УДК 004.65:005.33 DOI: 10.20837/0135-1710.2023.179.004

М.В. ЄВЛАНОВ, І.О. ЮР'ЄВ, Д.О. МІРОШНИЧЕНКО

ОЦІНЮВАННЯ ВИТРАТ НА ПРОВЕДЕННЯ РЕФАКТОРИНГУ БАЗИ ДАНИХ ІНФОРМАЦІЙНОЇ СИСТЕМИ, ЯКА ЕКСПЛУАТУЄТЬСЯ

Розглянуто основні особливості прийняття рішень щодо доцільності здійснення рефакторингу бази даних. Встановлено, що подібні рішення рекомендується приймати виключно на основі інтуїції та досвіду особи, яка приймає рішення. Розроблені моделі оцінювання кількості рядків основних SQL-команд, які використовуються в процесі рефакторингу бази даних. Вдосконалено модель COCOMO II, яка дозволяє оцінити трудовитрати та витрати часу на проведення рефакторингу бази даних. Проведено експериментальну перевірку отриманих результатів.

1. Вступ

Актуальне визначення, запропоноване М. Фаулером, трактує термін «рефакторинг» як зміну внутрішньої структури програмного забезпечення з метою зробити його легшим для розуміння та дешевшим для модифікації без зміни його спостережуваної поведінки [1]. Суть рефакторингу за М. Фаулером полягає у серії невеликих трансформацій, що зберігають поведінку програмної системи. Кожна така трансформація вносить незначні зміни до структури системи, але послідовність цих трансформацій може призвести до значної реструктуризації. Після кожного рефакторингу працездатність системи повністю підтримується, що зменшує ймовірність того, що система може отримати серйозний злам під час реструктуризації, що зменшує ймовірність того, що система може отримати серйозний злам під час реструктуризації [1].

Серед розмаїття методів рефакторингу особливо виділяють рефакторинг бази даних (БД). Даний різновид рефакторингу у [2] визначається як розвиток існуючої схеми БД шляхом одноразового внесення невеликих змін з метою підвищення якості проєкту БД без зміни її семантики. Рефакторингу можуть бути піддані або структурні аспекти схеми БД (визначення таблиць, представлень тощо), або функціональні аспекти (процедури, що зберігаються, тригери тощо). Під час проведення рефакторингу схеми БД доводиться не тільки перевизначати саму схему, а й вносити зміни у зовнішні системи, такі як бізнес-додатки чи засоби вибірки даних, які прив'язані до конкретної схеми [2].

Сучасні дослідження в галузі рефакторингу спрямовані, переважно, на вирішення прикладних питань з організації та особливостей виконання окремих дій під час проведення рефакторингу. Для рефакторингу БД вирішення цих питань привело до створення Скоттом Амблером та його колегами прикладної методики проведення рефакторингу БД [2, 3]. Вона включає в себе набір процедур та практик, які допомагають безпечно та ефективно проводити рефакторинг БД у рамках проєкту розробки програмного забезпечення систем різного призначення.

Використання цієї методики передбачає проведення усіх операцій з рефакторингу БД тільки в тому випадку, якщо на початку процесу приймається рішення щодо доцільності проведення такого рефакторингу. Робота з формування та прийняття цього рішення у [2] розглядається як послідовний пошук відповідей на такі питання:

а) «Чи має сенс операція рефакторингу, яка пропонується?»;

б) «Чи повинні зміни бути виконані негайно?»;

в) «Чи виправдовує результат витрачені зусилля?».

Для вирішення цих питань у [2] пропонується використовувати інтуїцію та досвід, набуті особою, що приймає ці рішення, під час професійної діяльності в галузі проєктування і розробки БД.

Ця рекомендація залишається актуальною й досі. Так, проведення рефакторингу БД в умовах використання Agile-методологій для управління відповідним IT-проєктом рекомендується починати з визначення необхідності і доцільності проведення рефакторингу БД. Після цього рекомендується оцінити ймовірність того, що зміни дійсно потрібні. Ці визначення та оцінювання повинні виконуватися з використанням «інтуїції», яка виникає на основі попереднього досвіду роботи інженера з обробки даних Agile з розробником програми [3]. Але в таких випадках рішення про доцільність рефакторингу БД є значною мірою суб'єктивним і сильно залежить від тієї особи, яка приймає це рішення у кожному конкретному IT-проєкті. Моделі і методи, які дозволяють приймати об'єктивне рішення про доцільність проведення рефакторингу БД, залишаються майже недослідженими.

2. Аналіз літературних даних і постановка проблеми дослідження

Аналіз когнітивної структури досліджень з рефакторингу та розвитку цих досліджень з часом був проведений у [4]. Результати цього аналізу підтверджують, що дослідження з рефакторингу залишалися в переліку головних тем наукових публікацій в період з 2001 р. до 2014 р. Але, починаючи з 2005 р., головним фактором об'єднання публікацій досліджень в галузі рефакторингу у тематичні групи стали Agile-методології управління IT-проєктами. Ця тенденція спостерігалася до 2009 р., після чого головними групами досліджень з рефакторингу стають «Experimentation» і (меншою мірою) «Software-Evolution» і «Specific-Refactorings» [4].

Тенденціями розвитку досліджень з рефакторингу, починаючи з 2015 р., у [4] вважалися:

а) поява значної кількості вузькоспеціалізованих тем досліджень, що, ймовірно, пов'язано зі збільшенням використання інструментів рефакторингу, недоступних у минулому;

б) присутність тематики «Agile methodologies» серед рушійних (тобто, перспективних) тем у дослідницькій галузі рефакторингу, хоча й не в такому обсязі, як у 2005-2009 рр.

Цей прогноз значною мірою виправдався. Протягом трьох наступних років з'являються дослідження, присвячені вузькоспеціалізованим питанням рефакторингу БД. Так, у [5] досліджувалися наслідки прикладних робіт з рефакторингу БД системи оптимізації логістичних процесів. Результати досліджень були сформульовані у вигляді п'яти ключових уроків, які були присвячені виключно практичним аспектам і не давали нічого нового теорії комп'ютерних наук і, зокрема, рефакторингу БД. У [6] досліджувалися питання допомоги адміністраторам БД у діагностиці антишаблонів SQL і пропонувалися методи рефакторингу для вирішення антишаблонів. Було встановлено, що виявлення антишаблонів SQL все ще значною мірою залежить від семантики даних, а інструмент виявлення слід використовувати напівавтоматично, тобто він може вказувати на потенційно проблемні місця в схемі БД, які потребують подальшої діагностики адміністратором цієї БД. Ці результати дослідження мали також виключно прикладний характер.

Слід також зазначити появу дослідницьких робіт, присвячених розробці інструментальних засобів рефакторингу БД. Так, у [7] було запропоновано метод автоматичного синтезу нової версії БД з урахуванням її оригінальної версії, вихідної та цільової схем. Цей метод не вимагає ручного керування користувачем і гарантує, що

синтезована БД еквівалентна вихідній. Але запропонований у [7] метод не дозволяв вирішувати розглянуті вище питання доцільності проведення рефакторингу БД.

Окремо серед цієї множини дослідницьких робіт слід виділити роботу [8], в якій розглядається питання розробки методу реінжинірингу реляційних БД з урахуванням наявності неявних взаємопов'язаних функціонально залежних даних, які впливають на структуру логічної моделі. Одним з результатів, отриманих у [8], є дослідження спільного для задач адаптації та рефакторингу етапу формування цільової логічної схеми БД. Однак цей результат є цікавим з точки зору теорії організації виконання робіт з рефакторингу БД вже після того, як прийнято позитивне рішення щодо доцільності такого рефакторингу.

Сучасні дослідження також підтверджують цю тенденцію. Так, у [9] отримали значний розвиток питання пошуку та ліквідації антишаблонів SQL. У [10] розглядаються питання вдосконалення підходу до автоматизації рефакторингу одного з можливих антишаблонів, а саме антишаблону Unlimited Records. Одним з результатів цього вдосконалення є алгоритми виявлення та рефакторингу для двох різних підвипадків антишаблону Unlimited Records. Цікавішим з наукової точки зору є дослідження [11], в якому пропонується автоматизований підхід для визначення еквівалентності SQL-запитів та пов'язаних з ними програм до та після проведення рефакторингу. У цьому підході використовується спеціальне логічне моделювання SQL-запитів першого порядку, яке відповідає імперативний семантиці. У [12] результатом дослідження є розроблені методи такого специфічного різновиду рефакторингу, як рефакторинг БД для нормалізації. Але у всіх цих дослідженнях питання визначення доцільності проведення рефакторингу БД практично не розглядаються і не вирішуються.

Необхідність подолання розриву між рефакторингом як дослідженням і його впровадженням на практиці шляхом виділення загальних намірів рефакторингу, які більше підходять для того, з чим стикаються розробники в реальності, визначена у [13]. Результатом цього дослідження є, зокрема, визначення п'яти областей, у яких розробникам зазвичай потрібна допомога з рефакторингом: оптимізація коду, інструменти та IDE, архітектура та шаблони дизайну, модульне тестування та БД. Крім того, у [13] проведено детальне дослідження можливості і доцільності використання такого різновиду рефакторингу, як рефакторинг Stack Overflow, за допомогою серії кількісних і якісних експериментів. Але результати, отримані у [13], дають лише загальні відповіді щодо можливості та доцільності виконання рефакторингу БД. Проблема пошуку відповіді на питання «Чи виправдовує результат витрачені зусилля?» залишається нерозв'язаною.

Аналіз розглянутих публікацій дозволяє сформулювати висновок про те, що вирішення проблеми формування та прийняття рішення щодо доцільності проведення рефакторингу БД за останні роки набуває актуальності не тільки з теоретичної, а й з прикладної точки зору. Однак навіть загальні аспекти вирішення питань з визначення доцільності рефакторингу БД залишаються майже недослідженими. Між тим, вирішення цих питань хоча б на загальному рівні є актуальним не тільки для формування прикладних технік формування і прийняття подібних рішень, а й для управління IT-проєктами. Зокрема, проблема пошуку відповіді на питання «Чи виправдовує результат витрачені зусилля?» під час прийняття рішення щодо доцільності проведення рефакторингу БД стає особливо важливою, якщо проводити оцінювання витрат на рефакторинг БД як на різновид IT-проєкту. Тому проблему даного дослідження слід сформулювати як проблему проведення кількісного оцінювання витрат на рефакторинг БД як на різновид IT-проєкту.

3. Мета і задачі дослідження

Метою даного дослідження є вдосконалення параметричної моделі оцінювання трудовитрат на ІТ-проєкти COCOMO II для її використання під час оцінки витрат на рефакторинг БД. Застосування моделі COCOMO II хоча й відзначається підвищеною складністю, дозволяє отримати досить точні оцінки трудовитрат на виконання різних ІТ-проєктів. Тому досягнення цієї мети дозволить обґрунтовано прийняти рішення щодо доцільності проведення рефакторингу БД в умовах діючих обмежень ресурсів та часу виконання ІТ-проєкту такого рефакторингу.

Для досягнення цієї мети у статті вирішуються такі задачі:

– розробка моделей оцінювання кількості рядків вихідного коду SQL-команд, які використовуються під час проведення рефакторингу БД;

– вдосконалення елементів моделі COCOMO II для її застосування під час оцінювання трудовитрат на проведення рефакторингу БД;

– експериментальна перевірка результатів вдосконалення моделі COCOMO II.

4. Моделі, що використовуються для оцінки витрат на ІТ-проєкти

У ході дослідження пропонується використовувати моделі COCOMO II. Вибір даних моделей обумовлений можливістю їх використання не тільки для оцінювання трудовитрат на ІТ-проєкти створення програмних систем різного призначення, а й для оцінювання трудовитрат на модифікацію чи розвиток програмних систем, які експлуатуються.

Основою моделі COCOMO II є модель розрахунку трудовитрат у людино-місяцях (person months, PM), яка має загальний вигляд [14]:

$$PM = A \times [Size']^E \times \Pi_{i=1}^{7} EM_i + PM_M , \quad (1)$$

де A – числовий коефіцієнт, який для моделей COCOMO та COCOMO II дорівнює 2,94; $Size'$ – кількість рядків вихідного програмного коду, яка розраховується за такою формулою [14]:

$$Size' = Size \times \left(1 + \frac{BRAK}{100}\right); \quad (2)$$

$BRAK$ – показник, який відображує змінність вимог у ІТ-проєкті і являє собою відсоток коду, який видаляється з ІТ-проєкту в результаті зміни вимог; Е – показник ступеня, в який слід возвести значення $Size'$, розрахунок цього показника здійснюється за формулою [14]:

$$E = 0{,}91 + 0{,}01 \times \sum_{j=1}^{5} SF_j , \quad (3)$$

SF_j – драйвери масштабів ІТ-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 – драйвери витрат ІТ-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 COCOMO II, яка не використовується для формули зміни коду [14].

Для оцінки трудовитрат на модифікацію чи розвиток програмних систем, які експлуатуються, модель (1) була скоригована. Суть цієї корекції полягала у використанні замість елемента *Size* елемента $(Size)_M$, який розраховується за формулою [14]

$$(Size)_M = (SizeAdded + SizeModified) \times MAF, \tag{4}$$

де *SizeAdded* – кількість рядків вихідного коду, які було додано під час модифікації програмної системи; *SizeModified* – кількість рядків вихідного коду, які було змінено під час модифікації програмної системи; *MAF* – коригуючий коефіцієнт модифікації програмної системи, який розраховується за формулою [14]

$$MAF = 1 + \left(\frac{SU}{100} \times UNFM \right); \tag{5}$$

SU – штраф, викликаний витратами на розуміння програмного коду; *UNFM* – показник рівня відокремленості команди.

Значення драйверів масштабів ІТ-проєкту і драйверів витрат ІТ-проєкту наведені у [14].

Отримана оцінка трудовитрат (1) далі використовується для розрахунку витрат часу за формулою [14]

$$TDEV = \left[3,67 \times \left(PM_{NS} \right)^F \right] \times \frac{SCED\%}{100}, \tag{6}$$

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 формулою [14]

$$F = \left(0,28 + 0,2 \times \left[E - 0,91 \right] \right), \tag{7}$$

де E – масштабний показник для рівняння зусиль, який розраховується за формулою (3).

Моделі COCOMO II створювалися виключно для оцінювання витрат на створення чи модифікацію програмних систем. Використання моделей COCOMO II для оцінювання витрат на проведення рефакторингу БД стикається із значними труднощами. Ці труднощі пов'язані із визначенням правил розрахунку значень елементів *SizeAdded* та *SizeModified*. Тому виникає задача модифікації моделей COCOMO II з врахуванням особливостей рефакторингу БД як різновиду ІТ-проєкту.

5. Вирішення задачі оцінювання витрат на проведення рефакторингу бази даних інформаційної системи, яка експлуатується.

5.1. Оцінювання кількості рядків вихідного коду SQL-команд які використовуються під час проведення рефакторингу бази даних

Оцінювання рефакторингу БД з використанням такого показника, як кількість рядків вихідного коду, неодмінно приводить до визначення кількості рядків вихідного коду, який написано мовами створення, модифікації та управління даними в БД. Найрозповсюдженішою подібною мовою є SQL. Тому пропонується розраховувати кількісні значення елементів *SizeAdded* і *SizeModified* формули (4) як сукупну кількість рядків коду SQL-команд, що відповідають за додавання та модифікацію елементів БД, яка підлягає рефакторингу.

Найчастіше під час рефакторингу використовуються SQL-команди, які відносяться до групи команд Data Definition Language. Ці команди безпосередньо відповідають за роботу з таблицями БД і дозволяють виконувати такі операції [15]:

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

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

Розглянемо шаблони цих команд, враховуючи їх типові структури. Так, шаблон SQL-команди створення наведено на рис. 1 [16]. Шаблон SQL-команди видалення наведено на рис. 2 [16]. Шаблон SQL-команди редагування наведено на рис. 3 [16]. Шаблон SQL-команди очищення наведено на рис. 4 [16].

Використання цих шаблонів дає змогу оцінити кількість рядків вихідного коду для кожної з цих SQL-команд під час рефакторингу БД за умови, що ця БД може бути описана у термінах моделі «сутність – зв'язок» (Entity – Relation Model, ERM). Для цього пропонується використовувати такі моделі:

```
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>)
);
```

Рис. 1. Шаблон SQL-команди створення

$$Size_{CR_r} = \left| \left\langle atr_{E_r}^{j} \right\rangle \right| + 4; j = 1,...,N, \quad (8)$$

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

Рис. 2. Шаблон SQL-команди видалення

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

Рис. 3. Шаблон SQL-команди редагування

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

Рис. 4. Шаблон SQL-команди очищення

$$Size_{DR_r} = 1, \tag{9}$$

$$Size_{AL_r} = \left| \left\langle atr_{E_r}^{j} \right\rangle \right| + 1; j = 1,...,N, \tag{10}$$

$$Size_{TR_r} = 1, \tag{11}$$

де $Size_{CR_r}$ – кількість рядків SQL-команди створення; r – ідентифікатор сутності E_r із запиту на зміну інформаційної системи (та, відповідно, її БД); $\left\langle atr_{E_r}^{j} \right\rangle$ – множина атрибутів сутності E_r; N – кількість атрибутів, з яких складаються усі сутності із запиту на зміну інформаційної системи; $Size_{DR_r}$ – кількість рядків SQL-команди видалення; $Size_{AL_r}$ – кількість рядків SQL-команди редагування; $Size_{TR_r}$ – кількість рядків SQL-команди очищення.

У моделі (8) додатковий доданок становить суму кількості строк, в яких здійснюються чотири визначення: типу команди, первинного ключа, вторинного ключа та посилання цих ключів на іншу таблицю. Тому додатковий доданок дорівнює 4. У моделі (9) немає операцій з атрибутами, тому залишається тільки додатковий доданок, який дорівнює сумі кількісті строк в команді видалення, яка завжди дорівнює 1. У моделі (10) додатковий доданок становить 1 і описує кількість рядків, у яких здійснюється об'явлення типу команди. У моделі (11) немає операцій з атрибутами, тому залишається тільки додатковий доданок, який дорівнює сумі кількісті строк в команді очищення, що завжди дорівнює 1.

Використання моделей (8)-(11) дає змогу оцінити кількість рядків вихідного коду SQL-команд, які необхідно виконати для здійснення рефакторингу БД як ІТ-проєкту, що реалізує запити на зміну інформаційної системи та її БД, які експлуатуються. Для цього необхідно знати описи сутностей та атрибутів у запитах на зміну, які визнано необхідним реалізувати. Але ця інформація стає відомою ще під час пошуку відповіді на запитання «Чи має сенс операція рефакторингу, яка пропонується?», як показано авторами у [17].

5.2. Вдосконалення елементів моделі COCOMO II для її застосування під час оцінювання трудовитрат на проведення рефакторингу бази даних

Використання моделей (8)-(11) дозволяє вдосконалити модель (4) як елемент моделей COCOMO II, що дозволяє визначити кількість рядків вихідного коду для випадку ІТ-проєктів

вдосконалення, розвитку чи рефакторингу існуючих систем. Так, для ІТ-проєкту рефакторингу БД інформаційної системи, яка експлуатується, значення елементу *SizeAdded* формули (4) можна розрахувати за формулою

$$SizeAdded = \sum_{r=1}^{N_{CR}} Size_{CR_r}, \tag{12}$$

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

Значення елементу *SizeModified* формули (4) в цьому випадку можна розрахувати за формулою

$$SizeModified = \sum_{r=1}^{N_{DR}} Size_{DR_r} + \sum_{r=1}^{N_{AL}} Size_{AL_r} + \sum_{r=1}^{N_{TR}} Size_{TR_r}, \tag{13}$$

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.

Тоді модель (4) як елемент моделей COCOMO II для ІТ-проєкту рефакторингу БД інформаційної системи, яка експлуатується, матиме вигляд

$$(Size)_M = \left(\sum_{r=1}^{N_{CR}} Size_{CR_r} + \sum_{r=1}^{N_{DR}} Size_{DR_r} + \sum_{r=1}^{N_{AL}} Size_{AL_r} + \sum_{r=1}^{N_{TR}} Size_{TR_r} \right) \times MAF . \tag{14}$$

Таке вдосконалення дає змогу використовувати параметричні моделі COCOMO II для оцінювання трудовитрат на ІТ-проєкти рефакторингу БД.

5.3. Експериментальна перевірка результатів вдосконалення моделі COCOMO II.

Для експериментальної перевірки було обрано інформаційно-облікову систему управління послугами однієї з організацій, яка надає послуги Інтернет-провайдера. ER-діаграма БД цієї системи наведена на рис. 5.

Користувачі цієї інформаційно-облікової системи незадоволені тим, що вони не мають можливості самостійного гнучкого налаштування послуг (їх налаштування апріорно визначені). Ці недоліки планується виправити шляхом рефакторингу системи і, зокрема, рефакторингу БД цієї системи. Для опису запитів на зміни системи, яка експлуатується, були визначені запити на зміни, UserStory яких наведені на рис. 6 та рис. 7.

Лідер сумісно з командою розробників, проаналізувавши описи UserStory запитів на зміни, наведені на рис. 7 та рис. 8, створюють ER-діаграми, які описують схеми даних для цих запитів на зміни. Ці ER-діаграми наведені на рис. 9 та рис. 10.

На рис. 9 та рис. 10 українською мовою позначено таблиці та поля, що додаються запитами на зміну, англійською мовою позначено таблиці та поля актуальної БД, яка показана на рис. 6.

З рис. 9 видно, що під час проведення рефакторингу актуальної БД інформаційно-облікової системи як реалізації запиту на зміну «Створення замовлення» (далі позначається ідентифікатором 1) слід додати одну таблицю, яка складається з 5 атрибутів, та модифікувати одну таблицю,

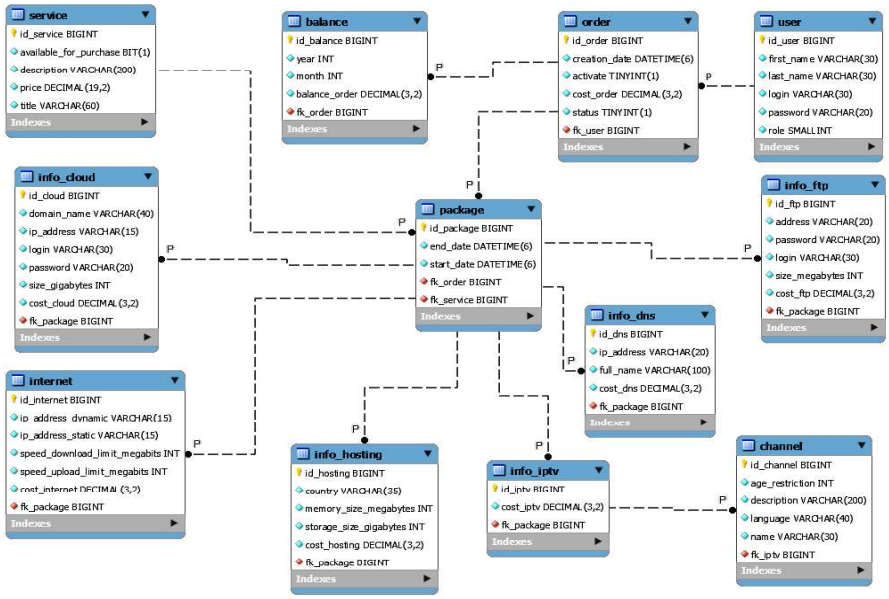

Рис. 5. ER-діаграма бази даних інформаційно-облікової системи організації, яка надає послуги Інтернет-провайдера

Рис. 6. UserStory запиту на зміну «Створення замовлення»

Рис. 7. UserStory запиту на зміну «Облік розподілу користувачів за населеним пунктом»

додавши один атрибут. Тому елементи *SizeAdded* і *SizeModified* будуть визначатися таким чином:

$$SizeAdded_1 = \sum_{r=1}^{1} Size_{CR_r} = 5+4 = 9\,, \quad SizeModified_1 = \sum_{r=1}^{1} Size_{AL_r} = 1+1 = 2\,. \qquad (15)$$

Під час проведення рефакторингу актуальної БД інформаційно-облікової системи як реалізації запиту на зміну «Облік розподілу користувачів за населеним пунктом» (далі

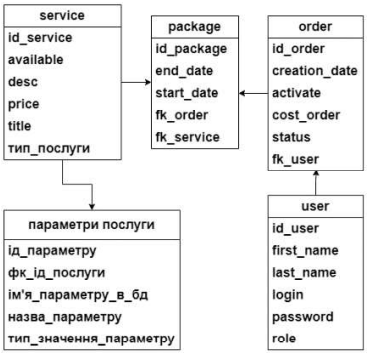

Рис. 8. ER-діаграма схеми даних для запиту на зміни «Створення замовлення»

Рис. 9. ER-діаграма схеми даних для запиту на зміни «Облік розподілу користувачів за населеним пунктом»

позначається ідентифікатором 2) слід додати дві таблиці,, перша з яких складається з 7 атрибутів, а друга – з 5 атрибутів. Тому для цього випадку визначається лише значення елементу *SizeAdded* таким чином:

$$SizeAdded_2 = \sum_{r=1}^{2} Size_{CR_r} = (7+4)+(5+4) = 20\,. \tag{16}$$

Інформаційно-облікова система має такі характеристики: високий рівень інтеграції, слабкий ступінь зв'язності елементів, добру кореляцію між програмою і додатком, добрий рівень присутності в коді коментарів та заголовків; корисну документацію, але є окремі слабкі місця. Тоді штраф, викликаний витратами на розуміння програмного коду, дорівнює 20. Команда проєкту повністю згуртована. Отже, розрахунок коригуючого коефіцієнту проведення рефакторингу БД *MAF* має наступний вигляд:

$$MAF = 1+\left(\frac{SU}{100}\times UNFM\right) = 1+\left(\frac{20}{100}\times 0\right) = 1+0 = 1\,. \tag{17}$$

Тоді загальна кількість рядків вихідного SQL-коду в запропонованому IT-проєкті рефакторингу БД за формулою (14) приймає таке значення:

$$(Size)_M = (9+2+20)\times 1 = 31\,. \tag{18}$$

Оскільки запити на зміни не змінювалися під час їх надходження та аналізу, коефіцієнт BRAK буде дорівнювати 0. Тому кількість рядків вихідного програмного коду для моделі (1), яка розраховується за формулою (2), приймає таке значення:

$$Size' = Size\times\left(1+\frac{BRAK}{100}\right) = 31\times\left(1+\frac{0}{100}\right) = 31\,. \tag{19}$$

Виходячи з наявності досвіду виконання подібних проєктів, незначних відхилень у гнучкості, завдання повних (100 %) специфікацій елементів, повної взаємодії між учасниками команди виконавців, а також першого рівня зрілості організації, яка надає послуги Інтернет-провайдера, розрахунок значення показника ступеня E має вигляд:

$$E = 0,91 + 0,01 \times \sum_{j=1}^{5} SF_j = 0,91 + 0,01 \times (0,03 + 0,02 + 0,05 + 0,05 + 0,01) = \\ = 0,91 + 0,01 \times 0,16 = 0,9116. \quad (20)$$

Значення драйверів витрат IT-проєкту EM_i визначалися, виходячи з таких особливостей:

– малий розмір БД;

– незначна складність продукту;

– основна увага приділяється надійності системи і, зокрема, документації;

– повторне використання коду дозволено в рамках продукту;

– можливі незначні зміни платформи розробки під час виконання проєкту;

– мінливість кадрів за рік становить менше 4 %;

– програмісти мають гарні навички та досить довго працюють над проєктом;

– кожен розробник має мінімум 4 роки досвіду виконання аналогічних проєктів;

– всі члени команди знаходяться в рамках одного будинку та мають достатню кількість засобів електронної комунікації;

– під час виконання робіт членами команди виконавців розклад цих робіт дотримується в середньому на 85 %.

Ці особливості дозволяють розрахувати значення драйверів витрат IT-проєкту EM_i таким чином:

$$П_{i=1}^{7} EM_i = 0,93 \times 1,14 \times 1,11 \times 0,7 \times 0,75 \times 0,78 \times 1,1 = 0,53. \quad (21)$$

Тоді значення трудовитрат на IT-проєкт рефакторингу БД за розглянутими запитами на зміну буде дорівнювати

$$PM = A \times [Size']^E \times П_{i=1}^{7} EM_i = 2,94 \times [0,031]^{0,9116} \times 0,53 = 0,065. \quad (22)$$

Таким чином, трудовитрати на рефакторинг БД інформаційно-облікової системи оцінюються у 0,065 людино-місяця.

Для розрахунку витрат часу спочатку вирахуємо значення показника масштабування для рівняння розкладу *F* за виразом (7). Це значення дорівнюватиме

$$F = (0,28 + 0,2 \times [E - 0,91]) = (0,28 + 0,2 \times [0,9116 - 0.91]) = 0,28032.$$

Тоді значення витрат часу на рефакторинг БД інформаційно-облікової системи дорівнюватиме (за виразом (6))

$$TDEV = \left[3,67 \times \left(PM_{NS} \right)^{F} \right] \times \frac{SCED\%}{100} = \left[3,67 \times (0,065)^{0,28032} \right] \times \frac{110}{100} = 1,876 .$$

Таким чином, витрати часу на рефакторинг БД інформаційно-облікової системи оцінюються у 1,876 місяця.

Отримані оцінки трудовитрат і витрат часу на рефакторинг БД дозволяють позитивно відповісти на питання «Чи виправдовує результат витрачені зусилля?» для розглянутого рефакторингу БД інформаційно-облікової системи як ІТ-проєкту з реалізації двох запитів на зміни цієї системи.

6. Обговорення результатів дослідження

Запропоноване вдосконалення моделей COCOMO II дозволяє, на відміну від загальноприйнятної версії цих моделей, оцінити трудовитрати на виконання рефакторингу БД інформаційної системи, яка експлуатується. Вибір для оцінювання доцільності проведення рефакторингу БД таких параметричних моделей, як моделі COCOMO II, дозволяє отримати доволі точні оцінки за рахунок використання при побудові цих моделей досвіду великої кількості ІТ-проєктів різного призначення.

Використання запропонованих у дослідженні рішень дає змогу адміністратору інформаційної системи, яка експлуатується, або менеджеру проєкту отримати достатньо об'єктивні оцінки витрат на виконання подібного ІТ-проєкту рефакторингу цієї системи. Таке вдосконалення, хоча і вимагає деякого збільшення обсягу робіт під час формування і прийняття рішення щодо доцільності проведення рефакторингу БД, дозволяє приймати ці рішення, базуючись не тільки на власному досвіді та інтуїції (як рекомендується у [2, 3]), а й на результатах об'єктивного аналізу складності схеми актуальної БД та запитів на зміну системи, яка експлуатується.

Головним недоліком отриманих результатів дослідження слід вважати обмеженість під час оцінки витрат на рефакторинг БД тільки чотирма типами SQL-команд. Хоча ці команди вважаються найуживанішими під час проведення рефакторингу БД, можливо також використання інших SQL-команд, які не були враховані авторами. Ще одним недоліком отриманих результатів є їх перевірка тільки на одному ІТ-проєкті рефакторингу БД.

Виходячи із зазначених недоліків, пропонується розглядати такі подальші перспективи проведення досліджень за напрямом кількісного оцінювання витрат на рефакторинг БД інформаційної системи, яка експлуатується:

– проведення досліджень із встановлення частоти використання окремих SQL-команд в процесі рефакторингу БД;

– експериментальні перевірки запропонованого вдосконалення моделей COCOMO II під час проведення рефакторингу БД інших інформаційних систем.

7. Висновки

У ході даного дослідження було вирішено задачу вдосконалення параметричної моделі оцінювання трудовитрат на ІТ-проєкти COCOMO II для її використання під час оцінки витрат на рефакторинг БД. Під час вирішення цієї задачі було здійснено:

– розробку моделей (8)-(11) оцінювання кількості рядків вихідного коду SQL-команд створення, видалення, редагування та очищення, які найчастіше використовуються під час проведення рефакторингу БД;

– вдосконалення моделі COCOMO II (4) шляхом представлення її у вигляді виразу (14), який дозволяє оцінити кількість рядків вихідного SQL-коду, які треба додати або модифікувати під час проведення рефакторингу БД;

– експериментальну перевірку розроблених і вдосконалених моделей в процесі оцінювання витрат на проведення рефакторингу БД інформаційно-облікової системи, яка експлуатується.

Перелік посилань:

1. Fowler M. Refactoring Home Page. *Refactoring*. URL: https://www.refactoring.com/ (дата звернення: 20.04.2023).

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

3. Database Refactoring: Improve Production Data Quality. *The Agile Data (AD) Method - Strategies for effective data-oriented development*. URL: http://agiledata.org/essays/databaseRefactoring.html (date of access: 07.04.2023).

4. The evolution of knowledge in the refactoring research field / M. Orrú et al. *XP 2015 Workshops: XP 2015 Scientific Workshops Proceedings*, Helsinki Finland. New York, NY, USA, 2015. https://doi.org/10.1145/2764979.2764989.

5. Vial G. Database Refactoring: Lessons from the Trenches. *IEEE Software.* 2015. Vol. 32. No. 6. P. 71–79. https://doi.org/10.1109/ms.2015.131.

6. Khumnin P., Senivongse T. SQL antipatterns detection and database refactoring process. *2017 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, Kanazawa, Japan, 26–28 June 2017. 2017. https://doi.org/10.1109/snpd.2017.8022723.

7. Synthesizing database programs for schema refactoring / Y. Wang et al. *PLDI '19: 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, Phoenix AZ USA. New York, NY, USA, 2019. https://doi.org/10.1145/3314221.3314588.

8. Filatov V., Semenets V. Methods for Synthesis of Relational Data Model in Information Systems Reengineering Problems. *2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T)*, Kharkiv, Ukraine, 9–12 October 2018. 2018. https://doi.org/10.1109/infocommst.2018.8632144.

9. Bohm L. Refactoring Legacy T-SQL for Improved Performance. Berkeley, CA: Apress, 2020. https://doi.org/10.1007/978-1-4842-5581-0.

10. Fernandes I. P., Terra-Neves M., Seco J. C. Automated Refactoring of Unbounded Queries in Software Automation Platforms. *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, Fukuoka, Japan, 10–15 October 2021. 2021. https://doi.org/10.1109/models-c53483.2021.00065.

11. Spasić M., Janičić M. V. Verification supported refactoring of embedded SQL. *Software Quality Journal*. 2020. Vol. 29, no. 3. PP. 629–665. https://doi.org/10.1007/s11219-020-09517-y.

12. Managing Technical Debt in Database Normalization / M. Albarak et al. *IEEE Transactions on Software Engineering*. 2020. P. 1. https://doi.org/10.1109/tse.2020.3001339.

13. How do i refactor this? An empirical study on refactoring trends and topics in Stack Overflow / A. Peruma et al. *Empirical Software Engineering*. 2021. Vol. 27, no. 1. https://doi.org/10.1007/s10664-021-10045-x.

14. COCOMO II Model Definition Manual Version 2.1. 2000. 90 p. URL: https://www.rose-hulman.edu/class/cs/csse372/201310/Homework/CII_modelman2000.pdf (дата звернення: 12.04.2023).

15. 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).

16. SQL DDL Commands | Learn the DDL Commands of SQL. *EDUCBA*. URL: https://www.educba.com/sql-ddl-commands/ (дата звернення: 12.04.2023).

17. Євланов М.В., Юр'єв І.О., Мірошниченко Д.О. Оцінювання доцільності проведення рефакторингу бази даних інформаційної системи, яка експлуатується. *АСУ і прилади автоматики.* 2022. Вип. 178. С. 13–22. DOI:10.30837/0135-1710.2022.178.013

Надійшла до редколегії 25.04.2023 р.

Євланов Максим Вікторович, доктор технічних наук, професор, професор кафедри ІУС ХНУРЕ, м. Харків, Україна, e-mail: maksym.ievlanov@nure.ua, ORCID: https://orcid.org/0000-0002-6703-5166

Юр'єв Іван Олексійович, кандидат технічних наук, доцент, доцент кафедри ІУС ХНУРЕ, м. Харків, Україна, e-mail: ivan.iuriev@nure.ua, ORCID: https://orcid.org/0000-0002-5178-519X

Мірошниченко Дмитро Олександрович, здобувач вищої освіти, група УПГІТм-21-1, факультет комп'ютерних наук, ХНУРЕ, м. Харків, Україна, e-mail: dmytro.miroshnychenko@nure.ua.