє вирішувати задачі кількісного оцінювання складності актуальної БД, складності сукупності запитів на зміни інформаційної системи, яка експлуатується, а також обсягу змін, які слід внести під час рефакторингу БД, що пропонується, шляхом підрахунків кількості атрибутів, які беруть участь в утворенні сутностей та зв'язків, виділених у описах відповідних БТ та сукупності запитів на зміни.

Виходячи із запропонованого у [17] теоретико-множинного апарату опису ERD, складність актуальної БД можна оцінити через кількість окремих об'єктів, з яких складається ця БД. У загальному випадку, виходячи з формули (2.1), цю кількість можна підрахувати наступним чином:

$$C_{DB} = |E| + |R| + |D|. \quad (2.21)$$

Однак, дана формула не враховує особливості проведення рефакторингу. Так, передбачається, що для загального домену БД під час рефакторингу, викликаного найчастіше додаванням і зміною окремих послуг, зміни не очікуються. Тому враховувати кількість описів елементів домену БД загалом недоцільно. Крім того, у даній формулі відсутня можливість уніфікованого опису сутностей та зв'язків аналізованої БД.

Таку можливість уніфікованого опису виникає, коли під час кількісного оцінювання складності актуальної БД відбувається перехід від підрахунків окремих сутностей та зв'язків цієї БД до підрахунків окремих атрибутів, що беруть участь в утворенні відповідних сутностей та зв'язків. Так, кількісну оцінку складності сутностей, що утворюють актуальну БД, згідно з формулами (2.2) та (2.3), пропонується визначити як кількість атрибутів, які утворюють заголовок кожної конкретної сутності, яка входить до опису актуальної БД. Цю кількість атрибутів можна підрахувати за наступним виразом:

$$C_{DB_E} = \sum_{i=1}^{t} \left| \langle atr_{E_i}^{j} \rangle \right|. \quad (2.22)$$

За аналогією з формулою (2.22), кількісну оцінку складності зв'язків, які наявні між сутностями актуальної БД, пропонується визначити як кількість атрибутів, які беруть участь у утворенні кожного конкретного зв'язку між батьківською сутністю E_m та дочірньою сутністю E_n. Цю кількість атрибутів можна підрахувати за наступним виразом:

$$C_{DB_R} = \sum_{i=1}^{z} \left| Atr_{E_m}^{R_i} \right| + \sum_{i=1}^{z} \left| Atr_{E_n}^{R_i} \right|, \quad (2.23)$$

де z – в даному випадку кількість зв’язків, визначених на сутностях множини *E* актуальної БД та елементах цих сутностей.

Тоді кількісну оцінку складності актуальної БД (2.21) можна, згідно з урахуванням формул (2.22) та (2.23), представити таким чином:

$$C_{DB} = \sum_{i=1}^{t}\left|\langle atr_{E_i}^{j}\rangle\right| + \left(\sum_{i=1}^{z}\left|Atr_{E_m}^{R_i}\right| + \sum_{i=1}^{z}\left|Atr_{E_n}^{R_i}\right|\right), \qquad (2.24)$$

Отже, у пункті було проаналізовано теоретико-множинний апарат опису ERD [17], виведено формулу (2.24) для розрахунку кількісної оцінки складності БД до проведення рефакторингу. Використання цієї формули дозволить провести порівняльний аналіз складності рефакторингу, що проводиться, з актуальною БД.

2.3 Розробка кількісної оцінки складності опису змін, що вносяться для подальшої оцінки складності рефакторингу, що проводиться

Для пошуку кількісної оцінки складності опису змін, що вносяться використовується розробка запитів на зміни.

Запит на зміни (Request for Changes, RFC) в IT-проєкті – це формальний документ, який містить запит на внесення змін до проєкту. Він включає опис того, що необхідно змінити в проєкті, чому це необхідно і які можуть бути наслідки внесення цих змін. У RFC має бути зазначена причина запиту, а також опис необхідних змін та очікуваних результатів.

RFC дозволить виділити сутності та їх атрибути з яких вони складаються. Процес перетворення RFC до виду ERD у роботі не розглядається, вважається, що він вже приведений до виду ERD. У цій роботі

RFC буде представлено у вигляді UserStory у двох станах: «поточний» та «бажаний» (рисунок 2.3).

Рисунок 2.3 – Шаблон представлення системи у вигляді UserStory

UserStory 1. Функція «Додавання нової послуги. Адміністратор» – поточний стан.

Як Адміністратор, я хочу додавати нову послугу для того, щоб користувачі могли її замовляти.

Критерії приймання:

Враховуючи, що набір пакетів та налаштувань вже відомий.

Коли додається нова послуга в систему.

Тоді система створює нову таблицю та створює набір пакетів (рисунок 2.4).

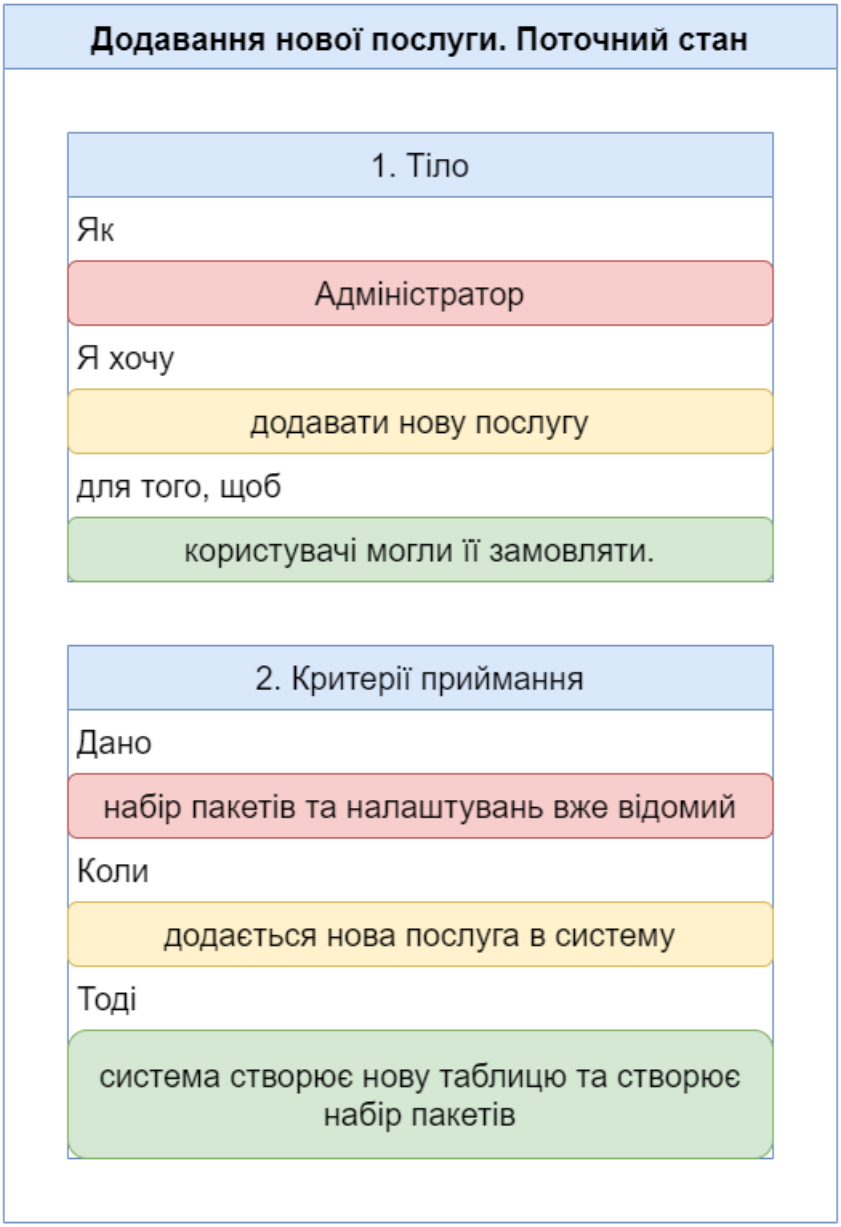

Рисунок 2.4 – UserStory 1. Функція «Додавання нової послуги. Адміністратор» – поточний стан

UserStory 2. Функція «Створення замовлення. Користувач» – поточний стан.

Як Користувач, я хочу замовити послугу для того, щоб нею користуватися.

Критерії приймання:

Враховуючи, що послуга існує.

Коли обираю послугу в системі.

Тоді система надає заздалегідь створений набір пакетів (рисунок 2.5).

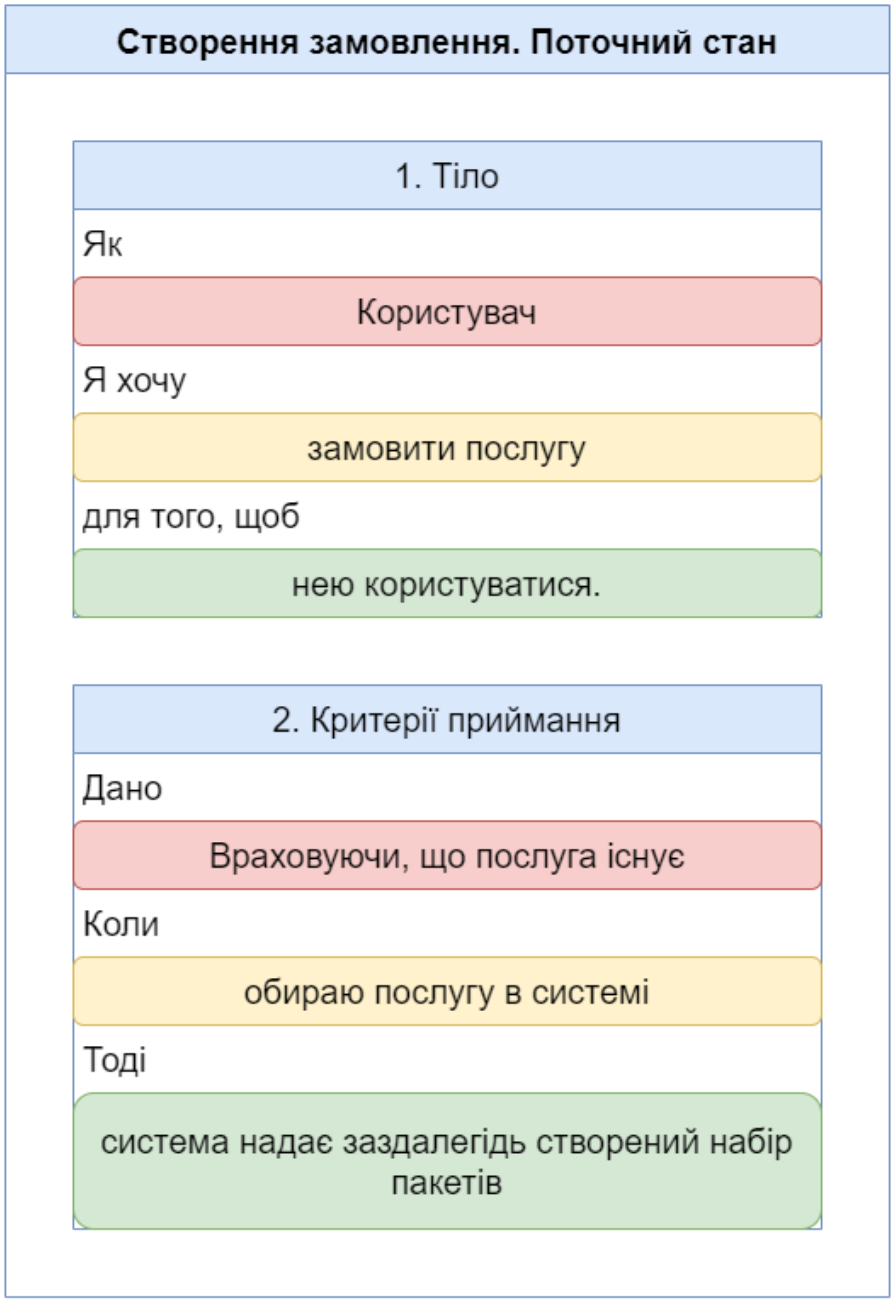

Рисунок 2.5 – UserStory 2. Функція «Створення замовлення. Користувач» – поточний стан

UserStory 3. Функція «Створення замовлення. Користувач» – бажаний стан.

Як Користувач, я хочу замовити послугу для того, щоб нею користуватися.

Критерії приймання:

Враховуючи, що послуга існує.

Коли обираю послугу в системі.

Тоді система надає можливість налаштувати параметри пакету (рисунок 2.6).

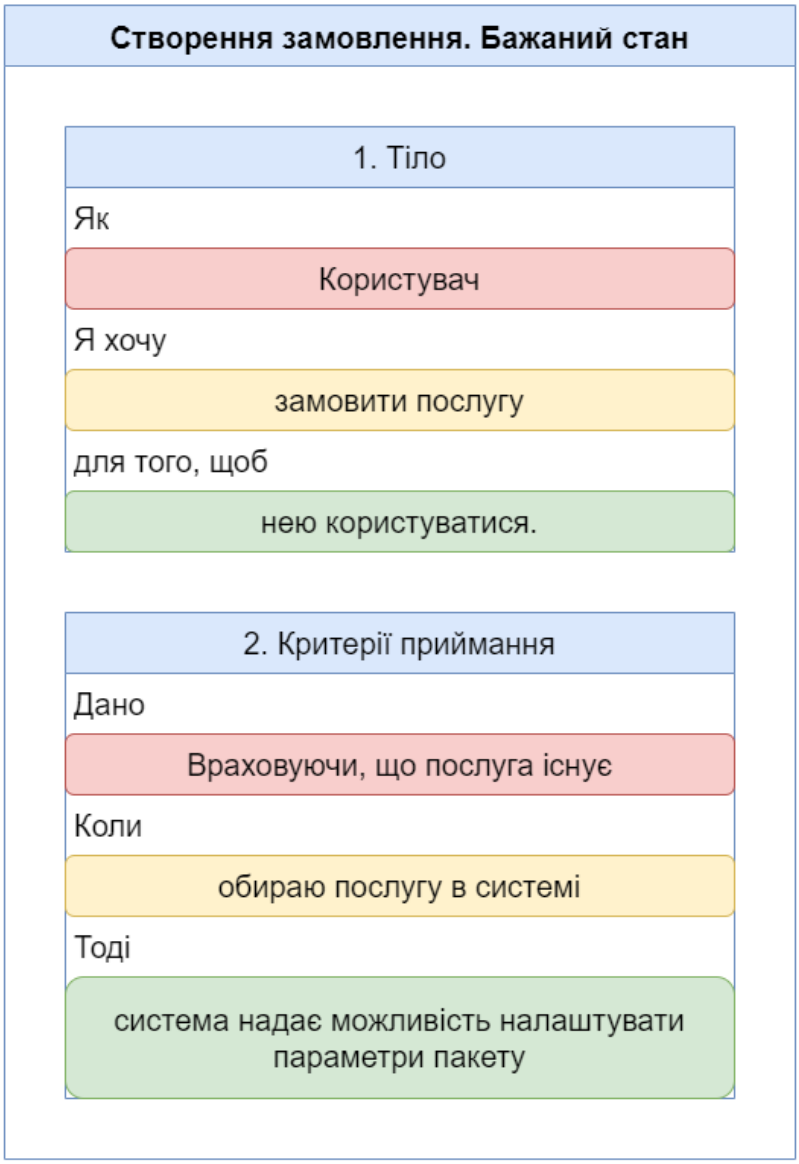

Рисунок 2.6 – UserStory 3. Функція «Створення замовлення. Користувач» – бажаний стан

На основі наведених вище UserStory буде розраховуватися кількісна оцінка проведення рефакторингу.

Для розрахунку кількісної оцінки складності сукупності запитів на зміни ІС, яка експлуатується, визначимо можливість опису таких запитів на зміну класичною моделлю вимог, запропонованою у методології SSADM [18]. За цією моделлю кожний запит на зміну може бути описаний відповідною ERD.

Тому, згідно з формулою (2.1), множина ERD, які описують усю сукупність запитів на зміну.

Запит на зміни можна описати наступним кортежем [17]:

$$ERD_{RFC} = \bigcup_{r=1}^{q} ERD_{RFC_r} = \bigcup_{r=1}^{q} \langle E_r, R_r, D_r \rangle, \quad (2.25)$$

де r – ідентифікатор ERD, яка використовується для опису r-го запиту на зміну;

q – кількість запитів на зміну у сукупності, на основі якої робиться оцінювання можливості проведення рефакторингу БД.

Згідно з формулою (2.21), в загальному випадку кількість окремих об’єктів, з яких складаються ERD, що описують сукупність запитів на зміни, можна підрахувати наступним чином:

$$C_{RFC} = \left|\bigcup_{r=1}^{q} E_r\right| + \left|\bigcup_{r=1}^{q} R_r\right| + \left|\bigcup_{r=1}^{q} D_r\right|. \quad (2.26)$$

Однак, дана формула не враховує особливості проведення рефакторингу. Так, передбачається, що загальний домен БД під час рефакторингу, викликаного найчастіше додаванням і зміною окремих послуг, не змінюватиметься, тому враховувати кількість описів елементів домену БД загалом недоцільно. Крім того, у формулі (2.26) відсутня можливість уніфікованого опису сутностей та зв’язків аналізованої БД. Це означає, що для кількісного оцінювання складності запитів на зміни ІС, яка експлуатується, треба враховувати тільки ті атрибути, описи яких у цих запитах повністю або частково відрізняються від описів аналогічних атрибутів у актуальній БД.

Виходячи з цього положення, визначимо оцінку складності сукупності запитів на зміни як оцінку кількості елементів у множинах, які є результатами різниці множини описів атрибутів з ERD, яка описує кожен запит на зміну, та множини описів атрибутів з ERD актуальної БД.

Для розрахунку такої оцінки введемо наступні підмножини:

$$Atr_{E_{DB}} = \bigcup_{i=1}^{t} \langle atr_{E_i}^{j} \rangle; \qquad Atr_{E_{RFC}} = \bigcup_{r=1}^{q} \bigcup_{i=1}^{t} \langle atr_{E_{i_r}}^{j} \rangle. \qquad (2.27)$$

Ці підмножини описують, відповідно, підмножину атрибутів актуальної БД та підмножину атрибутів сукупності запитів на зміни ІС, яка експлуатується.

Тоді оцінку складності сутностей, що утворюють ERD сукупності запитів на зміни, можна розрахувати за наступною формулою:

$$C_{RFC} = \left| Atr_{E_{RFC}} \backslash Atr_{E_{DB}} \right|, \qquad (2.28)$$

де $Atr_{E_{RFC}}$ – множина атрибутів, які зазнали змін;

$Atr_{E_{DB}}$ – множина атрибутів усієї БД.

Для оцінювання складності зв’язків, що утворюють ERD сукупності запитів на зміни, введемо наступні підмножини:

$$Atr_{E_m}^{DB} = \bigcup_{i=1}^{t} \langle atr_{E_m}^{R_i} \rangle; \qquad Atr_{E_n}^{DB} = \bigcup_{i=1}^{t} \langle atr_{E_n}^{R_i} \rangle; \qquad (2.29)$$

$$Atr_{E_m}^{RFC} = \bigcup_{r=1}^{q} \bigcup_{i=1}^{t} \langle atr_{E_m}^{R_{i_r}} \rangle; \qquad Atr_{E_n}^{RFC} = \bigcup_{r=1}^{q} \bigcup_{i=1}^{t} \langle atr_{E_n}^{R_{i_r}} \rangle. \qquad (2.30)$$

Ці підмножини описують сукупності атрибутів, які беруть участь в утворенні зв’язків між сутностями ERD актуальної БД (вираз (2.29)) та сутностями сукупності ERD запитів на зміни (вираз (2.30)).

Тоді оцінку складності зв’язків, що утворюють ERD сукупності запитів на зміни, можна розрахувати за наступною формулою:

$$C_{RFC_R} = \left| Atr_{E_m}^{RFC} \backslash Atr_{E_m}^{DB} \right| + \left| Atr_{E_n}^{RFC} \backslash Atr_{E_n}^{DB} \right|. \qquad (2.31)$$

Виходячи з отриманих результатів, оцінку (2.26) складності сукупності запитів на зміни ІС, яка експлуатується, можна описати наступним виразом:

$$C_{RFC} = \left|Atr_{E_{RFC}} \backslash Atr_{E_{DB}}\right| + \left(\left|Atr_{E_m}^{RFC} \backslash Atr_{E_m}^{DB}\right| + \left|Atr_{E_n}^{RFC} \backslash Atr_{E_n}^{DB}\right|\right). \quad (2.32)$$

Отримані у вигляді виразів (2.24) та (2.32) оцінки складності актуальної БД та сукупності запитів на зміни ІС, яка експлуатується, пропонується використати для кількісного оцінювання обсягу змін, які слід внести під час рефакторингу БД. Але на відміну від абсолютних кількісних оцінок (2.24) та (2.32) показник, який пропонується використовувати для кількісного оцінювання такого обсягу змін, слід розробити відносним. Це дозволить спростити прийняття рішення щодо верхньої межі доцільності проведення рефакторингу БД, за якою процес власне рефакторингу БД переходить у процес повного реінжинірингу цієї ж БД.

В загальному випадку цей показник буде мати наступний вигляд:

$$C_{REF} = {}^{C_{RFC}}/_{C_{DB}}. \quad (2.33)$$

Згідно з формулою (2.24) та (2.32) вираз (2.33) прийме такий вигляд:

$$C_{REF} = \frac{\left|Atr_{E_{RFC}} \backslash Atr_{E_{DB}}\right| + \left(\left|Atr_{E_m}^{RFC} \backslash Atr_{E_m}^{DB}\right| + \left|Atr_{E_n}^{RFC} \backslash Atr_{E_n}^{DB}\right|\right)}{\sum_{i=1}^{t}\left|\langle atr_{E_i}^{j}\rangle\right| + \left(\sum_{i=1}^{z}\left|Atr_{E_m}^{R_i}\right| + \sum_{i=1}^{z}\left|Atr_{E_n}^{R_i}\right|\right)}. \quad (2.34)$$

Для спрощення технології розрахунку значень показника (2.34) визнаємо як вірні такі рівняння:

$$\left|Atr_{E_{DB}}\right| = \sum_{i=1}^{t}\left|\langle atr_{E_i}^{j}\rangle\right|;\ \left|Atr_{E_m}^{DB}\right| = \sum_{i=1}^{z}\left|Atr_{E_m}^{R_i}\right|;\ \left|Atr_{E_n}^{DB}\right| = \sum_{i=1}^{z}\left|Atr_{E_n}^{R_i}\right|. \quad (2.35)$$

Тоді правила розрахунку значень за виразом (2.35) технологічно спростяться за рахунок уніфікації процедур формування окремих підмножин, а сам вираз (2.34) буде мати наступний вигляд:

$$C_{REF} = \frac{\left|Atr_{E_{RFC}} \backslash Atr_{E_{DB}}\right| + (|Atr_{E_m}^{RFC} \backslash Atr_{E_m}^{DB}| + |Atr_{E_n}^{RFC} \backslash Atr_{E_n}^{DB}|)}{|Atr_{E_{DB}}| + \left(\left|Atr_{E_m}^{DB}\right| + \left|Atr_{E_n}^{DB}\right|\right)}, \quad (2.36)$$

Використання запропонованого показника (2.36) дозволяє визначити верхню межу доцільності проведення рефакторингу БД. Виходячи з практичного досвіду тут і в подальшому пропонується визначати процес рефакторингу доцільним, якщо значення показника (2.36) не буде перевищувати 0,5. Якщо показник (2.36) буде приймати під час визначення доцільності проведення рефакторингу БД значення, більші за 0,5 (тобто при виконанні запитів на зміни потрібно змінити більше половини актуальної БД), то слід визнати відповідні роботи роботами з реінженирінгу БД. Але цей висновок слід вважати попереднім і таким, що потребує підтвердження чи спростування шляхом подальших досліджень.

2.4 Висновки до другого розділу

Розроблені формальні описи оцінок складності актуальної БД та сукупності запитів на зміни ІС, яка експлуатується, дозволяють оцінити кількість елементів, з яких складаються ERD, що описують актуальну БД та кожен із сукупності запитів на зміни. В основу запропонованих оцінок покладено теоретико-множинний апарат, який описує, у тому числі, ERD через кортежі та підмножини атрибутів. Використання такого уніфікованого формального опису дозволяє в подальшому використовувати для автоматизованого розрахунку значень цих оцінок одні й ті ж процедури.

На основі розроблених формальних оцінок було запропоновано показник кількісного оцінювання обсягу змін, які слід внести під час рефакторингу БД. Визначено два основних варіанти розрахунку цього показника за виразами (2.34) та (2.36) відповідно. Описано особливість використання значень цього показника під час процесу перевірки доцільності проведення робіт з рефакторингу БД.

Слід зазначити, що отримані результати дозволяють визначити складність та обсяг змін, які слід внести під час рефакторингу БД. Але ці результати не дозволяють оцінити доцільність проведення робіт з рефакторингу БД як окремого різновиду ІТ-проєктів. Тому найближча перспектива подальших досліджень полягає у модифікації існуючих моделей оцінювання ІТ-проєктів (зокрема, параметричних моделей COCOMO2) для оцінювання трудовитрат, витрат часу та потреби у персоналі ІТ-проєкту рефакторингу БД на основі отриманих оцінок.

3 АДАПТАЦІЯ МОДЕЛІ ОЦІНЮВАННЯ ТРУДОВИТРАТ НА ПРОВЕДЕННЯ РЕФАКТОРИНГУ БАЗИ ДАНИХ ІНТЕРНЕТ-ПРОВАЙДЕРА

3.1 Опис SQL-команд які використовуються під час проведення рефакторингу бази даних

Існує багато типів SQL-команд, але для проведення рефакторингу необхідно розглядати тільки перший тип команд, а саме Data Definition Language (DDL). Ці команди відповідають за роботу з таблицями (рисунок 3.1) [19].

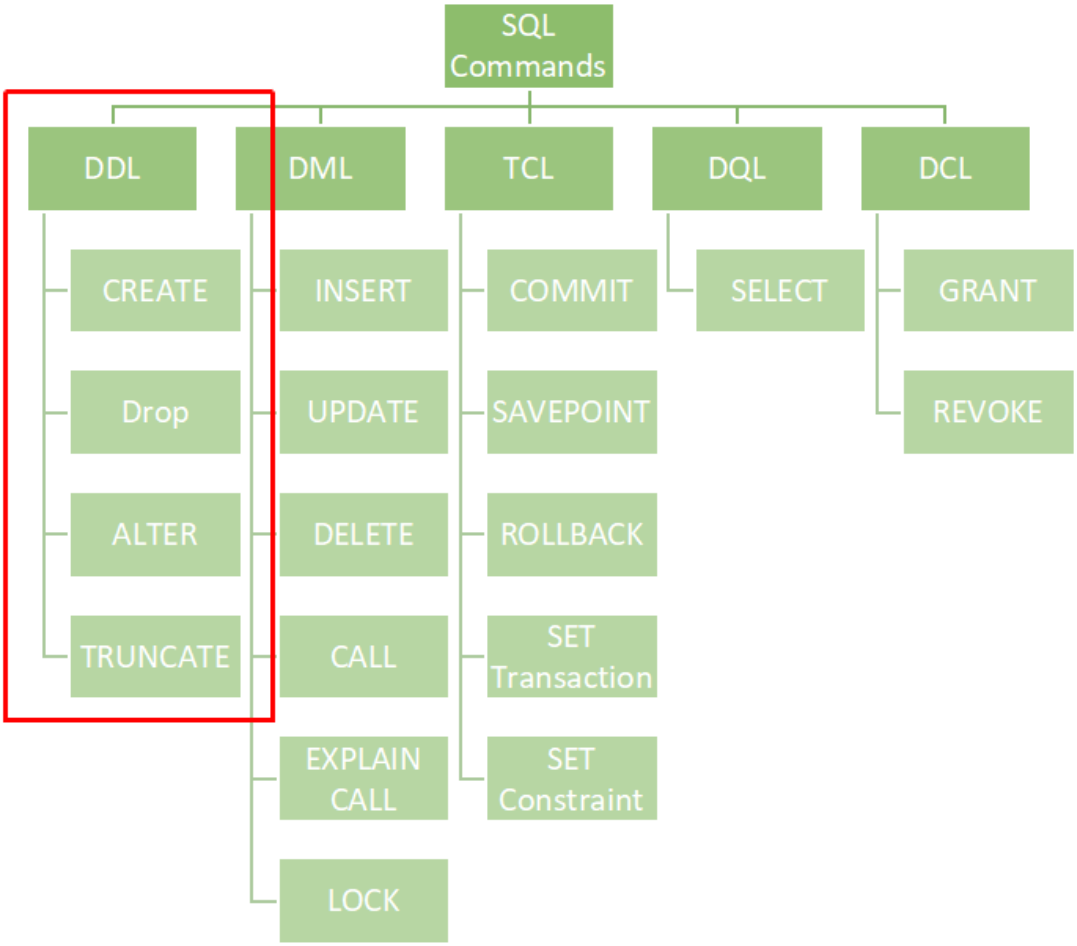

Рисунок 3.1 – Існуючі типи SQL-команд

До цього типу команд належать наступні операції над таблицями [19]:

– створення (create): використовується при рефакторингу БД для створення нової таблиці при додаванні нової функції системи;

– видалення (drop): використовується при рефакторингу БД для видалення існуючої таблиці, наприклад, коли треба відмовитися від існуючої сутності або функції системи;

– редагування (alter): використовується при рефакторингу БД для зміни структури таблиці, наприклад, додавання чи видалення стовпців, змінювання типу даних, встановлювання обмежень та індексів. Цю операцію також можна використовувати для перейменування таблиці або її стовпців;

– очищення (truncate): використовується при рефакторингу БД для очищення існуючої таблиці.

Шаблон команди створення зображено на рис. 3.2 [20].

```
CREATE TABLE <name> (
    <column 1>,
    <column 2>,
    .........,
    <column n>,
    PRIMARY KEY (<column 1>, <column 2>, ..., <column n>),
    FOREIGN KEY (<column 1>, <column 2>, ..., <column n>)
        REFERENCES <table name> (<column 1>, <column 2>, ..., <column n>)
);
```

Рисунок 3.2 – Шаблон команди створення

Шаблон команди видалення зображено на рис. 3.3 [20].

```
DROP TABLE/VIEW/CONSTRAIN <name>;
```

Рисунок 3.3 – Шаблон команди видалення

Шаблон команди редагування зображено на рис. 3.4 [20].

```
ALTER <ObjectType> <ObjectName>
    <Operation 1> <Parameters 1>,
    <Operation 2> <Parameters 2>,
    .........,
    <Operation n> <Parameters n>;
```

Рисунок 3.4 – Шаблон команди редагування

Шаблон команди очищення зображено на рис. 3.5 [20].

```
TRUNCATE TABLE <name>;
```

Рисунок 3.5 – Шаблон команди очищення

Для розрахунку кількості рядків кожної з команд пропонується використовувати пропорцію до кількісті атрибутів один до одного:

$$Size_{CR_r} = \left|\langle atr_{E_r}^{j}\rangle\right| + 4; j = 1 \dots N\ , \tag{3.1}$$

$$Size_{DR_r} = 1\ , \tag{3.2}$$

$$Size_{AL_r} = \left|\langle atr_{E_r}^{j}\rangle\right| + 1; j = 1 \dots N\ , \tag{3.3}$$

$$Size_{TR_r} = 1, \tag{3.4}$$

де r – сутність з запиту на зміни;

N – кількість атрибутів у сутності.

У формулі (3.1) додатковий доданок становить суму кількості строк, які реалізують наступне: тип команди, оголошення первинного ключа, оголошення вторинного ключа та посилання цих ключів на іншу таблицю. Тому додатковий доданок становить 4.

У формулі (3.2) немає операцій з атрибутами, тому залишається тільки додатковий доданок, який дорівнює сумі кількісті строк в команді видалення, яка завжди дорівнює 1.

У формулі (3.3) додатковий доданок становить 1, який включає тільки об’явлення типу команди.

У формулі (3.4) немає операцій з атрибутами, тому залишається тільки додатковий доданок, який дорівнює сумі кількісті строк в команді очищення, яка завжди дорівнює 1.

Отже, у подальших розрахунках оцінки трудовитрат та витрат часу на проведення рефакторингу БД Інтернет-провайдера доцільно буде використовувати SQL-команди тільки типу DDL.

3.2 Оцінка трудовитрат та витрат часу на проведення рефакторингу бази даних Інтернет-провайдера за моделлю COCOMO2

Модель COCOMO2 полягає в розрахунку трудовитрат (PM – person-months) та має загальний вигляд [21]:

$$PM = 2{,}94 \times [Size']^E \times \prod_{i=1}^{7} EM_i + PM_M, \quad (3.5)$$

де A – числовий коефіцієнт, який для моделей COCOMO та COCOMO2 дорівнює 2,5;

$Size'$ – кількість рядків вихідного програмного коду, яка розраховується за наступною формулою [21]:

$$Size' = Size \times \left(1 + \frac{BRAK}{100}\right), \quad (3.6)$$

де BRAK – показник, який відображує змінність вимог у IT-проєкті і являє собою відсоток коду, який викидається з IT-проєкту в результаті зміни вимог;

E – показник ступеня в який слід возвести значення Size′, значення якого розраховується за наступною формулою [21]:

$$E = 0{,}91 + 0{,}01 \times \sum_{j=1}^{5} SF_j, \quad (3.7)$$

де SF_j – драйвера масштабів IT-проєкту (PREC (показник безпрецедентності продукту); FLEX (показник гнучкості розробки продукту); RESL (показник надання переваги архітектурним або ризикованим рішенням); TEAM (показник згуртованості команди); PMAT (показник зрілості процесів розробки продукту));

EM_i – драйвера витрат IT-проєкту (RCPX (реалізованість і складність продукту); RUSE (повторне використання, яке вимагається); PDIF (складність платформи); PERS (характеристики персоналу); PREX (досвід персоналу); FCIL (використовувані засоби); SCED (розклад)).

PM_M – частина моделі COCOMO2, яка не використовується для формули зміни коду.

У формулі (3.6) для випадку модифікації продукту, що експлуатується, оцінка трудовитрат розраховується наступним чином [21]:

$$(Size)_M = (SizeAdded + SizeModified) \times MAF, \quad (3.8)$$

де $SizeAdded$ – кількість рядків SQL-коду для операцій додавання при проведенні рефакторингу БД;

$SizeModified$ – кількість рядків SQL-коду для операцій редагування при проведенні рефакторингу БД;

MAF – коригуючий коефіцієнт проведення рефакторингу БД, який розраховується за наступною формулою [21]:

$$MAF = 1 + \left(\frac{SU}{100} \times UNFM\right), \quad (3.9)$$

де SU – штраф, викликаний витратами на розуміння ПЗ;

$UNFM$ – показник рівня відокремленості команди.

Отримана оцінка трудовитрат (PM) далі використовується для розрахунку витрат часу за наступною формулою [21]:

$$TDEV = [3{,}67 \times (PM_{NS})^{F}] \times \frac{SCED\%}{100}, \quad (3.10)$$

де PM_{NS} – людино-місяці, оцінені без драйвера витрат SCED (номінальний графік);

SCED – необхідне стиснення розкладу;

F – показник масштабування для рівняння розкладу, значення якого розраховується за наступною формулою [21]:

$$F = (0{,}28 + 0.2 \times [E - 0{,}91]), \quad (3.11)$$

де E – масштабний показник для рівняння зусиль.

Проблема моделі COCOMO2 полягає в тому, що вона не надає формули для розрахунку $SizeAdded$ та $SizeModified$. Тому необхідно розробити формулу для їх розрахунку.

3.3 Розрахунок обсягу коду

За допомогою формул (3.1-3.4) можна розрахувати кількість рядків SQL-коду для операцій додавання та редагування при проведенні рефакторингу БД.

$$SizeAdded = \sum_{r=1}^{N_{CR}} Size_{CR_r}, \quad (3.12)$$

де N_{CR} – загальна кількість операцій додавання у запиті на зміни;

$$SizeModified = \sum_{r=1}^{N_{DR}} Size_{DR_r} + \sum_{r=1}^{N_{AL}} Size_{AL_r} + \sum_{r=1}^{N_{TR}} Size_{TR_r}, \quad (3.13)$$

де N_{DR} – загальна кількість операцій видалення у запиті на зміни;

N_{AL} – загальна кількість операцій редагування у запиті на зміни;

N_{TR} – загальна кількість операцій очищення у запиті на зміни.

На основі розроблених формул (3.12 - 3.13) для розрахунку $SizeAdded$ та $SizeModified$ їх необхідно підставити у формулу (3.8). Таким чином розрахунок кількості строк SQL-коду, якими буде реалізовано запити на зміни, буде мати наступний вигляд:

$$(Size)_M = \begin{pmatrix} \sum_{r=1}^{N_{CR}} Size_{CR_r} + \sum_{r=1}^{N_{DR}} Size_{DR_r} + \\ + \sum_{r=1}^{N_{AL}} Size_{AL_r} + \sum_{r=1}^{N_{TR}} Size_{TR_r} \end{pmatrix} \times MAF, \quad (3.14)$$

Розроблені формальні описи оцінки обсягу коду для сукупності запитів на зміни ІС, яка експлуатується, дозволяють оцінити кількість строк SQL-коду на проведення рефакторингу БД. В основу запропонованих оцінок покладено параметричну модель COCOMO2 для розрахунку трудовитрат. Використання такого уніфікованого формального опису дозволяє в подальшому використовувати, для автоматизованого розрахунку значень цих оцінок, одні й ті ж процедури.

На основі розроблених формальних оцінок було розроблено розрахунок показника кількісного оцінювання обсягу змін, які слід внести під час рефакторингу БД. Визначено формулу (3.14) для розрахунку цього показника.

Слід зазначити, що отримані результати дозволяють визначити обсяг трудовитрат та витрат часу для написання коду на проведення рефакторингу БД на основі запитів на зміни.

3.4 Висновки до третього розділу

За результатами розробки моделі і методу управління ІТ-проєктом рефакторингу БД Інтернет-провайдера, а саме: опису SQL-команд; опису оцінки трудовитрат та витрат часу на проведення рефакторингу БД за моделлю COCOMO2 та розрахунку обсягу коду при проведенні рефакторингу БД – було проведено дослідження особливостей реалізації моделі і методу для управління ІТ-проєктом рефакторингу БД Інтернет-провайдера.

По-перше, на основі дослідження сценарію використання методу і моделі управління ІТ-проєктом рефакторингу БД Інтернет-провайдера, було отримано список SQL-команд, які будуть використані для розрахунку оцінки трудовитрат та витрат часу на проведення рефакторингу БД Інтернет-провайдера, а саме: команда створення (create), команда видалення (drop), команда редагування (alter) та команда очищення (truncate).

По-друге, було проаналізовано параметричну модель COCOMO2, описано її використання для розрахунку оцінки трудовитрат та витрат часу на проведення рефакторингу БД Інтернет-провайдера та виявлено проблему цієї моделі, яка полягає в тому, що вона не надає формули для розрахунку $SizeAdded$ та $SizeModified$, які необхідні для розрахунку оцінки трудовитрат та витрат часу при модифікації системи.

По-третє, було розроблено формальні описи оцінки обсягу коду для сукупності запитів на зміни ІС, яка експлуатується.

Вони дозволяють оцінити трудовитрати на проведення рефакторингу БД. В основу запропонованих оцінок покладено параметричну модель COCOMO2 для розрахунку трудовитрат. Використання такого уніфікованого формального опису дозволяє в подальшому використовувати, для автоматизованого розрахунку значень цих оцінок, одні й ті ж процедури.

4 ПРАКТИЧНА АПРОБАЦІЯ РЕЗУЛЬТАТІВ МАГІСТЕРСЬКОЇ РОБОТИ

4.1 Аналіз предметної галузі, що визначається діяльністю фірми Інтернет-провайдера

За завданням кваліфікаційної роботи розглядається діяльність фірми-провайдера, яка надає послуги доступу до глобальної мережі Інтернет.

Клієнтом Інтернет-провайдера може бути як фізична, так і юридична особа (фірма, компанія тощо). Провайдер надає доступ до сервісів, перенаправляючи Інтернет-трафік клієнтів через власні сервери. Основною метою роботи провайдерів є надання стабільного доступу до сервісів та забезпечення захисту й цілісності даних. Організаційна структура фірми Інтернет-провайдера, що розглядається, подана на рис. 4.1.

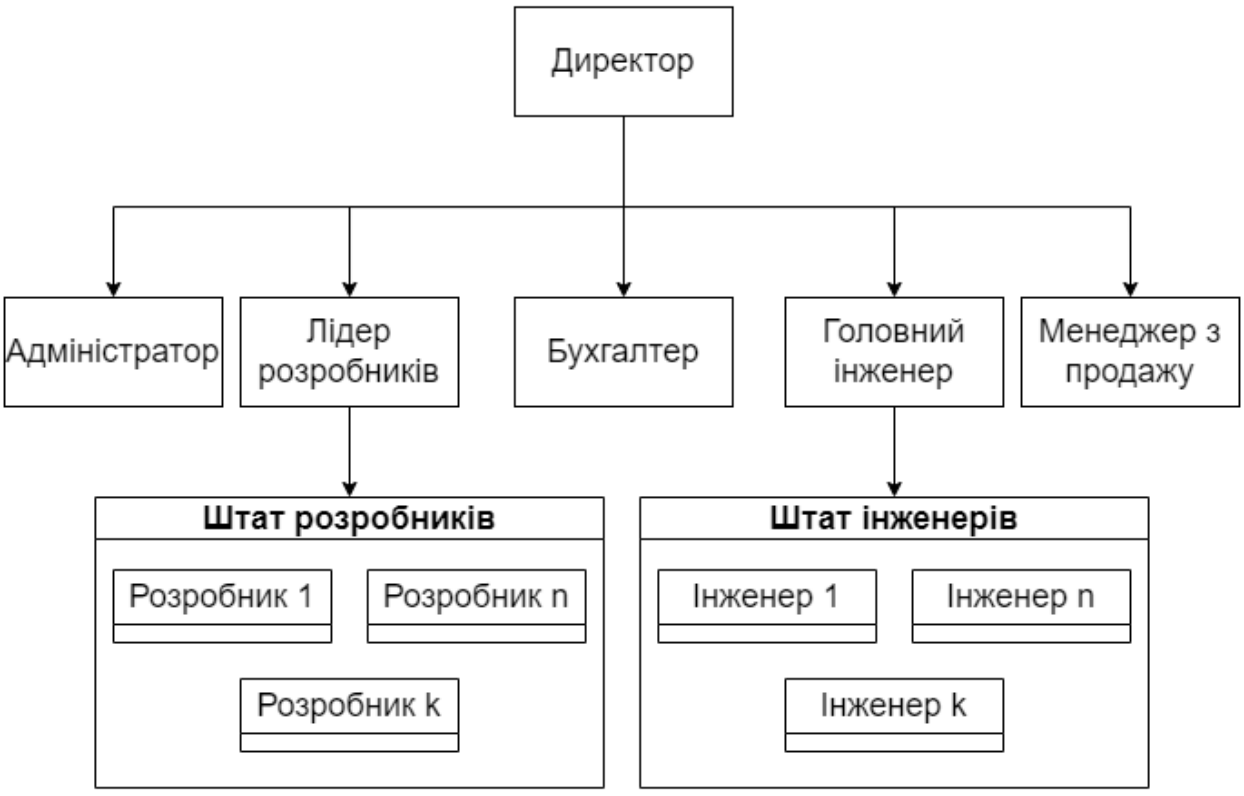

Рисунок 4.1 – Схема організаційної структури фірми Інтернет-провайдера

Діяльність фірми Інтернет-провайдера підпорядковується таким нормативним актам України як закон України «Про електронні комунікації» [22], закон України «Про затвердження Правил здійснення діяльності у сфері телекомунікацій» [23] та закон України «Про захист персональних даних» [24].

Директор – засновник компанії, він керує загальним процесом, стежить за успішністю фірми Інтернет-провайдера, приймає рішення щодо подальшого розвитку Інтернет-провайдеру. Йому безпосередньо підпорядковуються бухгалтер, менеджер з продажу, адміністратор та інженери.

Менеджер з продажу відповідає за облік послуг, переглядає статистику, встановлює які послуги більш популярні та які треба деактивувати.

Адміністратор – стежить за ІС, за якістю виконання роботи в терміни які задані керівництвом. Керує штатом розробників системи та виступає їх менеджером.

Лідер розробників – людина, яка керує штатом розробників та виконує роль аналітика.

Штат розробників – люди, які розробляють новий функціонал в ІС, додають нові послуги, стежать за якістю та працездатністю системи.

Бухгалтер – відповідає за організацію та ведення бухгалтерського та податкового обліку фірми Інтернет-провайдера.

Головний інженер – людина, яка підпорядковується директору фірми, керує та відповідає за штат інженерів.

Штат інженерів – люди, які займаються ручною працею, тобто виїжджають на виклики клієнтів, ремонтують та слідкують за обладнанням.

Кількість ресурсів і сервісів Інтернет постійно збільшується. Тому провайдери надають клієнтам послугу у вигляді комплексного рішення, що стосується доступу до ресурсів і сервісів. До таких послуг відносяться: доступ мережі Інтернет за допомогою комутованих ліній надійного зв'язку (телефонним або мережевим); надання дискового простору для зберігання та забезпечення роботи сайтів (хостинг); підтримка сервісів електронних

поштових скриньок або віртуального поштового сервера; розташування обладнання клієнтів на власних технічних потужностях та лініях зв'язку (колокація); надання в оренду відокремлених та віртуальних серверів, резервування даних тощо.

На поточний час дані інформаційно-облікової системи фірми Інтернет провайдера зберігаються у реляційній БД на платформі MySQL. Схема даних інформаційно-облікової системи Інтернет-провайдера у вигляді ER-моделі подана на рис. 4.2.

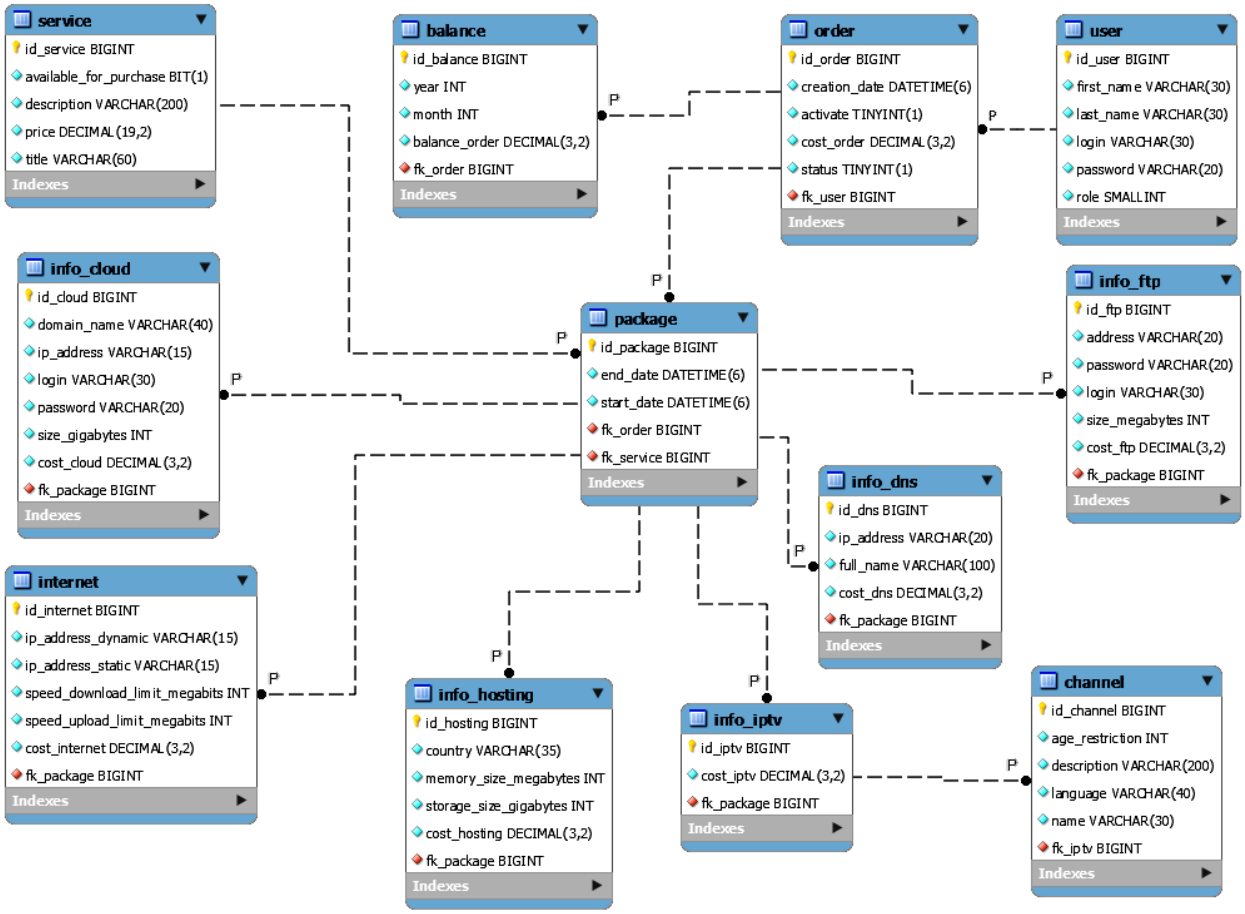

Рисунок 4.2 – ER-модель БД інформаційно-облікової системи Інтернет-провайдера

Для поточної БД були визначені усі типи даних полів таблиць та зв’язки між таблицями.

Визначена така посилальна цілісність зв’язків між таблицями:

– зв’язок між таблицею «service» (первинний ключ id_service) та «package» (зовнішній ключ fk_service) для операції delete – restrict, а для операції update – restrict;

– зв’язок між таблицею «order» (первинний ключ id_order) та «balance» (зовнішній ключ fk_order) для операції delete – restrict, а для операції update – restrict;

– зв’язок між таблицею «order» (первинний ключ id_order) та «package» (зовнішній ключ fk_order) для операції delete – cascade, а для операції update – restrict;

– зв’язок між таблицею «user» (первинний ключ id_user) та «order» (зовнішній ключ fk_user) для операції delete – restrict, а для операції update – restrict;

– зв’язок між таблицею «package» (первинний ключ id_package) та «info_ftp» (зовнішній ключ fk_package) для операції delete – restrict, а для операції update – restrict;

– зв’язок між таблицею «package» (первинний ключ id_package) та «info_dns» (зовнішній ключ fk_package) для операції delete – restrict, а для операції update – restrict;

– зв’язок між таблицею «package» (первинний ключ id_package) та «info_iptv» (зовнішній ключ fk_package) для операції delete – restrict, а для операції update – restrict;

– зв’язок між таблицею «info_iptv» (первинний ключ id_iptv) та «channel» (зовнішній ключ fk_iptv) для операції delete – restrict, а для операції update – restrict;

– зв’язок між таблицею «package» (первинний ключ id_package) та «info_hosting» (зовнішній ключ fk_package) для операції delete – restrict, а для операції update – restrict;

– зв’язок між таблицею «package» (первинний ключ id_package) та «internet» (зовнішній ключ fk_package) для операції delete – restrict, а для операції update – restrict;

– зв’язок між таблицею «package» (первинний ключ id_package) та «info_cloud» (зовнішній ключ fk_package) для операції delete – restrict, а для операції update – restrict.

Наведена схема даних дозволить кількісно оцінити обсяг поточної БД та використовувати цю оцінку для оцінки трудовитрат, витрат часу та кількісного оцінювання запитів на зміни на проведення рефакторингу.

ІС Інтернет-провайдера на поточний час має недоліки, які планується виправити за рахунок проведення рефакторингу БД, а саме: гнучке додавання нової послуги та відсутність функціоналу гнучкого створення та зберігання замовлень. З цього випливає, що користувачі системи Інтернет-провайдеру не задоволені тим, що вони не мають можливості самостійного гнучкого налаштування послуг тому, що їх налаштування вже визначені. Послуги повинні бути різноманітними та підлаштовуватися під змінення ринку.

4.2 Опис запитів на зміни, які надійшли до початку рефакторингу

Відповідно до третього розділу запити на зміни описуються лідером розробників у форматі UserStory з текстовим описом.

UserStory «Створення замовлення. Користувач» – бажаний стан.

Як Користувач, я хочу замовити послугу для того, щоб нею користуватися.

Критерії приймання:

Враховуючи, що послуга існує.

Коли обираю послугу в системі.

Тоді система надає можливість налаштувати параметри пакету.

Після формування вищевказаного текстового опису лідер розробників створює UserStory для цього запиту на зміни (рисунок 4.3).

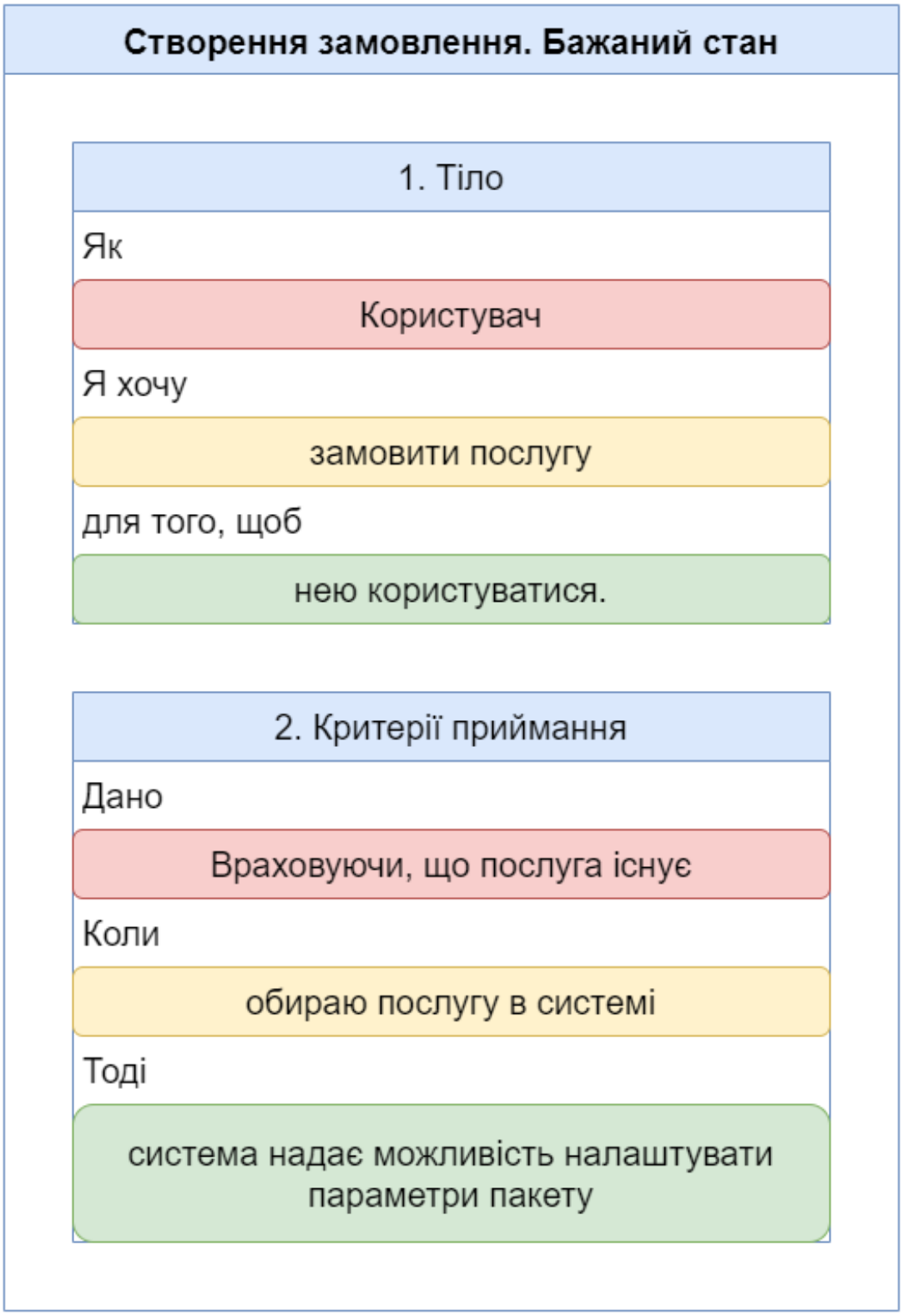

Рисунок 4.3 – UserStory запиту на зміну «Створення замовлення»

Лідер сумісно з командою розробників, проаналізувавши опис, наведений вище, створюють ER-модель схеми даних для запиту на зміни (рисунок 4.4).

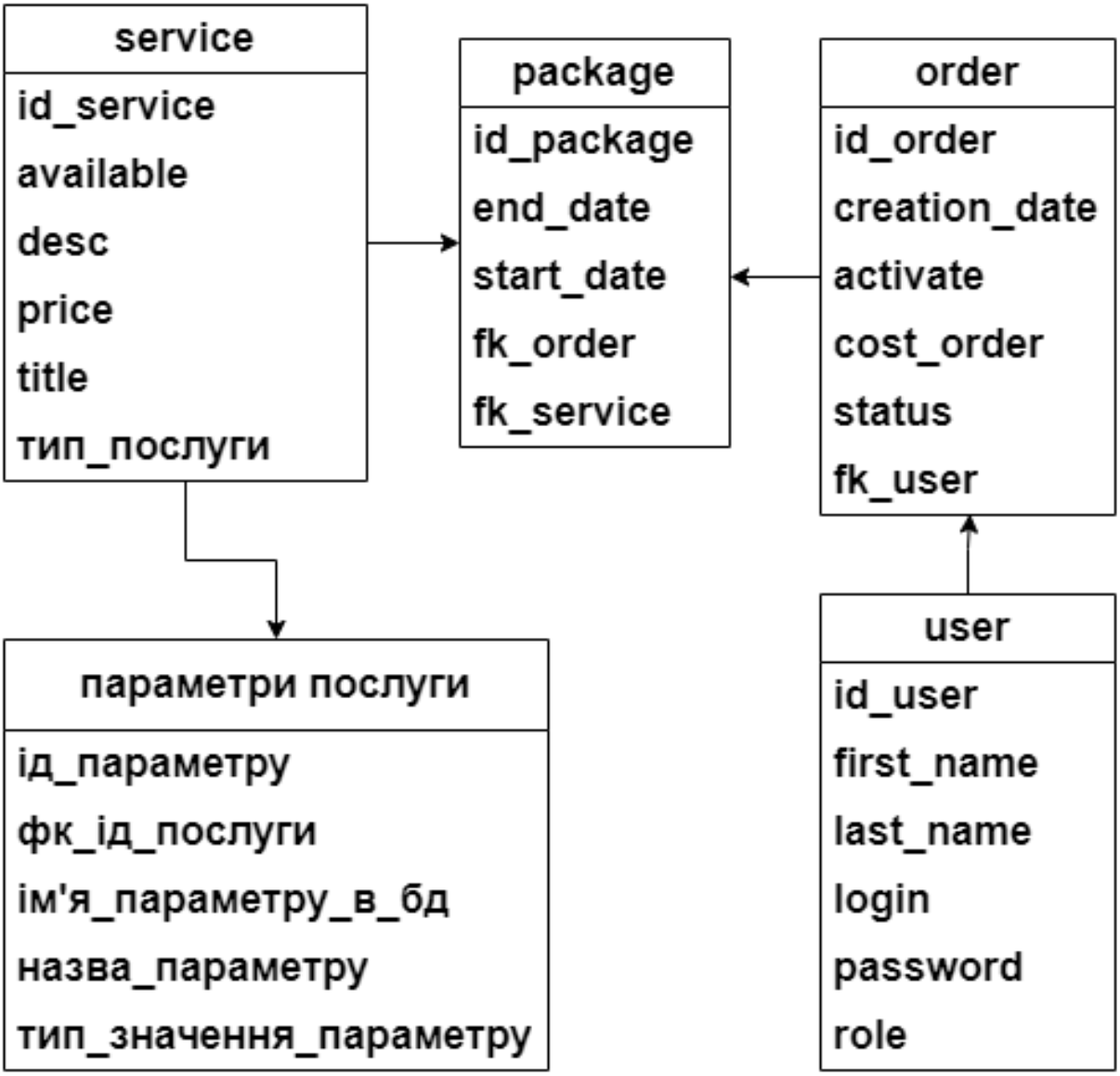

Рисунок 4.4 – ER-модель схеми даних для запиту на зміни «Створення замовлення»

На рисунку 4.4 українською мовою позначено таблиці та поля, що додаються запитом на зміну, англійською мовою позначено таблиці та поля актуальної БД.

Наступний запит на зміни – це функція обліку розподілу користувачів за населеним пунктом. Він формується наступним чином.

UserStory «Облік користувачів за населеним пунктом» – бажаний стан.

Як Адміністратор, я хочу відстежувати розподіл користувачів за населеним пунктом для того, щоб надавати актуальні послуги та відстежувати популярність послуг.

Критерії приймання:

Враховуючи, що клієнт ще не зареєстрований у системі.

Коли клієнт реєструється в системі.

Тоді система просить обрати населений пункт та вказати фактичну адресу проживання.

Після формування вищевказаного текстового опису лідер розробників створює UserStory для цього запиту на зміни (рисунок 4.5).

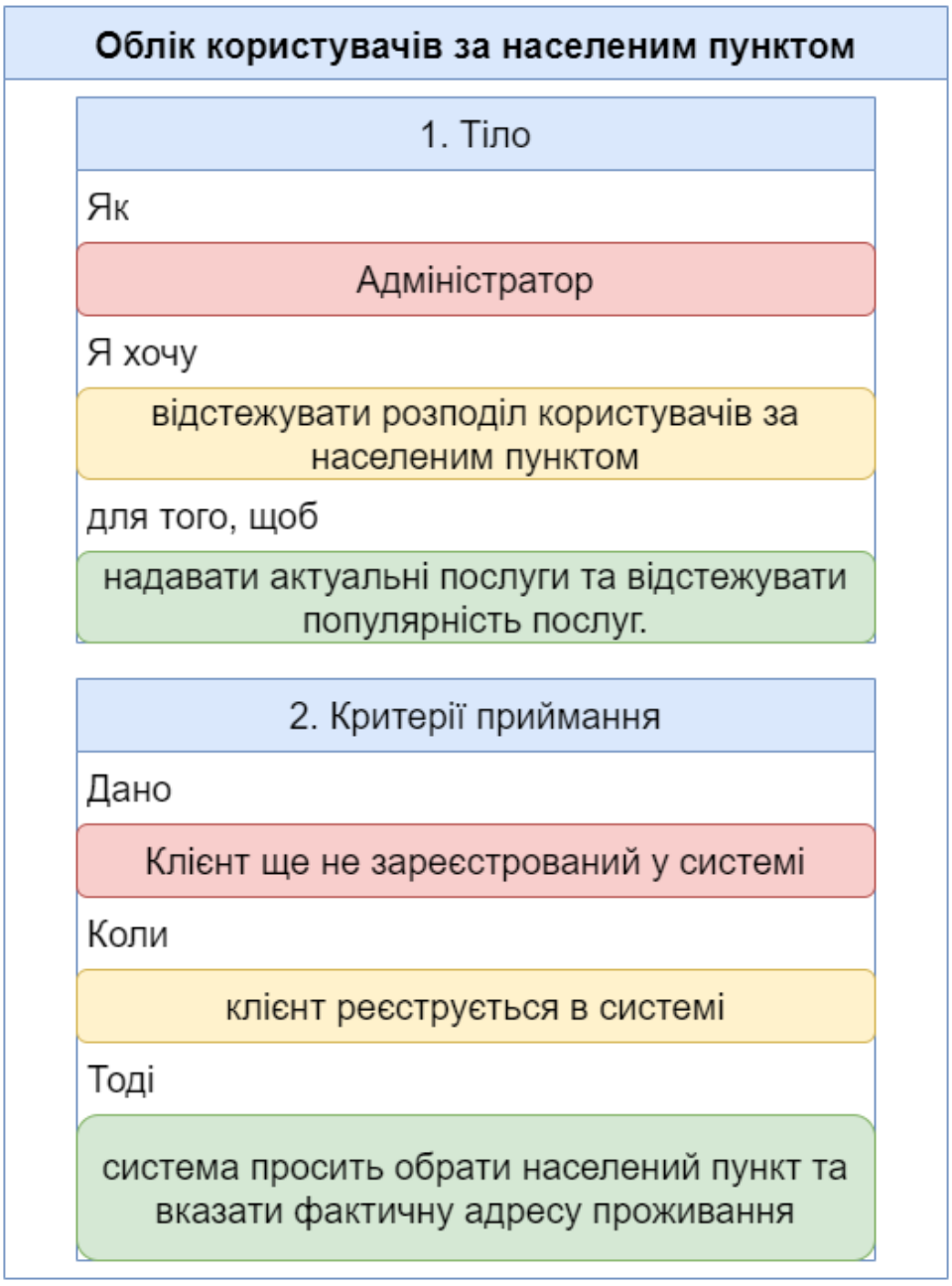

Рисунок 4.5 – UserStory запиту на зміну «Облік розподілу користувачів за населеним пунктом»

Лідер розробників спільно з командою, проаналізувавши опис, наведений вище, створюють ER-модель схеми даних для запиту на зміни (рисунок 4.6).

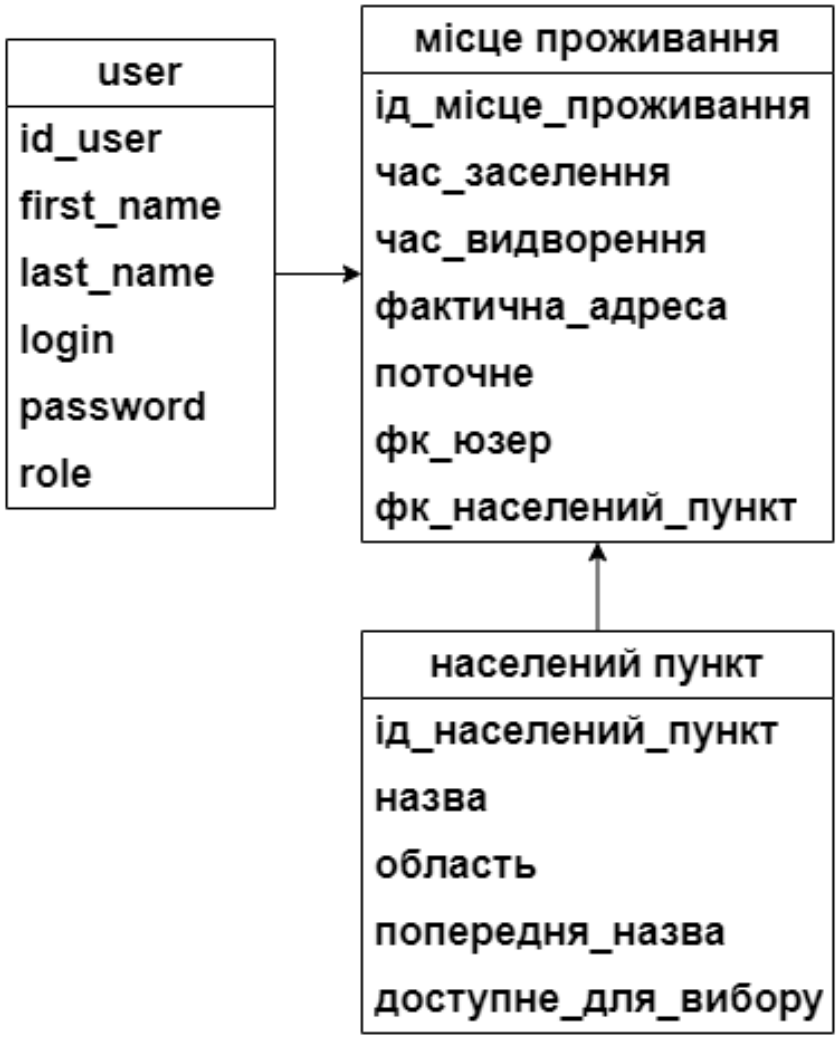

Рисунок 4.6 – ER-модель схеми даних для запиту на зміни «Облік розподілу користувачів за населеним пунктом»

На рисунку 4.6 українською мовою позначено таблиці та поля, що додаються запитом на зміну, англійською мовою позначено таблиці та поля актуальної БД.

Отже, результатом опису запитів на зміни є наступне: описано функції «Створення замовлення» та Облік розподілу користувачів за населеним пунктом» системи у бажаному стані. Створено UserStory для кожного запиту. Окрім цього, побудовано ER-моделі схеми даних для кожного запиту на зміни.

4.3 Опис процесу прийняття рішення щодо доцільності проведення рефакторингу бази даних

4.3.1 Оцінка доцільності проведення рефакторингу бази даних

За формулою (2.22) розрахунок кількісної оцінки складності сутностей, що утворюють актуальну БД Інтернет-провайдеру має наступний вигляд:

$$C_{DB_E} = \sum_{i=1}^{12} \left|\langle atr_{E_i}^{j} \rangle\right| = 5 + 5 + 6 + 6 + 8 + 5 + 7 + 5 + 7 + 6 + 3 + 6 = 69, \quad (4.1)$$

За формулою (2.23) розрахунок кількісної оцінки складності зв'язків, які наявні між сутностями актуальної БД Інтернет-провайдеру має наступний вигляд:

$$C_{DB_R} = \sum_{i=1}^{11} \left|Atr_{E_m}^{R_i}\right| + \sum_{i=1}^{11} \left|Atr_{E_n}^{R_i}\right| = 11 + 11 = 22, \quad (4.2)$$

Отже, на основі розрахунків (4.1 та 4.2) розрахунок кількісної оцінки складності актуальної БД (2.24) Інтернет-провайдеру має наступний вигляд:

$$C_{DB} = \sum_{i=1}^{12} \left|\langle atr_{E_i}^{j} \rangle\right| + \left(\sum_{i=1}^{11} \left|Atr_{E_m}^{R_i}\right| + \sum_{i=1}^{11} \left|Atr_{E_n}^{R_i}\right|\right) = 69 + 22 = 91, \quad (4.3)$$

Загальна кількісна оцінка складності актуальної БД (4.1) Інтернет-провайдеру становить 91.

Кількісна оцінка складності першого запиту на зміни розраховується за формулою (2.32) та має вигляд:

$$C_{RFC_1} = \left|\begin{matrix}(\text{тип\_послуги, ід\_параметру, фк\_ід\_послуги,} \\ \text{ім'я\_параметру\_в\_бд, назва\_параметру} \\ \text{тип\_значення\_параметру})\end{matrix}\right| + \\ +(|\emptyset| + |(\text{фк\_ід\_послуги})|) = 6 + 0 + 1 = 7, \quad (4.4)$$

Для уникнення парадоксу, при якому кількість елементів у порожній множині дорівнює одиниці, враховується наступне: якщо в результаті віднімань виходить порожня множина, то ми вважаємо, що кількість атрибутів, що беруть участь в утворенні зв'язку та введених нами, дорівнює нулю.

За аналогією з виразом (4.4), розрахунок кількісної оцінки складності другого запиту на зміни розраховується за формулою (2.32) та має вигляд:

$$C_{RFC_2}\left|\begin{array}{c}(\text{ід\_місце\_проживання}, \text{час\_заселення}, \text{час\_видворення},\\ \text{фактична\_адреса}, \text{поточне}, \text{фк\_юзер}, \text{фк\_населений\_пункт},\\ \text{ід\_населений\_пункт}, \text{назва}, \text{область}, \text{попередня\_назва},\\ \text{доступне\_для\_вибору})\end{array}\right| +$$

$$+(|(\text{ід\_населений\_пункт})| + |(\text{фк\_юзер}, \text{фк\_населений\_пункт})|) =$$

$$= 12 + 1 + 2 = 15, \quad (4.5)$$

Враховуючи, що запитів на зміни може бути декілька – пропонується враховувати одразу кілька запитів на зміни у формулі кількісної оцінки рефакторингу:

$$C_{REF} = \sum_{i=1}^{K} C_{RFC_i} \Big/ C_{DB}, \quad (4.6)$$

де K – загальна кількість запитів на зміни.

Отже, розрахунок фактичної кількісної оцінки складності проведення рефакторингу БД Інтернет-провайдеру за формулою (4.6) має наступний вигляд:

$$C_{REF} = \frac{7+15}{91} = \frac{22}{91} = 0.24, \quad (4.7)$$

За результатами проведених розрахунків (4.1-4.7) кількісна оцінка складності проведення рефакторингу БД Інтернет-провайдеру становить 0,24. Ця оцінка не перевищує, визначену у другому розділі, верхню межу доцільності проведення рефакторингу 0,5. Тому пропонується зробити висновок про те, що процес проведення рефакторингу БД інформаційно-облікової системи Інтернет-провайдеру вважається доцільним.

4.3.2 Оцінка трудовитрат і витрат часу на проведення рефакторингу

Розрахунок оцінки трудовитрат та витрат часу необхідно почати з розрахунку $SizeAdded$ та $SizeModified$ за формулами (3.12 та 3.13 відповідно).

Розрахунок $SizeAdded$ для першого запиту на зміни має вигляд:

$$SizeAdded_1 = \sum_{r=1}^{1} Size_{CR_r} = (5 + 4) = 9. \quad (4.8)$$

Розрахунок $SizeModified$ для першого запиту на зміни має вигляд:

$$SizeModified_1 = \sum_{r=1}^{1} Size_{AL_r} = (1 + 1) = 2. \quad (4.9)$$

Розрахунок $SizeAdded$ для другого запиту на зміни має вигляд:

$$SizeAdded_2 = \sum_{r=1}^{2} Size_{CR_r} = (7 + 4) + (5 + 4) = 20. \quad (4.10)$$

Другий запит на зміни складається лише з операцій додавання.

Інформаційно-облікова система Інтернет-провайдеру має наступні характеристики: високий рівень інтеграції, слабкий ступінь зв’язкості

елементів, добру кореляцію між програмою і додатком, добрий рівень присутності в коді коментарів та заголовків; корисну документаці., але є окремі слабкі місця. Тоді, штраф, викликаний витратами на розуміння ПЗ дорівнює 20. Команда проєкту повністю згуртована.

Отже, розрахунок коригуючого коефіцієнту проведення рефакторингу БД має наступний вигляд:

$$MAF = 1 + \left(\frac{SU}{100} \times UNFM\right) = 1 + \left(\frac{20}{100} \times 0\right) = 1 + 0 = 1, \qquad (4.11)$$

Для розрахунку обсягу модифікацій продукту необхідно підставити результати (4.11) у формулу (3.14):

$$(Size)_M = (9 + 2 + 20) \times 1 = 31, \qquad (4.12)$$

Оскільки запити на зміни не змінювалися під час їх надходження та аналізу, коефіцієнт $BRAK$ буде дорівнювати 0. Тоді, кількість рядків вихідного програмного коду розраховується за формулою (3.6) та має вигляд:

$$Size' = 31 \times \left(1 + \frac{BRAK}{100}\right) = 31 \times \left(1 + \frac{0}{100}\right) = 31, \qquad (4.13)$$

Виходячи з того, що подібний проєкт раніше виконували, є незначні відхилення у гнучкості, специфікації елементів задані на 100%, команда має повну взаємодію та фірма Інтернет-провайдеру має перший рівень зрілості, розрахунок показнику ступеня E (3.7) має наступний вигляд:

$$E = 0{,}91 + 0{,}01 \times \sum_{j=1}^{5} SF_j =$$
$$= 0{,}91 + 0{,}01 \times (0{,}03 + 0{,}02 + 0{,}05 + 0{,}05 + 0{,}01) =$$
$$= 0{,}91 + 0{,}01 \times 0{,}16 = 0{,}9116, \qquad (4.14)$$

Розмір БД проєкту маленький, складність продукту та акцент на надійність, документацію незначні. Повторне використання коду дозволено в рамках продукту. Платформа незначно змінна. Мінливість кадрів за рік менше 4 відсотків, програмісти мають гарні навички та досить довго працюють над проєктом. Також кожен розробник має мінімум 4 роки досвіду з розробки. Всі члени команди знаходяться в рамках одного будинку та мають задоволену кількість засобів електронної комунікації. Дотримання розкладу проєкту дорівнює приблизно 85%. Тоді, множина витрат розраховується наступним чином:

$$\prod_{i=1}^{7} EM_i = 0{,}93 \times 1{,}14 \times 1{,}11 \times 0{,}7 \times 0{,}75 \times 0{,}78 \times 1{,}1 = 0{,}53, \quad (4.15)$$

Тепер необхідно підставити результати (4.13-4.15) у формулу (3.5) та розрахувати трудовитрати:

$$PM = 2{,}94 \times [0{,}031]^{0{,}9116} \times 0{,}53 = 2{,}94 \times 0{,}042 \times 0{,}53 = 0{,}065, \quad (4.16)$$

Використовуючи результати формули (4.14), розрахунок показника масштабування для рівняння розкладу (3.11) має вигляд:

$$F = (0{,}28 + 0{,}2 \times [0{,}9116 - 0{,}91]) = 0{,}28032, \quad (4.17)$$

Необхідне стиснення розкладу дорівнює 110. Тоді, необхідно підставити результати (4.16-4.17) у формулу (3.10) для розрахунку витрат часу:

$$TDEV = [3{,}67 \times (0{,}065)^{0{,}28032}] \times \frac{110}{100} = 1{,}876, \quad (4.18)$$

За результатами проведених розрахунків (4.16 та 4.18) кількісна оцінка трудовитрат на проведення рефакторингу БД Інтернет-провайдеру становить 0,065 людино-місяців. Оцінка витрат часу становить 1,876 місяців.

Ці оцінки будуть використані командою розробників для подальшого прийняття остаточного рішення щодо проведення рефакторингу БД.

4.4 Висновки з четвертого розділу кваліфікаційної роботи

В процесі практичної апробації магістерської роботи було зроблено наступне.

По-перше, проаналізовано предметну галузь, результатами цього є схема організаційної структури об’єкту автоматизації з визначеними обов’язками для кожної ланки структури. Також проаналізовано актуальну БД Інтернет-провайдера, наведено її схему у вигляді ER-моделі та визначено посилальну цілісність.

По-друге, проведено формальний опис запитів на зміни, які надійшли до початку рефакторингу у текстовому вигляді, до них створено UserStory та побудовано ER-моделі схеми даних для кожного запиту на зміни.

По-третє, проведено розрахунки оцінок доцільності проведення рефакторингу БД, трудовитрат та витрат часу на проведення рефакторингу БД Інтернет-провайдера.

За результатами практичної апробації отримано наступне: кількісна оцінка складності проведення рефакторингу БД Інтернет-провайдеру дорівнює 0,24 та не перевищує верхню межу доцільності проведення рефакторингу 0,5, оцінка трудовитрат дорівнює 0,065 людино-місяців та кількість часу на проведення рефакторингу становить 1,876 місяців.

Отже, в розділі було проведено практичну апробацію магістерської роботи. Як результат розділу є висновок, що процес проведення рефакторингу БД інформаційно-облікової системи Інтернет-провайдеру вважається доцільним.

ВИСНОВКИ

За завданням магістерської кваліфікаційної роботи було проведено дослідження моделей і методів проведення рефакторингу БД, розроблено модель оцінювання складності проведення рефакторингу БД та отримано подальший розвиток моделі оцінювання трудовитрат COCOMO2 для IT-проєктів рефакторингу БД.

У рамках магістерської кваліфікаційної роботи було доведено актуальність дослідження, проаналізовано існуючі моделі, методи та рішення проведення рефакторингу, обґрунтовано мету розробки моделі оцінювання трудовитрат та витрат часу на проведення рефакторингу, яка розглядається в цій роботі, поставлено задачу дослідження. Робота оформлена згідно з стандартами [25, 26].

На основі цих досліджень була розроблена модель оцінювання, а саме – описано етапи сценарію виконання бізнес-процесу проведення рефакторингу, розроблено кількісну оцінку складності проведення рефакторингу БД, адаптовано модель оцінювання трудовитрат та витрат часу на проведення рефакторингу БД та проведено практичну апробацію цієї моделі на БД Інтернет-провайдера.

В якості практичної апробації моделі розраховано оцінку доцільності, трудовитрат та витрат часу на проведення рефакторингу БД Інтернет-провайдера для запитів на зміни.

Отже, усі завдання кваліфікаційної роботи виконані – дослідження моделей і методів управління IT-проєктом рефакторингу БД Інтернет-провайдера проведено, модель оцінювання створено.

Результати кваліфікаційної роботи було опубліковано в роботі [2].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методичні вказівки щодо розробки та оформлення магістерської атестаційної роботи за спеціальністю 122 Комп'ютерні науки (освітня програма «Управління проєктами в галузі інформаційних технологій» освітньо-кваліфікаційного рівня «магістр» / Упоряд.: Петров К.Е., Левикін В.М., Чалий С.Ф., Євланов М.В., Саєнко В.І., Міхнов Д.К., Міхнова А.В., Чала О.В. – Харків: ХНУРЕ, 2019. – 28 с.

2. Мірошниченко Д.О. Дослідження моделей і методів управління ІТ-проєктом рефакторингу бази даних Інтернет-провайдера задля підтримки гнучкого створення замовлень. //27-й Міжнародний молодіжний форум «Радіоелектроніка та молодь XXI століття». Зб. матеріалів форуму. Т.6. Конференція «Інформаційні інтелектуальні системи» – Харків: ХНУРЕ. 2023. – С. 96-97.

3. Рефакторинг коду: що потрібно про це знати – Тестування Вдосконалення Навичок. Огляди, Ігри, Розваги, March 2023. URL: https://uk.myservername.com/code-refactoring-what-you-need-know-about-it (дата звернення: 07.03.2023).

4. Ambler S. W., Sadalage P. Refactoring Databases: Evolutionary Database Design. Addison-Wesley Longman, Incorporated, 2006. 384 с.

5. Струзік В. А., Грибков С. В., Литвин А. О. Дослідження методів і підходів проведення рефакторингу БД. Науковій журнал "Комп'ютерно-інтегровані технології: освіта, наука, виробництво". Луцьк, 2018. № 30-31.

6. ISO/IEC/IEEE 15288:2015. Systems and software engineering – System life cycle processes. Вид. офіц. 2015. 108 с.

7. Everything you need to know about PRINCE2 project management. URL: https://asana.com/resources/prince2-methodology (дата звернення: 23.02.2023).

8. Mathis B. Prince2 for Beginners: Prince2 study guide for certification & project management., 2014. 140 c.

9. The Process of Database Refactoring: Strategies for Improving Database Quality. AgileData.org. URL: http://agiledata.org/essays/databaseRefactoring.html (дата звернення: 24.02.2023).

10. Data Quality Management Model (2015 Update) – Retired. HIM Body of Knowledge. URL: https://library.ahima.org/PB/DataQualityModel (дата звернення: 24.02.2023).

11. What is Database Change Management (DCM)?. Database CI/CD | Database DevOps | Bytebase. URL: https://www.bytebase.com/blog/what-is-database-change-management (дата звернення: 24.02.2023).

12. Refactoring Databases with SQL Prompt | Redgate. Redgate. URL: https://www.red-gate.com/hub/product-learning/sql-prompt/refactoring-databases-with-sql-prompt (дата звернення: 21.02.2023).

13. Lawrence S. DBmaestro Resources | Database DevOps | Database CICD. DBmaestro. URL: https://www.DBmaestro.com/resources (дата звернення: 02.03.2023).

14. FAQs ApexSQL Refactor. SQL Server tools for DBAs and Developers | ApexSQL. URL: https://www.apexsql.com/sql-tools-refactor/faq.aspx (дата звернення: 02.03.2023).

15. A Guide to the Project Management Body of Knowledge (PMBOK® Guide). Newtown Square, USA : Project Management Institute, 2021.

16. The Process of Database Refactoring: Strategies for Improving Database Quality // Agile Data. URL: http://agiledata.org/essays/databaseRefactoring.html (дата звернення: 12.04.2023).

17. Левикін В. М., Эвланов М. В., Сугробов В. С. Паралельне проектування інформаційного та програмного комплексів інформаційної системи // Радіотехніка. – 2006. – Вип. 146. – С. 89-98.

18. Petrov E.G., Chajnikov S.I., Ovezgeldyev A.O. Metodologiya strukturnogo sistemnogo analiza i proektirovaniya krupnomasshtabnyh IUS. Koncepcii i metody. Harkov: Rubikon, 1997. Chast 1. 140 s.

19. SQL | DDL, DQL, DML, DCL and TCL Commands - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/ (дата звернення: 12.04.2023).

20. SQL DDL Commands | Learn the DDL Commands of SQL. EDUCBA. URL: https://www.educba.com/sql-ddl-commands/ (дата звернення: 15.04.2023).

21. COCOMO II Model Definition Manual. Rose-Hulman Institute of Technology. URL: https://www.rose-hulman.edu/class/cs/csse372/201310/Homework/CII_modelman2000.pdf (дата звернення: 19.04.2023).

22. Про електронні комунікації : Закон України від 16.12.2020 р. № 1089-IX : станом на 31 берез. 2023 р. URL: https://zakon.rada.gov.ua/laws/show/1089-20 (дата звернення: 20.04.2023).

23. Про затвердження Правил здійснення діяльності у сфері телекомунікацій : Рішення Нац. коміс., що здійснює держ. регулювання у сфері зв'язку та інформатизації від 19.11.2019 р. № 541 : станом на 1 лип. 2022 р. URL: https://zakon.rada.gov.ua/laws/show/z1309-19 (дата звернення: 20.04.2023).

24. Про захист персональних даних : Закон України від 01.06.2010 р. № 2297-VI : станом на 27 жовт. 2022 р. URL: https://zakon.rada.gov.ua/laws/show/2297-17 (дата звернення: 20.04.2023).

25. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлювання. – Чинний від 22.06.2015. – Київ: ДП «УкрНДНЦ», 2016. – 31 с.

26. ДСТУ 8302:2015. Інформація та документація. Бібліографічні посилання. Загальні положення та правила складання. – Чинний від 04.03.2016. – Київ: ДП «УкрНДНЦ», 2016. – 20 с.