ISSN 2707-1898 (print)

Український журнал інформаційних технологій
Ukrainian Journal of Information Technology
http://science.lpnu.ua/uk/ujit
https://doi.org/10.23939/ujit2023.02.001
Article received 29.09.2023 p.
Article accepted 26.10.2023 p.
UDK 004.4

Correspondence author
O. B. Kernytskyy
oleh.b.kernytskyi@lpnu.ua

УДК 004.4

О. Б. Керницький, В. М. Теслюк
Національний університет "Львівська політехніка", м. Львів, Україна

МЕТОД РОЗРОБЛЕННЯ СПЕЦИФІКАЦІЙ ТА ВИМОГ В ПРОЦЕСІ РЕІНЖИНІРИНГУ ІТ-ПРОЄКТІВ

Мета дослідження – удосконалення методу синтезу вимог та специфікацій для реінжинірингу ІТ-проєктів з максимальною ефективністю та бізнес-орієнтованістю. Основне завдання – адаптація застарілих ІТ-систем до змінюваного технічного середовища, зокрема до хмарних технологій та вимог до систем безпеки. Для досягнення цих цілей запропоновано метод, що використовує аналіз архаїчних систем, метод зворотного розроблення, опитування експертів, аналіз документації та моделювання чорного ящика. Застосування цих методів дає змогу виявити та коригувати вимоги та специфікації, забезпечуючи високий рівень якості та ефективності у процесі реінжинірингу ІТ-проєктів. У статті розглянуто практичне використання методу, перспективи подальшого розвитку та особливості застосування різних статистичних методів у процесі покращення результатів реінжинірингу. Описано принципи роботи методу разом із основними підходами та техніками, які сприяють аналізу наявних ІТ-систем, синтезу вимог та специфікацій, контролю якості та ефективності реалізації проєктів реінжинірингу. Окремо проаналізовані складові методу передбачають збирання даних про наявну систему та здійснення аналізу архаїчних систем з метою визначення вимог до нової системи. Розглянуто застосування моделі чорного ящика для тестування розробленої системи, ураховуючи аналіз отриманих результатів, коректування вимог та покращення специфікацій. Метод включає засоби аналізу документації, реверсивного інжинірингу, опитувань і відображення даних, а також методики аналізу, наприклад формулу паралельного тестування, формулу матриці відповідності вимог та формулу прогнозування вимог на основі аналізу швидкості розбіжностей.

Ключові слова: реінжиніринг ІТ-проектів; архаїчні системи; паралельне тестування; модель чорного ящика; матриця трасування вимог; хмарні технології.

Вступ / Introduction

Одним із вкрай важливих завдань сьогодення є підвищення ефективності функціонування досі використовуваних програмних систем, розроблених у 80–90-ті роки минулого століття, а це неможливо реалізувати без сучасних технологій. Окрім того, загострилось і питання безпеки даних у таких системах, в умовах їх збереження у хмарі та використання інтернет-технологій. Зазвичай такі давнішні програмні системи написані на мовах Асемблер, Кобол чи Сі, що нині вважають застарілими мовами програмування, але з певних причин досі підтримують в робочому стані.

Замовники з реінжинірингу таких систем вимагають використовувати хмарні технології так, щоб із погляду бізнесу нова система працювала ідентично до старої. Саме тому, виникає потреба в їх вдосконаленні та/або розробленні ІТ-систем, які забезпечать ті самі функції, але застосовуватимуть сучасні інформаційні технології. В таких випадках, зазвичай, виникає потреба розділення та категоризації вимог на такі, що були зумовлені обмеженнями чи специфікою архаїчної системи, та на такі, що задовольняють бізнес-потреби.

Об'єкт дослідження – процес розроблення специфікацій та вимог до реінжинірингу ІТ-проєктів.

Предмет дослідження – метод за засоби розроблення специфікацій та вимог щодо реінжинірингу ІТ-проєктів.

Мета роботи – підвищення ефективності розроблення специфікацій та вимог під час реінжинірингу ІТ-проєктів з використанням хмарних технологій.

Для досягнення зазначеної мети визначено такі основні *завдання дослідження*:

- проаналізувати наявні методи, моделі та засоби синтезу вимог та специфікацій під час реінжинірингу ІТ-проєктів;
- вдосконалити метод синтезу специфікацій та вимог при реінжинірингу ІТ-проєктів;
- розробити засоби формування специфікацій та вимог стосовно реінжинірингу ІТ-проєктів і дослідити ефективність розробленого методу та навести приклади застосування удосконаленого методу.

Аналіз останніх досліджень та публікацій. Розробники у своїх працях запропонували кілька підходів до вирішення вищезазначеної проблеми. Зокрема, у статті Y. A. Luna-Herrera, J. C. Pérez-Arriaga, J. O. Ocharán-Hernández, Á. J. Sanchéz-García (2023) розглянута проблематика всебічного розуміння комп'ютерних програм за допомогою підходів і методів зворотного проєктування [1]. Автори проаналізували різні підходи та техніки реверсивного інжинірингу. Вони зауважили, що для розробників програмного забезпечення звично працювати над застарілими системами, проте більшість цих систем не мають відповідної документації та артефактів дизайну або їхня якість незадовільна та не відповідає сучасним стандартам. У статті R. Marinescu (2012) відзначається проблематика підтримки наявних IT-систем в робочому стані, коли відомо, що певні функції виконуються неправильно. Це призводить до технічного боргу, який впливає на рішення щодо дизайну, які забезпечують переваги в короткостроковій перспективі, але в довгостроковій перспективі призводять до збоїв у структурі системи. Внаслідок цього істотно зростають витрати на обслуговування IT-системи [2]. Автори статей P. Jamshidi, A. Ahmad, & C. Pahl (2013) [3] та V. Andrikopoulos, T. Binz, F. Leymann, & S. Strauch (2013) [4] досліджують проблеми, виклики та переваги заміни старих IT-систем на хмарні рішення, ураховуючи питання міграції, адаптації та інтеграції. Вони також розглядають наукові та практичні аспекти переходу до хмарних технологій, що можуть слугувати обґрунтуванням для заміни старих систем. В статті авторів A. K. Maji, S. Mitra, and B. Zhou (2015) запропоновано метод паралельного тестування, який включає генерацію тестових векторів та аналіз відгуків схеми. Цей метод ґрунтується на використанні графів управління (control-flow graphs) та графів даних (data-flow graphs) для ефективного тестування [5]. У статті M. Adnan, & N. Mirza висвітлено дослідження, спрямоване на розуміння і відновлення інформації з документів старої системи під час реінжинірингу застарілої системи. У статті наголошено на аналізі документів та процесі відновлення інформації, розробленні спеціальної метамоделі для цієї області та використанні цього підходу для подальшого вдосконалення нової системи [6]. У статті Galbois, J., & Bournez, G. досліджено використання Data Mining в контексті технічної документації для поліпшення процесів старих систем. Автори аналізують використання знань, що здобувають з технічних документів, для реінжинірингу процесів цих систем і визначення вимог для нових IT-рішень [7]. А в статті авторів Krishna, R., Alshayeb, M., Hattab, G., Zheng, Q., Chivers, M., & Lisitenko, D. досліджено проблеми та виклики, пов'язані з використанням зворотного програмування в контексті неперервного супроводу програмного забезпечення (Continuous Software Maintenance) [8]. Автори пропонують підхід до використання методології реверс-інжинірингу з метою покращення процесу програмного супроводу. У статті авторів Wu, Y., Brinkkemper, S., & Li, X. проаналізовано використання та співіснування агільних та структурованих методик для реалізації успішних програмних проєктів [9]. Chin, L., & Sturer, J. у своїй статті описують дослідження методології розроблення IT-систем для IBM та обговорюють питання, пов'язані з розбіжностями між старою та новою системами. У ній запропоновано методи для ідентифікації та аналізу конфліктів, що допоможуть виявити і вирішити проблеми під час реінжинірингу систем [10]. У книзі K. S. Rubin наведено докладний огляд Scrum – популярної агільної методології, яка пропонує методології з відстеження завдань та відповідного прогресу роботи над IT-проєктами. Вона підкреслює важливість ефективного планування та стеження за завданнями під час розроблення IT-проєктів [11]. Автори Ammann, P. & Offutt, J. у своїй книзі подають літопис методів тестування програмного забезпечення та аналізу результатів та порушують питання контролю якості під час розроблення нових IT-систем [12]. Матеріал надає керівництво щодо розроблення тестів та оцінювання їх ефективності для забезпечення якості системи та забезпечення продуктивності проєкту.

Отже, комплексний аналіз літературних джерел з проблематики дослідження дає підстави стверджувати, що підвищення ефективності синтезу специфікацій та вимог до IT-проєктів з перебудови архаїчних систем потребує розроблення нових і вдосконалення відомих методів і моделей та використання сучасних інформаційних технологій.

Результати дослідження та їх обговорення / Research results and their discussion

Особливості методу реінжинірингу IT-проєктів. Розглянемо особливості застосування методу реінжинірингу IT-проєктів на прикладі, коли первинне збирання вимог та специфікацій уже здійснено. Після проведеного аналізу документації, опитування експертів та продуктових менеджерів, отримання псевдокоду, як результату зворотного розроблення, його аналізу буде сформований перелік вимог для нового IT-продукту, чи його окремої частини. Кожного з перелічених джерел може бути недостатньо для розуміння функціонування архаїчної системи повною мірою і, як наслідок, для розроблення специфікації нової систем. Насправді розробники можуть навіть не підозрювати про те, що сформулювали некоректні вимоги. Причиною цього може бути неповна або неточна документація, обмежене чи фрагментоване знання відповідальних експертів, складність аналізу старого коду, а також обмежений людський та фінансовий ресурс для його виконання. BABOK Guide [13], згадуючи про необхідність проведення експериментів під час синтезу вимог, також вказує на те, що деяку інформацію неможливо отримати від людей чи з документації просто тому, що ця інформація нікому не відома. Отже, розробники опиняються в ситуації, коли є N вимог у новій специфікації, які верифіковані, проте не провалідовані. В такому випадку розробники можуть розглянути стару систему як чорний ящик та використати її для тестування вимог до нової системи.

Цей процес можна виконувати одразу після того, як синтезовано нові вимоги. Проте в такому випадку валідацію можна здійснити лише для тестових сценаріїв розроблених *N* вимог. Тобто розробники можуть частково корегувати вимоги за результатами виявлених розбіжностей між очікуваним і реальним результатом функціонування проєктованої системи.

На наступному етапі, після завершення основного розроблення системи, з'являється можливість запустити

на виконання нову систему паралельно до старої в прихованому режимі виробничого середовища [14]. Цей метод називають паралельним тестуванням (рис. 1). Такий метод зазвичай пов'язаний з додатковими фінансовими витратами [15], тому ми не можемо повноцінно тестувати нову систему поряд зі старою, з якої трафік і дані дублюватимуться в нову, імітуючи справжню роботу. Тому визначають певний набір критеріїв оцінки важливості тих випадків, які необхідно проаналізувати. На перших етапах можна вибрати невеличких малокритичних клієнтів, які користуються системою. В такому випадку можна буде оцінити обмежений набір випадків та проаналізувати роботу нової системи на відповідність вимогам та специфікації. На пізнішому етапі виникне необхідність залучати для оброблення дані, отримувані від "великих" клієнтів, і це будуть значні обсяги даних. Якщо подивитись на стару систему, то можна вважати, що відбувається тестування чорного ящика. Проте прямо не відбувається валідація *N* вимог до нової системи. Отримані результати функціонування систем з використанням справжніх виробничих даних продемонструють, чи окремі розроблені вимоги до нової системи є правильними та коректними. У цьому випадку розробники виконують ролі оглядачів, але не можуть впливати на вхідні дані та здійснювати власні експерименти [16]. Важливо швидко та ефективно аналізувати розбіжності, виявлені в роботі проєктованої системи.

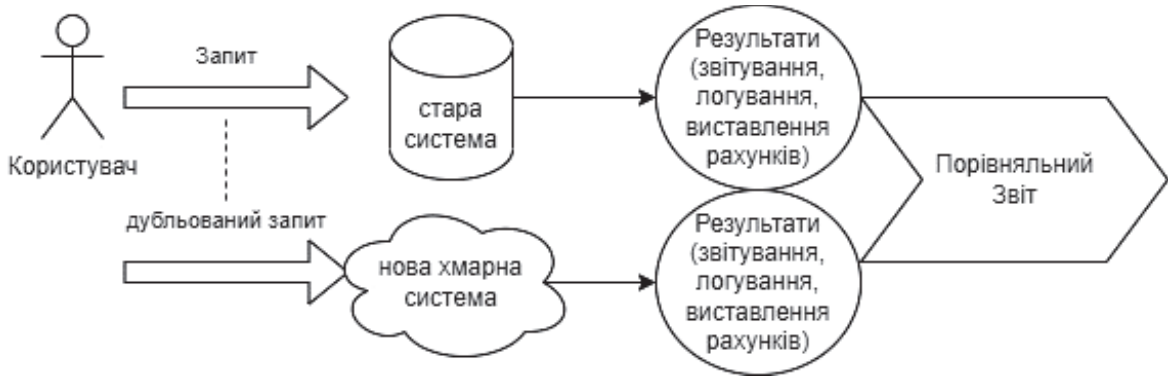

Рис. 1. Схема процесу паралельного тестування нової та старої системи /
Schematic representation of the parallel testing process for the old and new systems

Виявивши розбіжності в роботі нової системи, важливо швидко ідентифікувати, які з вимог у новій специфікації суперечать результатам функціонування старої архаїчної системи. Для цього необхідно виконати ідентифікацію виявлення суміжних правил та вимог і скласти перелік скомпрометованих правил.

Якщо розробники працюють вже з даними виробничої системи, вони мають змогу проаналізувати трафік та вибрати для аналізу усі можливі випадки, які так чи інакше мають критерії, які пов'язані з скомпрометованими вимогами. Розробники здійснюють декомпозицію бізнес-сценаріїв на основі виявлених випадків, які не вдається пояснити. Спостерігаючи за поведінкою системи, інженери намагаються формалізувати пропущені вимоги.

Якщо, для прикладу, в 95 % випадків стара система працює, як було очікувано, а в 5 % ні, проте у нас є чіткий додатковий критерій для описання цих 5 %, розробники можуть переходити на наступний етап верифікації вимог з продуктовим менеджментом. Якщо ж не вдалося зрозуміти причини розбіжностей у 5 % випадків, потрібно виконувати додатковий поглиблений аналіз саме цих випадків та виявити причини, які призводять до розбіжностей. Залежно від доступних засобів, необхідно знову повертатися до аналізу старої документації, опитування експертів та системних аналітиків старої системи. В загальному випадку може бути кілька причин таких розбіжностей, як і кілька шляхів вирішення щодо імплементації. Отже, причинами розбіжностей можуть бути:

1. Розбіжність у трактуванні вимог, неточність вимог або пропущені вимоги.

2. Об'єктивні нефункціональні обмеження старої системи.

3. Помилки в роботі. Можливі помилки як у старій, так і у новій системі.

Для формалізації цього процесу слід використати формулу аналізу швидкості розбіжностей:

$$V\_D = (N\_D - N\_A) / T, \quad (1)$$

де V_D – швидкість розбіжностей у паралельному тестуванні; N_D – кількість виявлених розбіжностей; N_A – кількість аналізованих випадків; T – час тестування.

Запропонований вираз (1) дає змогу відстежувати ефективність процесу розроблення та коригування вимог і специфікацій.

Необхідно взяти до уваги критичність розбіжностей в роботі між новою та старою системами. Критеріями оцінки критичності є такі фактори, як відсутність окремих важливих результатів функціонування нової системи, невідповідність результатів або процесу функціонування регулятивним та законодавчим вимогам, неможливість скористатися результатами функціонування нової системи з боку важливих користувачів. Якщо розбіжності охарактеризовані як некритичні або їх кількість менша від певного порога, наприклад, 1 %, приймають рішення про необхідність подальших досліджень та в результаті одержання чи корекції нових вимог та внесення їх до специфікації. Тоді отримаємо $N+n$ вимог, де N – первинна кількість вимог, а n – кількість нових вимог. Іншим варіантом подальших кроків може бути прийняття рішення не вносити жодних змін у нову специфікацію. Виявлену розбіж-

ність документують та приймають як допустиму з вказанням причин такого рішення. Описаний алгоритм дій можна зобразити у вигляді блок-схеми прийняття рішення та документування нових вимог у специфікації після виявлення розбіжності між наявними та очікуваними результатами під час тестування розробленої моделі чи паралельного функціонування старої та нової систем (рис. 2).

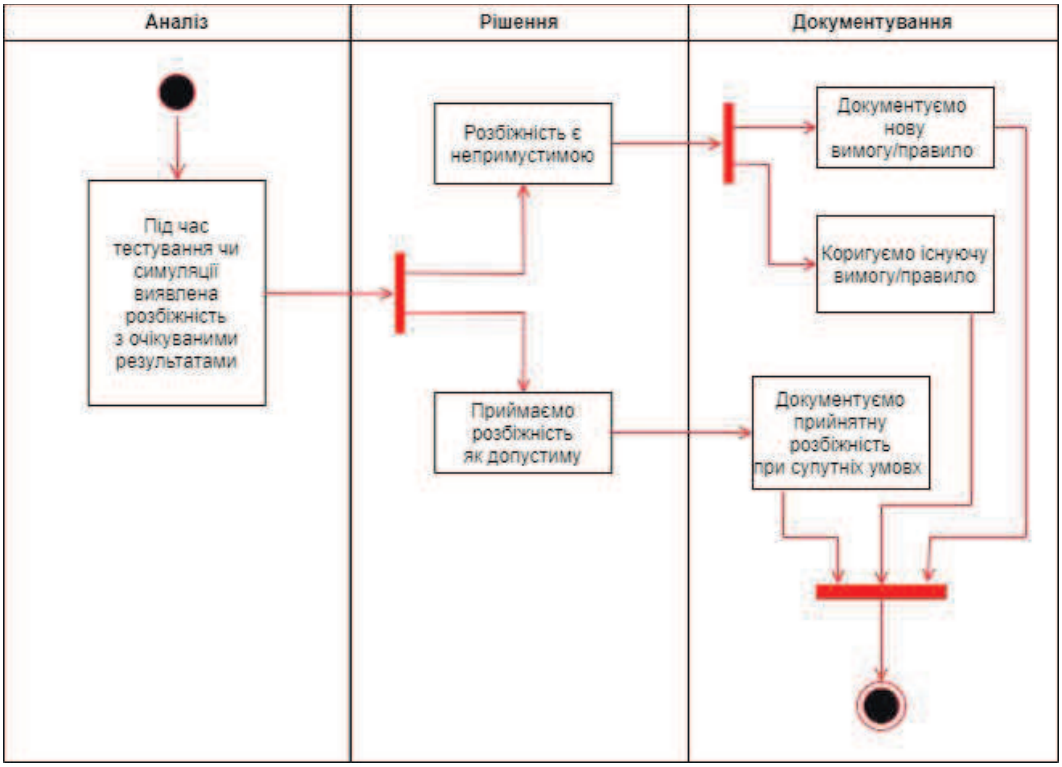

Рис. 2. Схема прийняття рішення та документування нових вимог у специфікації після виявлення розбіжності між наявними та очікуваними результатами під час тестування розробленої моделі чи паралельного функціонування старої та нової систем / Decision-making and documentation of new requirements in the specification after identifying discrepancies between existing and expected results during testing of the developed model or parallel functioning of the old and new systems

Для контролю над наявними вимогами N та новими вимогами n можна застосувати формулу матриці відповідності вимог (Requirement Traceability Matrix, RTM):

$RTM = [r\_ij]$, де r_ij – відповідність між i-м джерелом вимог (наприклад, наявною системою, документацією, висновками експертів, допустимою розбіжністю в роботі нової та старої систем) та j-ю вимогою до нової системи. *RTM* допомагає візуалізувати зв'язки між різними джерелами вимог та вимогами до нової системи, що полегшує аналіз та узгодження вимог.

Розроблення алгоритму методу реінжинірингу ІТ-проєктів. Алгоритм реалізації методу реінжинірингу ІТ-проєктів являє собою цілісний та структурований підхід до модернізації систем. Враховуючи вищезазначений матеріал, наведено алгоритм реалізації методу реінжинірингу ІТ-проєктів (рис. 3), який передбачає такі основні кроки:

Крок 1. Збирання даних про наявну систему (збирання вимог, аналіз документів, реверс-інжиніринг, опитування спеціалістів).

Крок 2. Аналіз отриманих даних та синтез попереднього варіанта вимог.

Крок 3. Побудова тестової моделі чорного ящика наявної системи.

Крок 4. Тестування розробленої моделі.

Крок 5. Аналіз результатів тестування. На основі отриманих результатів виконують корекцію вимог. Якщо істотно відрізняються, то перехід до кроку 2.

Крок 6. Розроблення нової системи. Паралельне виконання нової та старої систем.

Крок 7. Аналіз отриманих результатів.

Крок 8. Виконані всі вимоги. Якщо Ні, то коригують модель (на основі розбіжностей) до кроку 4.

Наведений алгоритм містить послідовність кроків, які спрямовані на забезпечення успішної модернізації системи, від збирання даних про проєктовану ІТ-систему до паралельного виконання та аналізу результатів. Це сприяє ефективному переходу від старої до нової системи з мінімізацією ризиків. Реінжиніринг архаїчних ІТ-систем є актуальним завданням для багатьох організацій, які стикаються із застарілими системами та технологіями. Розроблений алгоритм надає систематичний підхід до вирішення цієї проблеми. Це має допомогти розробникам забезпечити успішний перехід від старої до нової системи. Алгоритм застосовний до широкого спектра ІТ-проєктів, що потребують модернізації або заміни наявних систем, що робить його універсальним та корисним для різних контекстів.

Для оцінювання ефективності алгоритму доцільно використати формулу ефективності визначення вимог:

$$Eff = N\_C / N\_T, \quad (2)$$

де Eff – ефективність визначення вимог; N_C – кількість коректно визначених вимог; N_T – загальна кількість вимог.

Вираз (2) дає змогу оцінити ефективність процесу визначення вимог та специфікацій, що формують основу для розроблення нової ІТ-системи.

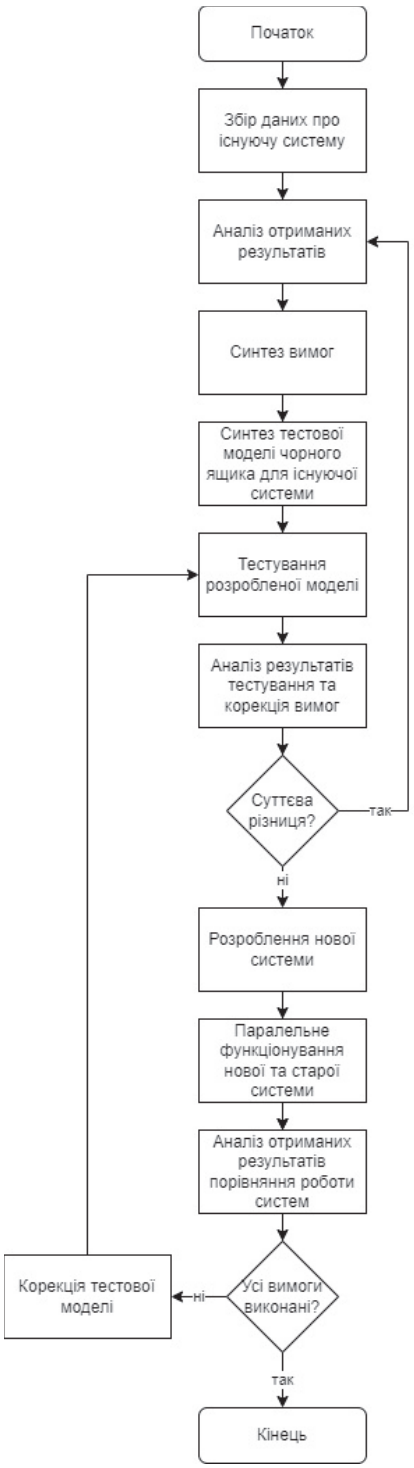

Рис. 3. Блок-схема алгоритму застосування методу розроблення специфікацій та вимог до IT-проєктів / Flowchart of the algorithm for applying the method for developing specifications and requirements for IT projects

Під час дослідження зібрано статистичні дані, що обліковують кількість нових вимог, кількість скоригованих вимог та кількість вимог, вилучених зі специфікації. Збирання даних відбувалося протягом періоду, що охоплював початкове збирання вимог, редагування вимог під час тестування тестової моделі чорного ящика, а також період паралельного тестування.

У лівій частині графіка (рис. 4) можна спостерігати завершення фази збирання і корекції вимог, яка припала на тестування розробленої моделі. Під час розроблення нової системи зареєстровано мінімальну кількість змін у вимогах. У правій частині графіка відображено активну фазу змін у вимогах, яка припадає на період паралельного тестування та аналізу отриманих результатів.

Одержавши історичні дані, можна скористатися формулою прогнозування вимог на основі регресійного аналізу, а саме:

$$y = a \times x + b, \qquad (3)$$

де y – цільове значення (наприклад, відповідність вимогам бізнесу), x – описове значення (наприклад, відповідність технологічним обмеженням), a та b – коефіцієнти регресії, розраховані на основі історичних даних.

Вираз (3) дає можливість прогнозувати вимоги та специфікації для нових IT-систем на основі аналізу ретроспективних даних.

Зауважимо, що вирази (1)–(3), інтегровані у метод збирання вимог під час реінжинірингу IT-проєктів (формули матриці відповідності вимог, аналізу швидкості розбіжностей, ефективності визначення вимог та формула прогнозування вимог на основі регресійного аналізу), підвищують ефективність процесу та допомагають досягти кращих результатів реінжинірингу. Проте треба виважено ставитися до їх використання на практиці, оскільки будь-яка інтеграція є трудомісткою.

Особливості засобів реалізації методу реінжинірингу IT-проєктів. Відповідно до вищевикладеного алгоритму, особливості засобів реалізації методу реінжинірингу IT-проєктів можна розділити на дві групи: аналітичні інструменти і методики підтримки прийняття рішень.

Аналітичні інструменти:

1. Засоби аналізу застарілої документації, які дають можливість виявити вміст, структуру та логіку старої системи для вдосконалення нової системи [6], [7].
2. Засоби зворотного програмування, які можуть спрощувати процес аналізу старої системи і визначення вимог до нової IT-системи (наприклад IDA Pro – Interactive Disassembler, Ghidra, jadx-Java Decompiler) [17].
3. Інструменти опитування експертів, які полегшують відстеження і систематизацію знань експертів у процесі синтезу вимог (наприклад SurveyMonkey, Google Forms, Qualtrics).
4. Засоби відображення даних та пошуку закономірностей (наприклад, Google Data Studio [18]) для візуалізації та ефективного аналізу даних.

Методики підтримки прийняття рішень:

1. Розроблення методів аналізу усвідомленості процесу для ідентифікації складності та негативних наслідків під час визначення вимог [9].
2. Розроблення методів аналізу конфліктів між поведінкою старої та нової систем для виявлення помилок та невідповідностей у розробленій системі [10].
3. Розроблення методологій для відстеження завдань, які забезпечують прогрес роботи над реінжинірингом IT-проєктів [11].
4. Розроблення методів контролю якості нової системи на основі використання засобів тестування, аналізу результатів та зворотного зв’язку [12].

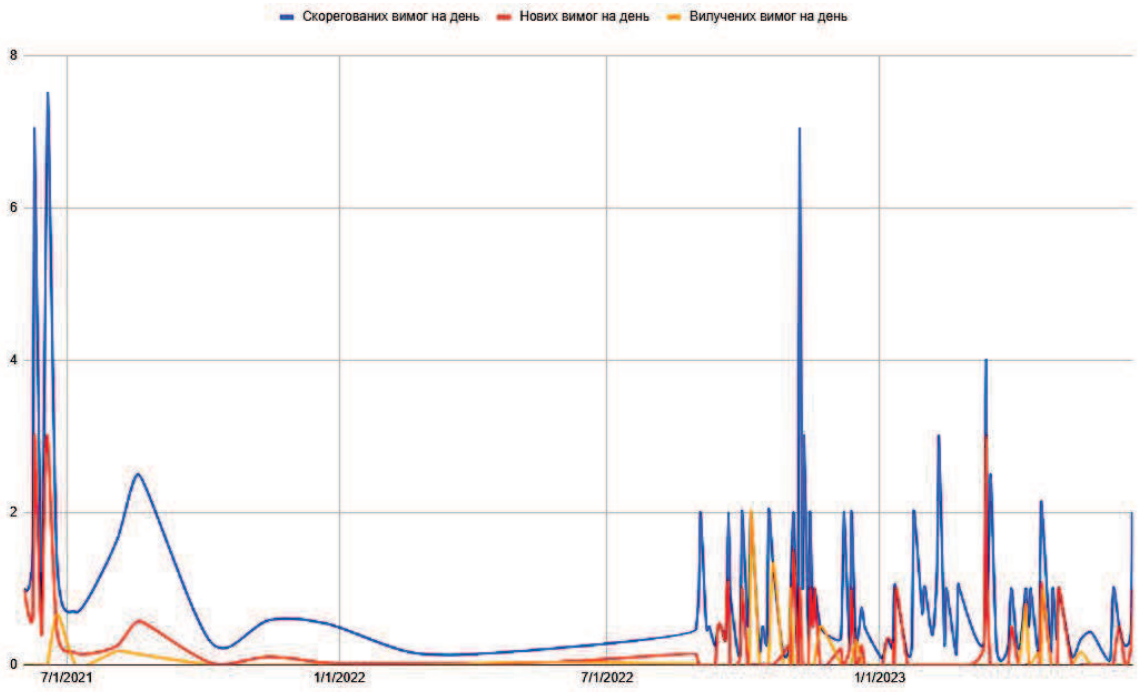

Рис. 4. Статистика змін вимог до нової IT-системи / Statistics of requirement changes for the new IT system

Особливості засобів реалізації методу реінжинірингу IT-проєктів із використанням цих інструментів та методів охоплюють систематичний та послідовний підхід до модернізації наявних IT-систем, забезпечують успішний перехід від старої до нової системи з мінімізацією ризиків, сприяють ефективному визначенню вимог та специфікацій та враховують специфіку роботи старої системи. Цей підхід є корисним для розроблення нових IT-проєктів та для оптимізації наявних, гарантуючи стійкість і надійність сучасної IT-інфраструктури.

Обговорення результатів дослідження. Розроблений метод та використані засоби випробувано в проєкті з перебудови архаїчної IT-системи кредитного бюро. Вибрано конкретне завдання із синтезу визначеного сегмента інформації, яка є частиною кредитного звіту споживача, а саме "Інформація для споживачів щодо Закону про кредитування військовослужбовців". Виконано паралельне тестування нової та старої систем на виробничих даних за вибрані проміжки часу без можливості впливати на вхідну інформацію.

У табл. 1 подано результати порівняльного аналізу роботи двох систем, а саме крок 6 вищезазначеного методу. Під ітерацією розумітимемо кожен перехід з кроку 8 до кроку 4. Вказана дата є днем проведення паралельного тестування протягом усіх 24 годин. Кількість випадків є кількісним обчисленням звітів, які містять у собі досліджуваний сегмент звіту внаслідок функціонування нової чи старої систем. Їх кількість прийнято за 100 відсотків. Збіги – кількість звітів, у яких інформація в досліджуваному сегменті повністю збігається. Відсотковий показник, якого прагнули розробники, становив 99 %. Додані та видалені – це ті звіти, у яких досліджуваний сегмент було неочікувано додано чи, навпаки, видалено. Змінені – звіти. у яких блок інформації є, проте, за результатами функціонування старої та нової систем, інформація відрізняється.

Для збирання даних кількісного аналізу використано Google Data Studio [18], [19], безкоштовний інструмент від Google для створення інтерактивних дашбордів та звітів на основі даних, які зберігаються у різних джерелах, таких як Google Sheets, Google Analytics, та у нашому випадку в BigQuery. Цей застосунок дав змогу візуалізувати та ефективно аналізувати дані, перетворивши їх на таблиці, що значно полегшило розуміння та сприяло швидкому прийняттю рішень.

Після кожної ітерації розробники мали змогу швидко отримати кількісний результат про невідповідність функціонування нової системи, отримати доступ до відповідних даних транзакцій для виконання детального аналізу, прийняти рішення про зміни чи доповнення вимог до нової системи, здійснювати імплементацію нових змін розробниками, та здійснювати нове паралельне тестування. Як бачимо з табл. 1, результат "Збіги, %", який характеризує правильність роботи системи, покращився – від 0,71 % до показника, який дещо перевищив той, якого прагнули розробники, 99,61 %.

Наукова новизна отриманих результатів дослідження – вдосконалено метод розроблення синтезу специфікацій та вимог під час реінжинірингу IT-проєктів, який враховує специфіку роботи старої системи у поєднанні з актуальними бізнес-потребами та дає змогу ефективно аналізувати і синтезувати вимоги до нових IT-систем на основі аналізу архаїчних систем.

Практична значущість результатів дослідження – розроблено алгоритм реалізації методу реінжинірингу IT-проєктів, що являє собою цілісний та структурований підхід до модернізації наявних систем та отримано результати досліджень.

Табл. 1. Результати порівняння роботи двох систем / Comparison results of the functioning of the two systems

Ітерація	Дата	Кількість випадків, що потрапляють у критерії вибірки	Збіги	Додані	Видалені	Змінені	Збіги, %	Додані, %	Видалені, %	Змінені, %
	Date	# occurrences	Matched	Added	Removed	Changed	Matched, %	Added, %	Removed, %	Changed, %
Перша	15.03.23	35038	249	6398	0	28391	0,71	18,26	0,00	81,03
	16.03.23	37172	270	6469	0	30433	0,73	17,40	0,00	81,87
	17.03.23	39887	305	6377	0	33205	0,76	15,99	0,00%	83,25
	20.03.23	35674	223	6681	0	28770	0,63	18,73	0,00	80,65
Друга	23.03.23	30450	249	2091	0	28110	0,82	6,87	0,00	92,32
Третя	24.03.23	29180	217	916	397	27650	0,74	3,14	1,36	94,76
	27.03.23	36485	223	780	288	35194	0,61	2,14	0,79	96,46
	28.03.23	16323	206	645	87	15385	1,26	3,95	0,53	94,25
	30.03.23	23680	182	775	326	22397	0,77	3,27	1,38	94,58
Четверта	04.04.23	35514	16547	0	294	18673	46,59	0,00	0,83	52,58
	11.04.23	31392	15386	0	215	15791	49,01	0,00	0,68	50,30
	18.04.23	33220	15619	1	262	17338	47,02	0,00	0,79	52,19
П’ята	03.05.23	32817	32716	0	18	83	99,69	0,00	0,05	0,25
	04.05.23	34305	34172	0	21	112	99,61	0,00	0,06	0,33

Висновок / Conslusions

Розроблено математичне, програмне та алгоритмічне забезпечення системи синтезу специфікацій та вимог під час реінжинірингу ІТ-проєктів. Виконано дослідження розробленого методу, оцінено його ефективність та наведено приклад застосування. Досягнутий ефект полягає у скороченні часу, необхідного для розроблення специфікацій, швидкому коригуванні вимог для нової інформаційної системи, високому відсотку коректності задокументованих вимог для нової інформаційної системи. Використання паралельного тестування та аналізу результатів сприяє забезпеченню відповідності нової системи до очікуваних результатів, виявленню розбіжностей та коригуванню вимог на основі виявлених проблем. Особливість такого методу полягає у використанні моделі чорного ящика та алгоритмів паралельного тестування систем, за допомогою яких можна виявити та коригувати вимоги для нової інформаційної системи.

Вдосконалення методу збирання вимог під час реінжинірингу ІТ-проєктів полягає у вирішенні деяких основних аспектів, таких як систематичний та послідовний підхід до аналізу старої системи, створенні ефективних механізмів для збирання та аналізу даних, координації дій з різними зацікавленими сторонами, а також контролю якості та успіху проєкту.

До вдосконалень розробленого методу також можна зарахувати:

1. Використання методів аналізу та редукції ризику, що допоможуть виявити потенційні проблеми і труднощі на ранніх стадіях розроблення проєкту.

2. Застосування хмарних технологій для підтримки послідовного процесу синтезу вимог та прискорення реінжинірингу ІТ-проєктів.

3. Вбудування механізмів відновлення та відповідності до сучасних стандартів безпеки, щоб забезпечити надійність нових ІТ-систем.

4. Упровадження системи документації та відстеження змін вимог та специфікацій, що сприяє продуктивності розробки й узгодженості змін у проєкті.

Окрім того, отримані знання та розуміння засобів синтезу специфікацій та вимог стануть базою для розвитку наукової основи та підвищення рівня відповідності нових ІТ-продуктів, що розробляються із використанням хмарних технологій.

References

[1] Luna-Herrera, Y. A., Pérez-Arriaga, J. C., Ocharán-Hernández, J. O., & Sanchéz-García, Á. J. (2023). Comprehension of Computer Programs Through Reverse Engineering Approaches and Techniques: A Systematic Mapping Study. In: Mejia J., Muñoz M., Rocha Á., Hernández-Nava V. (eds). New Perspectives in Software Engineering. CIMPS 2022. *Lecture Notes in Networks and Systems, 576*. Springer, Cham. https://doi.org/10.1007/978-3-031-20322-0_9

[2] Marinescu, R. (2012). Assessing technical debt by identifying design flaws in software systems. IBM J. Res. Dev. 56(5), 9:1–9:13. https://doi.org/10.1147/JRD.2012.2204512

[3] Jamshidi, P., Ahmad, A., & Pahl, C. (2013). Cloud Migration Research: A Systematic Review. *IEEE Transactions on Cloud Computing, 1*(2), 142–157. https://doi.org/10.1109/TCC.2013.10

[4] Andrikopoulos, V., Binz, T., Leymann, F., & Strauch, S. (2013). How to adapt applications for the Cloud environment: Challenges and solutions in migrating applications to the Cloud. *Computing, 95*(6), 493–535. https://doi.org/10.1007/s00607-012-0248-2

[5] Maji, A. K., Mitra, S., & Zhou, B. (2015). “Parallel Testing of Combinational Circuits in Linear Time”, 2015 33rd IEEE International Conference on Computer Design (ICCD), New York, NY, pp. 89–96. https://doi.org/10.1109/ICCD.2015.7357096

[6] Adnan, M., & Mirza, N. (2010). Document Analysis through Legacy System Reengineering. Data & Knowledge Engineering Lab (DKE), University of Engineering & Technology-Lahore

[7] Galbois, J., & Bournez, G. (2017). Technical Documentation Mining to Improve Legacy System Processes. *Control Engineering Practice, 62*, 59–164, ISSN 0967-0661
[8] Krishna, R., Alshayeb, M., Hattab, G., Zheng, Q., Chivers, M., & Lisitenko, D. (2021). Reverse engineering as a stepping stone to continuous software maintenance. *Software: Practice and Experience, 51*(10), 1914–1936. https://doi.org/10.1002/spe.2979
[9] Wu, Y., Brinkkemper, S., & Li, X. (2009). Balancing agility and structured methods for successful software engineering projects. *Empir Software Eng 14*, 450–471.
[10] Chin, L., & Sturer, J. (1998). Understanding and implementing the IBM unified process for system development. *IBM Systems Journal, 37*(4), 539–558.
[11] Rubin, K. S. (2012). Essential Scrum: A practical guide to the most popular agile process. Addison-Wesley.
[12] Ammann, P., & Offutt, J. (2008). Introduction to software testing. Cambridge University Press. https://doi.org/10.1017/CBO9780511809163
[13] IIBA (International Institute of Business Analysis), A Guide to the Business Analysis Body of Knowledge (BABOK Guide), 3rd ed., International Institute of Business Analysis, Toronto, Ontario, Canada, 2015.
[14] DiMarzio, J. F. (2011). "Parallel Testing of Cloud-Based Applications", 2011 IEEE Ninth International Symposium on Parallel and Distributed Processing with Applications Workshops, Busan, pp. 253–257. https://doi.org/10.1109/ISPAW.2011.49
[15] Goseva-Popstojanova, K., Guedem, A., & Singh, A. D. (2016). "Cost-Effective Software Testing in the Cloud", 2016 IEEE 9th International Conference on Software Testing, Verification and Validation Workshops (ICSTW), Chicago, IL, pp. 330–335. https://doi.org/10.1109/ICSTW.2016.25
[16] Petrenko, A., Schlingloff, H. & Ulrich, A. (2005). "Black-box testing of communicating systems", 5th International Conference on Application of Concurrency to System Design, St. Malo, France, 164–173. https://doi.org/10.1109/ACSD.2005.20
[17] 17 Eagle, C., & Younan, Y. (2021). Reverse Engineering with the Ghidra Platform. *Synthesis Lectures on Information Security, Privacy, and Trust, 13*(1), 1–271
[18] Lakes, R., Müller, D., & Kim, N. (2021). Extracting Population Dynamics Insights from Google Trends on Pandemics. PLoS ONE, 16(5), e0251867. https://doi.org/10.1371/journal.pone.0251867
[19] Kumar, R., & Upreti, G. (2019). Sentiment Analysis using Machine Learning Techniques: A Comparative Analysis. In 2019 4th International Conference on Information Systems and Computer Networks (ISCON), 254–259. https://doi.org/10.1109/ISCON47742.2019.8987091

O. B. Kernytskyy, V. M. Teslyuk

Lviv Polytechnic National University, Lviv, Ukraine

THE SYNTHESIS METHOD FOR SPECIFICATIONS AND REQUIREMENTS IN THE PROCESS OF IT PROJECT REENGINEERING

In this study, the aim is to create and improve a methodology for synthesizing requirements and specifications for the re-engineering of IT projects with maximum efficiency and business orientation. The main task is to adapt outdated IT systems to the changing technical environment, in particular to cloud technologies and security system requirements. To achieve these goals, the proposed methodology uses the analysis of archaic systems, the reverse engineering method, expert surveys, documentation analysis, and black-box modeling. The application of these methods allows for the identification and revision of requirements and specifications, ensuring a high level of quality and efficiency in the process of re-engineering IT projects.

The article further discusses the practical aspects of applying the methodology, prospects for further development, and the peculiarities of using various statistical methods in the process of improving re-engineering results. The operating principles of the method are described along with the main approaches and techniques that promote the analysis of existing IT systems, the synthesis of requirements and specifications, quality control, and successful project implementation. The individual components of the method include the collection of data about the existing system and the analysis of archaic systems to restore the definition of requirements. The use of the black-box model for testing the developed system is discussed, including the analysis of the obtained results, correction of requirements, and improvement of specifications.

The methodology includes documentation analysis tools, reverse engineering, surveys and data visualization tools, as well as analytical techniques such as a formula for parallel testing, a formula for requirement traceability matrix, and a formula for forecasting requirements based on discrepancy rate analysis. As a result of implementing the IT project reengineering method, successful transition from old to new technologies can be achieved, the IT industry can be optimized, and conditions can be created for adaptation to modern technical environments, ensuring stability and reliability of the implemented reengineering projects. Based on the analysis of modern sources, previous experience, and conducted research, it can be asserted that the method for synthesizing specifications and requirements in the process of reengineering IT projects is of great importance and relevance for the modern development of information technology and business processes.

Keywords: IT project reengineering; Archaic systems; Parallel testing; Requirement Traceability Matrix; Discrepancy rate analysis;, Cloud technologies.

Інформація про авторів:

Теслюк Василь Миколайович, д-р техн. наук, професор, завідувач кафедри автоматизованих систем управління.
Email: vasyl.m.teslyuk@lpnu.ua; https://orcid.org/0000-0002-5974-9310

Керницький Олег Богданович, аспірант, кафедра автоматизованих систем управління.
Email: oleh.b.kernytskyi@lpnu.ua; https://orcid.org/0009-0007-5318-6506

Цитування за ДСТУ: Керницький О. Б., Теслюк В. М. Метод розроблення специфікацій та вимог в процесі реінжинірингу ІТ-проєктів. *Український журнал інформаційних технологій.* 2023. Т. 5, № 2. С. 01–08.

Citation APA: Kernytskyy, O. B., & Teslyuk, V. M. (2023). The synthesis method for specifications and requirements in the process of it project reengineering. *Ukrainian Journal of Information Technology*, 5(2), 01–08. https://doi.org/10.23939/ujit2023.02.001