UDC 004.41

Михайло ЛИТВИНОВ

аспірант кафедри електронних обчислювальних машин, Дніпровський національний університет імені Олеся Гончара, пр. Науки, 72, м. Дніпро, Україна, 49000, litvinovma0@gmail.com

ORCID: *0009-0000-9765-1501*

Володимир ГЕРАСИМОВ

Кандидат технічних наук, доцент, завідувач кафедри комп'ютерних наук та інформаційних технологій, Дніпровський національний університет імені Олеся Гончара пр. Науки, 72, м. Дніпро, Україна, 49000, gerasimov@dnu.dp.ua

ORCID: *0000-0002-1366-715X*

Scopus Author ID*:* *57191378683*

Бібліографічний опис статті: Литвинов, М., Герасимов, В. (2026). Операційно-орієнтована модель та метод оцінювання трудовитрат архітектурної міграції між варіаціями Domain-Driven Design. *Information Technology: Computer Science, Software Engineering and Cyber Security*

ОПЕРАЦІЙНО-ОРІЄНТОВАНА МОДЕЛЬ ТА МЕТОД ОЦІНЮВАННЯ ТРУДОВИТРАТ АРХІТЕКТУРНОЇ МІГРАЦІЇ МІЖ ВАРІАЦІЯМИ DOMAIN-DRIVEN DESIGN

Мета роботи *полягає в підвищенні об'єктивності та точності оцінювання трудовитрат під час процедур архітектурної міграції в межах предметно-орієнтованого проєктування, зокрема, переходу від анемічної до збагаченої доменної моделі шляхом розробки комплексного методу оцінки на основі операційної складності, який дозволяє усунути суб'єктивність традиційних оцінок у людино-годинах та вплив людського фактору.*

Методологія. *Метод базується на моделі операційної складності міграції бізнес-логіки, що передбачає введення двох категорій операцій міграції (введення нової (N) та перенесення існуючої логіки (M)), класифікації базових операцій міграції бізнес-логіки, архітектурної класифікації бізнес-логіки за типами сутностей та розподілення операцій міграції по трьом шарам (Application, Domain, Infrastructure). Розширення методу Fuzzy Use Case Size Point шляхом додавання розширеної моделі розрахунку Unadjusted Use-Case Size Points (xUUSP) з ваговими параметрами., та адаптивне калібрування ваг за допомогою модифікації методу градієнтного спуску.*

Наукова новизна *полягає у розробці операційно-орієнтованої моделі оцінки трудовитрат, та методу оцінювання трудовитрат міграції між варіаціями DDD, який базується на розширеній моделі розрахунку Unadjusted Use-Case Size Points (xUUSP) з ваговими параметрами, розробці методу адаптивного калібрування ваг за допомогою модифікованого методу градієнтного спуску.*

Висновки. *Завдяки цьому розроблений метод забезпечує вищу об'єктивність і точність: на тестовому наборі з 53 прецедентів використання у двох обмежених контекстах (Bounded Contexts) досягнуто MMRE < 0.20 та PRED(0.25) ≥ 75–100% (залежно від класу сутностей), що перевищує стандартні пороги точності передбачення. У практиці метод може використовуватися для прогнозування трудовитрат, планування ризиків та контрольованої еволюційної міграції в корпоративних системах з бізнес-логікою побудованої на базі підходу DDD.*

Ключові слова: оцінка трудовитрат, архітектурна міграція, прогнозування, оптимізація, патерн.

Mykhailo LYTVYNOV

PhD Student, Department of Electronic Computing, Oles Honchar Dnipro National University, Nauky ave., 72, Dnipro, Ukraine, 49000, litvinovma0@gmail.com

ORCID: https://orcid.org/0009-0000-9765-1501

Volodymyr GERASIMOV
Candidate of Technical Sciences, Associate Professor, Head of the Department of Computer Science and Information Technologies, Oles Honchar Dnipro National University, 72 Nauky Ave., Dnipro, 49000, Ukraine, gerasimov@dnu.dp.ua
ORCID: *0000-0002-1366-715X*
Scopus Author ID: *57191378683*

To cite this article: Lytvynov, M., Gerasimov, V. (2026). Operatsiino-oriientovana model ta metod otsiniuvannia trudovytrat arkhitekturnoi mihratsii mizh variatsiiamy Domain-Driven Design [Operation-oriented model and method for effort estimation of architectural migration among Domain-Driven Design variations]. *Information Technology: Computer Science, Software Engineering and Cyber Security*

OPERATION-ORIENTED MODEL AND METHOD FOR EFFORT ESTIMATION OF ARCHITECTURAL MIGRATION AMONG DOMAIN-DRIVEN DESIGN VARIATIONS

Objective. *The objective of this article is to improve the objectivity and accuracy of software effort estimation for architectural migration within the Domain-Driven Design (DDD) approach, specifically the transition from the Anemic Domain Model to the Rich Domain Model, by developing a comprehensive effort estimation method based on operational complexity. The proposed method is intended to overcome the subjectivity of traditional person-hour estimation and reduce the influence of the human factor.*

Methodology*. The method is grounded in an operational complexity model of business logic migration, which introduces two categories of migration operations – new logic introduction (N) and existing logic relocation (M) – along with a classification of base business logic migration operations, architectural classification of business logic by entity types, and distribution of migration operations across three*

layers: Application, Domain, and Infrastructure. The method extends the Fuzzy Use Case Size Point approach by introducing an extended Unadjusted Use-Case Size Points calculation model (xUUSP) with weighted parameters, and incorporates adaptive weight calibration using a modified gradient descent method.

Scientific novelty. *The scientific novelty of this research lies in the development of an operation-oriented model and a method for estimating migration effort between Domain-Driven Design architectural variations. The proposed method is based on the extended Unadjusted Use-Case Size Points (xUUSP) model with weighted parameters and incorporates adaptive weight calibration using a modified gradient descent method.*

Conclusions. *As a result, the developed method provides higher objectivity and accuracy: on a dataset of 53 use cases across two Bounded Contexts, MMRE < 0.20 and PRED(0.25) ≥ 75–100% (depending on entity class) were achieved, exceeding standard prediction accuracy thresholds. In practice, the method can be applied to effort prediction, risk planning, and controlled evolutionary migration in enterprise systems with business logic built upon the Domain-Driven Design approach.*

Keywords: effort estimation, architectural migration, prediction, optimization, pattern.

Актуальність проблеми

Предметно-орієнтоване проєктування (DDD) (Evans, 2004) стало фундаментальним підходом до вирішення проблеми складності бізнес-доменів, пропонуючи широкий спектр архітектурних варіацій. Хоча деякі з варіацій добре задокументовані (Evans, 2004; Vernon, 2013; Martin, 2017), інші виникають як адаптації до конкретних обмежень та завдань проєкту. Ефективна архітектура надає змогу здійснювати контрольовану еволюцію, забезпечуючи більш ефективні рішення та запобігаючи «архітектурному дрейфу», що шкодить довгостроковій зручності супроводу (Ford et al., 2022).

Ця робота присвячена оптимізації процедур архітектурної міграції на основі аналізу їх операційної складності замість оцінювання зусиль в людино-

годинах, що залежать від «людського фактору», включаючи досвід команди та індивідуальні характеристики розробників. Це дозволяє здійснити кількісні оцінки трудовитрат та їх оптимізацію для архітектурних міграцій, керованих DDD, підвищити об’єктивність оцінки, зосередившись на структурній трансформації коду. Запропонований метод вимірює трудовитрати за допомогою метрик операцій пов'язаних з міграцією бізнес-логіки. Хоча подібні концепції оцінки бізнес-логіки були запропоновані в (Kamimura et al., 2015; Maeda et al., 2017), ці підходи орієнтовані на загальну оцінку складності коду та революційну міграцію, а не на операційно-орієнтоване вимірювання трудовитрат при еволюційній міграції в межах архітектурних варіацій DDD.

За основу взятий модифікований метод Fuzzy Use Case Size Point (Lytvynov et al., 2026), який був застосований для задачі прогнозування трудовитрат у людино-годинах і показав переваги порівняно з класичним методом USP. Пряме застосування методу (Lytvynov et al., 2026) для операційно-орієнтованого оцінювання не надало очікуваних результатів через відсутність специфічних метрик операційної складності та вдосконалення опису кроків.

Таким чином, дослідження, присвячене розробці методу для оцінки трудовитрат на основі операційної складності для DDD-міграцій, є актуальним.

Експериментальна валідація запропонованого методу базується на міграції від анемічної доменної моделі (ADM) до збагаченої доменної моделі (RDM). Це завдання було обрано, оскільки шар прикладної області (Domain) є кореневим шаром, який, згідно з класичним підходом, відповідає за фіксацію бізнес-знань та правил. Хоча класики підходу (Evans, 2004; Vernon, 2013) наполягають на використанні лише моделі RDM, тобто сутності-агрегати повинні інкапсульовувати поведінку, практикуючі фахівці часто віддають перевагу ADM через її простоту, легкість відображення DTO та тестованість. Так, використання ADM може бути доцільним, коли швидкість початкової реалізації важливіша за архітектурну чистоту (Khononov, 2021), а у ряді випадків практичним підходом є початок розробки саме з анемічної моделі з поступовим рефакторингом до збагаченої (Millett & Tune, 2015). Це часто призводить до необхідності міграції

на пізніх етапах розроблення проєкту або навіть під час його обслуговування. Оцінювання ризиків і трудовитрат такого переходу з використанням підходу, орієнтованого на операції, дозволяє здійснювати більш передбачуване управління еволюцією архітектури.

Аналіз останніх досліджень і публікацій

Під час розгляду питань міграції програмного забезпечення комп'ютерних систем необхідно розрізняти її різновиди, які суттєво впливають на вибір стратегії та обсяг трудовитрат з боку команди розробників. Можна визначити два базових типи міграції: еволюційну (Ford et al., 2022), що здійснюється в рамках архітектурного підходу, та революційну (Maeda et al., 2017), яка передбачає зміну архітектури і навіть парадигми. Для обох типів міграцій можуть мати місце технологічні та інфраструктурні зміни.

Параметричне визначення обсягу програмного забезпечення традиційно спирається на аналіз, пов'язаний з об'ємом вихідного коду. Проте дані методи, зокрема COCOMO II (Bajusova et al., 2024; Hjalmarsson et al., 2013) , базуються на оцінці розміру програмного проєкту, вимірюваного у тисячах рядків вихідного коду (далі KLOC). В умовах сучасної інженерії програмного забезпечення та еволюційної міграції, що розглядається у цій роботі, така залежність призводить до структурних похибок оцінювання:

Парадокс апріорного оцінювання. Прогнозування метрики KLOC із прийнятною для параметричних моделей похибкою вимагає вичерпного алгоритмічного визначення системи ще до її реалізації чи міграції. Проте в реальних проєктах застаріла або відсутня документація унеможливлює такий апріорний розрахунок (Hjalmarsson et al., 2013).

Генерація коду та обмеження оптимізаційних надбудов. Модель COCOMO II інтерпретує таке збільшення KLOC як пропорційне зростання трудовитрат, не відображаючи реального зміщення фокусу інженерів на складний когнітивний аналіз. Сучасні фреймворки та LLM-асистенти миттєво генерують тисячі рядків структурованого коду, що штучно роздуває метрику KLOC і викривляє

мультиплікативну базу параметричних методів. Навіть модифікації COCOMO II, оптимізовані еволюційними алгоритмами типу SOMA (Bajusova et al., 2024), виявляються інваріантними до походження коду.

Невірна оцінка семантичної складності коду. Фізична кількість рядків коду нівелює приховану когнітивну складність завдань. Масштабування на основі обсягу коду трактує ці принципово різні за складністю процеси як еквівалентні.

Як наслідок, зазначені недоліки зумовлюють пошук іншої бази відмінної від обсягу вихідного коду для оцінювання трудовитрат.

Серед підходів, що використовуються для розрахунку зусиль розробників, особливе місце займають методи оцінювання на основі специфікацій вимог (use cases). Вони вирізняються відносною простотою застосування оскільки не потребують детальних специфікацій програмних компонентів та алгоритмів як вхідних даних (Silhavy et al., 2021). З цієї позиції процес міграції програмного забезпечення комп'ютерних систем може розглядатися як інженерний процес, що дозволяє брати специфікації вимог як головне джерело інформації та застосовувати відповідні методи параметричного оцінювання трудовитрат розробників на базі метрик, орієнтованих на прецеденти (use case oriented metrics). Дані методи не вимагають надання специфікацій програмних компонентів та алгоритмів як вхідних даних, що обумовлює їх простоту.

Базовим методом є Use Case Points (UCP), проте для підвищення точності оцінювання застосовують його модифікації (Azzeh et al., 2021; Silhavy et al., 2021; Haryono et al., 2025). Зокрема, метод AUCSE (Silhavy et al., 2021) призначений для оцінки розміру систем на ранніх стадіях розробки з нуля і не передбачає сценаріїв міграції. Він базується на кількості акторів, прецедентів третього рівня складності (за шкалою складності UCP) та покроковій регресії. За висновками авторів це дозволило суттєво підвищити точність прогнозу порівняно з класичним UCP (MMRE = 0,05 проти 0,27) через спрощення моделі, а не через глибинний аналіз структури. AUCSE оперує зовнішніми характеристиками прецеденту та ігнорує внутрішню структуру кроків і їхню архітектурну прив'язку, що унеможливлює його застосування для операційно-орієнтованого

оцінювання трудовитрат, де ключовим є розрізнення операцій пов'язаних з міграцією в межах архітектурних компонентів: введення нової (N) та перенесення існуючої логіки (M).

У роботі (Azzeh et al., 2021) на вибірці зі 110 проєктів доведено, що гіпотетична пропорційна залежність між значеннями факторів оточення (EAF – Environmental Adjustment Factors) і фактичною продуктивністю команди не підтверджується для більшості з них. Сам метод (Azzeh et al., 2021) прогнозує продуктивність на основі кластеризації проєктів за середовищними факторами, але не оперує внутрішньою структурою прецедентів чи специфікою архітектурної трансформації, тому не може бути застосований для операційно-орієнтованого оцінювання трудовитрат міграції.

Підхід (Haryono et al., 2025) інтегрує два методи UCP з Activity-Based Costing, що покращило точність оцінки вартості розробки (відхилення від фактичних витрат становить 23,52% проти 33,35% для методу Adjusted Function Point, що є різновидом методом Function Point). Проте базова класифікація прецедентів здійснюється лише за кількістю транзакцій. Метод не враховує архітектурну специфіку цільової системи та не розрізняє операції перенесення існуючої логіки й створення нових компонентів, що унеможливлює його використання для операційно-орієнтованої оцінки міграції.

Незважаючи на свою корисність для оцінки трудовитрат на ранніх стадіях розвитку проєкту, ці моделі стикаються з наступними проблемами (Nhung et al., 2019), що заважають їх застосуванню для поставленої задачі:

– PRB1: Складність моделей оцінюється суб'єктивно, без урахування внутрішніх прихованих зв'язків.

– PRB2: Суб'єктивність у розрахунку коригувальних коефіцієнтів TAF (Technical Adjustment Factors – технологічні фактори впливу на складність) та EAF (Environmental Adjustment Factors – фактори операційного оточення).

– PRB3: Відсутність стандартизованої класифікації Use Case: велика кількість стилів специфікації прецедентів та різним ступенем формалізації.

– PRB4: Складність перетворення розміру в людино-години без історичних наборів даних.

У (Lytvynov et al., 2026) запропоновано метод, що вирішує PRB1, PRB3 і PRB4 для еволюційних DDD-міграцій: прецеденти формалізуються за набором строгих правил опису кроків, класифікуються за фреймами-класами для розрахунку Unadjusted Use Case Size Points, а розмір прецеденту уточнюється фазифікацією. Прогнозування трудовитрат у людино-годинах здійснюється через коефіцієнт перетворення FUSP, що обчислюється як історичне середнє (MMRE = 0,0343 проти 0,1094 для класичного FUSP). Проблема PRB2 стосується переважно революційних міграцій (Azzeh et al., 2021) і виходить за межі цього дослідження. Пряме застосування методу (Lytvynov et al., 2026) до операційно-орієнтованого оцінювання виявилося неможливим через відсутність задовільних метрик операційної складності міграції бізнес-логіки та відсутність урахування специфіки кроків прецеденту через параметри-фасети.

Найближчим за задачею операційної оцінки міграції бізнес-логіки є підхід (Maeda et al., 2017), який замінює оцінку в людино-годинах метрикою складності бізнес-логіки. Однак даний підхід орієнтований на автоматичний аналіз застарілого коду і революційну міграцію на мікросервісну архітектуру, а не на еволюційну міграцію в межах DDD, він не розрізняє операції перенесення та створення логіки, не враховує розподіл за архітектурними шарами й типами сутностей, а його метрики (кількість змінних і умовних операторів) недостатні для відображення специфіки міграції між варіаціями DDD.

Систематизацію результатів і локальних обмежень проаналізованих підходів для задачі DDD-міграції наведено в таблиці 1.

Таблиця 1

Систематизація обмежень розглянутих підходів

Джерело	**Рік**	**Результат**	**Локальна проблема для задачі DDD-міграції**
(Hjalmarsson et al., 2013)	2013	COCOMO II + EA-моделювання для оцінки вартості будь-якої міграції	Метод орієнтований на KLOC, має структурні обмеження COCOMO II. Не враховує

			архітектурну специфіку цільової системи.
(Bajusova et al., 2024)	2024	SOMA-оптимізація параметрів COCOMO II підвищує точність на стабільних датасетах	Орієнтований на KLOC і нову розробку. Оптимізація KLOC-бази не усуває її структурних обмежень для міграції зі складною бізнес-логікою.
(Nhung et al., 2019)	2019	Огляд UC-методів, формалізація PRB1–PRB4	Усі чотири проблеми залишаються актуальними за висновками огляду.
(Azzeh et al., 2021)	2021	Емпірично доведено ненадійність EF (PRB2) для прогнозування продуктивності (110 проєктів)	Робота зосереджена на продуктивності команди, не оперує внутрішньою структурою прецедентів.
(Silhavy et al., 2021)	2021	AUCSE: MMRE = 0,05 замість 0,27 за рахунок покрокової регресії (70 проєктів нової розробки)	Міграційні сценарії не розглядаються; суб'єктивність класифікації прецедентів (PRB1) обходиться, а не вирішується.
(Haryono et al., 2025)	2025	UCPabc: відхилення вартості 23,52 % проти 33,35 % (AFP)	Метод орієнтований на фінансову оцінку вартості; базова класифікація прецедентів незмінна відносно UCP.
(Lytvynov et al., 2026)	2026	Вирішено PRB1, PRB3, PRB4 для DDD-міграції; MMRE = 0,03	Відсутні метрики операційної складності; специфіка кроків не врахована, що ускладнює точність оцінки.
(Maeda et al., 2017)	2017	Метрика складності бізнес-логіки на основі таблиць рішень	Орієнтована на революційну міграцію; метрики на основі змінних/умовних операторів не відображають специфіку DDD-варіацій.

Аналіз (таблиця 1) показує, що параметричні методи на основі KLOC (Bajusova et al., 2024; Hjalmarsson et al., 2013) і методи на основі прецедентів (Silhavy et al., 2021; Azzeh et al., 2021; Haryono et al., 2025; Nhung et al., 2019) не враховують архітектурну специфіку цільової системи та не розрізняють операції перенесення й створення логіки, а підхід Maeda et al. (Maeda et al., 2017) орієнтований на революційну міграцію поза контекстом DDD. Єдиним підходом,

цілеспрямовано розробленим для оцінки DDD-міграції, є метод (Lytvynov et al., 2026), що вирішує PRB1, PRB3 і PRB4 з високою точністю (MMRE = 0,03), але потребує метрик операційної складності та фасетного аналізу кроків прецедентів. Отже, жоден із проаналізованих підходів не вирішує задачу операційно-орієнтованої оцінки еволюційної міграції бізнес-логіки між варіаціями DDD: параметричні методи спираються на обсяг коду, методи на основі прецедентів не розрізняють операції перенесення й створення логіки, а операційно-орієнтований підхід (Maeda et al., 2017) розрахований на революційну міграцію і не може бути застосованим до розглядаємого типу міграцій, а модель застосовуватися як основа формалізованої операційної моделі оцінки. Це зумовлює актуальність розробки формалізованої моделі операційної складності міграції бізнес-логіки та методу оцінювання трудовитрат еволюційної міграції в рамках підходу DDD.

Мета дослідження

Метою дослідження є розробка формалізованої моделі операційної складності міграції бізнес-логіки та методу оцінювання трудовитрат архітектурної міграції між варіаціями DDD. Для цього потрібно виконати наступні завдання:

– розробити формалізовану модель для оцінки операційної складності міграції бізнес-логіки;

– розробити метод для операційно-орієнтованого оцінювання, що враховує структуру кроків прецедентів та специфіку кроків з визначеними ваговими коефіцієнтами, та надає змогу їх автоматизованого калібрування;

– розробити та провести експерименти з порівняння розробленого методу з базовим методом (Lytvynov et al., 2026) адаптованим для операційно-орієнтованої оцінки міграції. Для здійснення експерименту доцільно вибрати задачу міграції системи з ADM до RDM. Вибір зумовлений виключною роллю шару прикладної області (domain layer), при цьому міграцію з ADM до RDM в межах DDD можна розглядати як натуральну фазу еволюції системи.

Виклад основного матеріалу дослідження

В роботі було прийнято наступні спрощення:

– Дослідження стосується тільки еволюційних міграцій варіацій в рамках підходу DDD і не стосується революційного типу міграцій;

– Типовий проєкт, розглянутий в експерименті, належить до широкого класу систем (наприклад, медичні, виробничі системи), але не є прототипом високопродуктивних або спрямованих на інтенсивну обробку даних систем;

– Під час проведення експерименту розглядається тільки клас простих прецедентів (53 з 89), що відносяться до класу модифікації.

Базовий метод оцінки трудовитрат міграції

Базовий метод оцінки трудовитрат міграції (Lytvynov et al., 2026) складається з чотирьох послідовних етапів.

На першому етапі здійснюється формалізація прецедентів (вирішення PRB1) через упровадження правил їх строгого визначення, адаптованих до специфіки DDD: принципу єдиної відповідальності кроків, декомпозиції логіки, ідентифікації альтернативних потоків і неявних прецедентів, а також зв'язку кроків із бізнес-правилами та доменними сутностями.

На другому етапі передбачаються фреймова класифікація та оцінювання прецедентів (вирішення PRB3) за чотирма класами, де на основі кількості кроків і ваг слотів розраховується показник Unadjusted Use Case Size Points (UUSP).

На третьому етапі для подолання обмежень дискретної класифікації на межах категорій застосовується фазифікація за допомогою трапецієподібної функції належності, що дозволяє обчислити точніше значення Fuzzy Use Case Size Point (FUSP).

На четвертому етапі забезпечується прогнозування трудовитрат (вирішення PRB4) через коефіцієнт архітектурної складності γ (щільність операцій на одиницю FUSP), формуючи на основі історичних даних оптимістичний (γ_{min}), песимістичний (γ_{max}) та середній (γ_{avg}) прогнози.

Повний опис методу і розрахункові приклади надано в первинному джерелі (Lytvynov et al., 2026) та репозиторії (Lytvynov, 2026).

Метод фазифікації та розрахунок FUSP

Метод оцінювання в базовому методі має проблеми масштабування, коли прецеденти використання перевищують 16 кроків + кількість сутностей. Для вирішення цієї проблеми в рамках даної роботи таблицю класифікації (табл. 2) було розширено додатковими рівнями складності, що забезпечує більш деталізований розрахунок FUSP для великомасштабних міграцій.

Таблиця 2

Класифікація рівнів складності

Складність	Кількість кроків+сутностей	a	b	c	d	USP
Very simple	[1;5]	–	1	3.5	6	4
Simple	[6;10]	3.5	6	8.5	11	6
Average	[11;15]	8.5	11	13.5	16	8
Complex (1)	[16;20]	13.5	16	18.5	21	12
Complex (2)	[21;25]	18.5	21	23.5	26	16
Complex (3)	[26;30]	23.5	26	28.5	31	20
Very complex (1)	[31;35]	28.5	31	33.5	36	24
Very complex (2)	[36;∞)	33.5	36	38.5	∞	28

Для розрахунку FUSP використано трапецієподібну функцію належності, яка описується формулою (1):

$$\mu_A(x) = \begin{cases} 0, x \le a \\ 0, x \ge d \\ 1, b < x \le c \\ \dfrac{x-a}{b-a}, a < x \le b \\ \dfrac{d-x}{d-c}, c < x \le d \end{cases}. \quad (1)$$

Параметри a та d визначають мінімальну та максимальну межі категорії складності відповідно, тоді як b та c задають інтервал, у межах якого прецедент

використання повністю належить до цієї категорії. Значення, що знаходяться між сусідніми категоріями, можуть одночасно належати до декількох нечітких множин з різними ступенями належності.

Формальна модель оцінки операційної складності міграції бізнес-логіки в межах предметно-орієнтованого підходу

Запропонований підхід орієнтований на міграцію вже існуючої логіки без урахування розробки нового функціоналу, тобто зміни системних вимог або додавання нових функцій до ПЗ в момент міграції не розглядаються. Даний підхід ураховує когнітивне навантаження розробника під час міграції, що відрізняє його від класичних метрик, орієнтованих виключно на зміни коду.

Визначимо фрейм прецеденту використання $U_i \subseteq U$ як клас прецедентів використання, адаптованих до конкретних архітектурних вимог. Цей фрейм пов'язаний з архітектурними елементами, які існують у глобальній структурі системи, але по-різному використовуються різними архітектурними варіаціями.

Відповідно до принципів теорії моделей, архітектурна модель $\mathfrak{A}$ визначається як структура, що забезпечує конкретну реалізацію абстрактних символів фрейму прецеденту використання.

Визначення. Архітектурна модель $\mathcal{A}_A$ для множини фрейму прецеденту використання U є впорядкованою парою $A, \mathcal{T}$, де $A \in \mathrm{A}$ – непорожня множина архітектурних компонентів, пов'язаних із варіацією A узагальненої архітектури (область реалізацій), а $\mathcal{T}$ – функція інтерпретації, така що: якщо $s \in U_i$ є слотом фрейму прецеденту використання U_i (наприклад, Validate command, Check business rules тощо), то $\mathcal{T}(s)$ відображає його на комбінацію компонентів із A, включаючи операційну логіку, сутності, винятки, події тощо.

Таким чином, в рамках еволюційної міграції перехід від однієї варіації до іншої: анемічної доменної моделі до збагаченої доменної моделі формально є зміною архітектурної моделі $\mathcal{A}_{ADM} \to \mathcal{A}_{RDM}$ через перевизначення функції інтерпретації $\mathcal{T}$ та базової множини компонентів A. Наприклад, якщо $\mathcal{T}_{ADM}(s)$

у моделі A_{ADM} відображає слот $s \in U_i$ на метод класу Handler, то $\tau_{RDM}(s)$ у моделі A_{RDM} відображає цей самий слот на метод Aggregate.

Функціональна поведінка системи залишається інваріантною з точки зору контракту функціональних вимог, оскільки в обох випадках реалізується $s \in U_i$, однак внутрішня реалізація проходить трансформацію $\tau_{ADM} \to \tau_{RDM}$.

Простий диференціальний підхід визначає різницю трудовитрат між двома реалізаціями за формулою (2):

$$\Delta E_D(U_i) = \left| Cost\left(\tau_{ADM}(U_i)\right) - Cost\left(\tau_{RDM}(U_i)\right) \right|. \quad (2)$$

Однак ця функція обчислює лише різницю між двома реалізаціями, створеними «з нуля», і не враховує специфічну операційну складність міграції вже існуючого рішення.

Для коректної оцінки міграції вводиться функція міграції $\psi : \tau_{ADM} \to \tau_{RDM}$, яка забезпечує перехід між інтерпретаціями. Цей перехід може бути описаний як композиція двох функцій: ψ_N, що відповідає за створення нової логіки відповідно до вимог цільової архітектурної варіації, та ψ_M, яка відображає перенесення (рефакторинг і переміщення) вже існуючої логіки.

$$\psi_N \circ \psi_M : \tau_{ADM} \to \tau_{RDM}; \quad (3)$$

$$\Delta E_M(U_i) = Cost\left(\psi_N(U_i)\right) + Cost\left(\psi_M(U_i)\right). \quad (4)$$

Формули (2)–(4) приводять до двох ключових висновків щодо архітектурної трансформації:

– Неатомарність міграції. Міграція прецеденту використання не є атомарною операцією. Вона являє собою складений процес, що включає два взаємопов'язані підпроцеси – перенесення існуючої логіки (ψ_M) та реалізацію нових архітектурних шаблонів (ψ_N), кожен з яких має власну функцію вартості.

– Недостатність delta-метрик. Загальні трудовитрати міграції (ΔE_M) не можуть бути обчислені лише як різниця в обсязі коду або розмірі між вихідною та цільовою архітектурними варіаціями. Такий delta-підхід (ΔE_D) не враховує операційні витрати на реструктуризацію та когнітивне навантаження.

Таким чином, у контексті розглядуваного типу міграції кодова база поділяється на три категорії: існуюча логіка, яка залишається без змін; існуюча логіка, що переноситься та адаптується до нового архітектурного компонента (наприклад, з application service до aggregate root); новий код, необхідний для підтримки інфраструктури або інваріантів цільової архітектури (наприклад, domain events або спеціалізованих валідаторів).

Для врахування когнітивних трудовитрат розробника було запропоновано класифікацію базових операцій міграції з врахуванням категорій-вимірів M та N (табл. 3). Слід відзначити, що дані операції міграції часто поєднані. Наприклад, створення агрегатів на основі існуючих сутностей включає: перенесення логіки із application handlers (M) в нові методи агрегатів (N). З операціями зв'язані ваги, що відображають їхню відносну складність та когнітивне навантаження, але ваги слід визначати окремо для M та N, оскільки створення нової логіки за трудовитратами відрізняється від адаптації вже існуючої.

Таблиця 3

Класи базових операцій міграції бізнес-логіки

Операція	Вага	Клас операцій міграції	Приклад реалізації
Створення нового класу	3	*N*	Створення валідатора домену
Введення перевірки умови з винятком	3	*M/N*	Перевірка версії або забезпечення інваріантів
Створення нового інтерфейсу	2	*N*	Створення інтерфейсу валідатора домену
Введення перевірки умови	2	*M/N*	Логіка на основі правил (наприклад, право на бонус)
Створення нового методу	2	*N*	Інкапсуляція поведінки всередині агрегату
Цикл	2	*M/N*	Ітерація по дочірнім сутностям домену
Виклик запиту до колекції	2	*M/N*	Запит до внутрішньої колекції
Зміна властивості	1	*M/N*	Пряме оновлення стану (наприклад, зміна пароля)

Виклик методу / Отримання властивості	1	*M/N*	Виклик методу валідації або читання стану (наприклад, властивості команди)
Створення екземпляра класу	1	*M/N*	Створення екземпляра події домену

Метою такої класифікації є порівняння кількості операцій міграції M та N категорій, що дозволяє формально описати трудовитрати як співвідношення між трансформованою логікою та введенням нової логіки. Таким чином, загальні трудовитрати для ***i***-го прецеденту можна визначити як кортеж оцінок нової та перенесеної логіки (5-7):

$$E_i = \left(E_{i,N}, E_{i,M}\right), \tag{5}$$

де

$$E_{i,N} = \sum_{j \in C_N} \left(w_j * Op_{i,j,N}\right); \tag{6}$$

$$E_{i,M} = \sum_{k \in C_M} \left(w_k * Op_{i,k,M}\right). \tag{7}$$

У цій моделі C_N та C_M – категорії операцій, визначені в таблиці 3, а w – ваги, призначені кожній категорії.

Для формального оцінювання трудовитрат необхідно врахувати варіативність процесу отримання даних та розподілення операцій по шарам бізнес-логіки за якими відбувається модифікація (Application (a), Domain (d), Infrastructure (i)). У збагаченій доменній моделі існують три основні сценарії переходу від сутності до агрегату: Aggregate (A) – агрегат містить вкладені сутності; Entities (E) – логіка, що в анемічній доменній моделі працювала з окремою сутністю, у збагаченій доменній моделі повинна звертатися до кореня агрегату; Single Aggregates (SA) – окремі об’єкти без вкладених сутностей, для яких складність міграції логіки вибору даних є мінімальною.

Через таку структурну різноманітність точне визначення операцій міграції неможливе без поділу домену на ці класи. Крім того, логіка перевірки правил (d.chk.a-rules), модифікації (d.modify) та перевірки цілісності даних

(d.chk.integrity) мають різну складність залежно від типу агрегату. Тому процес міграції поділено на три базові категорії: Aggregate – комплексні агрегати, Single Aggregate – агрегати без вкладених сутностей, Entity – сутності.

Таким чином, запропонована модель операційної складності міграції бізнес-логіки формально визначається як кортеж (8):

$$OM = \langle O,\ L,\ ET,\ W \rangle, \tag{8}$$

де:

$O = \{M, N\}$ – категорії-виміри операцій міграції;

$L = \{a, d, i\}$ – де a – Application layer, d – Domain layer, i – Infrastructure layer. Описує архітектурні шари, між якими розподіляються операції міграції;

$ET = \{A, SA, E\}$ – множина класів сутностей предметної області (Aggregate, Singular Aggregate, Entity), що визначають структурну різноманітність домену і впливають на склад та складність операцій міграції;

$W = \{w_1, w_2, ..., w_n\}$ – множина вагових коефіцієнтів операцій (табл. 3), що відображають відносну складність та когнітивне навантаження на розробника.

Операційна складність міграції C прецеденту UC_i є сумою складності міграції, пов'язаної з кроками прецеденту (9):

$$C(UC_i) = \sum_{k=1}^{p_i} C(s_{k,i}), \tag{9}$$

де $s_{k,i} \in UC_i$ означає k-ий крок у i-му прецеденті, p_i – кількість кроків i-го прецеденту.

Складність міграції кроку прецеденту визначається як зважена сума атомарних операцій у просторі $O \times L \times ET$:

$$C(s_{k,i}) = \sum_{j=1}^{n} w_j \cdot o_{j,k,i}, \tag{10}$$

де $o_{j,k,i} \in O \times L \times ET$ означає j-ту операцію, застосовану до міграції k-го кроку i-го прецеденту.

Для аналізу процесу міграції доцільно розглядати не просто загальну складність міграції, а її розподілення по вимірам таким чином:

$$C_\omega(UC_i) = \pi_\omega(C(UC_i)), \tag{11}$$

де π означає операцію проекції, $\omega \in O$ означає визначений клас операцій міграції. Так, наприклад, $C_M(UC_i)$ визначає складність міграції, пов’язану з перенесенням логіки. При цьому цінну інформацію може надати розгляд розподілення цих операцій за шарами:

$$C_{\omega,\lambda}\left(UC_i\right) = \pi_\lambda\left(C_\omega\left(UC_i\right)\right) = \pi_\lambda \circ \pi_\omega\left(C\left(UC_i\right)\right), \tag{12}$$

де $\lambda \in L$ означає один з шарів бізнес-логіки. Наприклад, $C_{M,a}(UC_i)$ визначає складність пов'язану з перенесенням логіки у шар a – Application layer.

Така багатовимірна структура моделі дозволяє формально описати патерн міграції через співвідношення між трансформованою (M) та новою (N) логікою в розрізі архітектурних шарів і класів сутностей.

Хоча більшість трудовитрат можна обчислити детерміновано, окремі активності, зокрема правила, що потребують додаткових джерел даних, правила, що передбачають використання зовнішніх сервісів та основна логіка прецеденту використання, зміна стану даних, можуть містити складну бізнес-логіку, яку неможливо безпосередньо звести до атомарних операцій. Наприклад, модифікація властивостей сутності може залежати від умовної логіки, пов'язаної з податками, бонусами чи правилами знижок. Точне оцінювання таких випадків вимагало б надто детальної формалізації структури прецеденту використання, що зробило б модель занадто складною для практичного використання.

Для вирішення цієї проблеми пропонується підхід на базі прогнозування з урахуванням історії, що базується на коефіцієнті архітектурної складності $\overline{\Gamma}_{i,q}$, який відображає ефективність архітектурного патерну через щільність логіки (операцій) на одиницю архітектурної складності FUSP:

$$\overline{\Gamma}_{i,q} = \frac{\sum_{k=0}^{i-1} E_{k,q}}{\sum_{k=0}^{i-1} FUSP_k}, \tag{13}$$

де:

– $FUSP_k$ – значення Fuzzy Use Case Size Point для k-го прецеденту використання, яке можна розглядати як міру складності прецеденту використання;

– $q \in \{M, N\}$ – клас трудовитрат (нова логіка або перенесена логіка);

– $E_{k,q}$ – кількість операцій класу $q \in \{M, N\}$ необхідних для реалізації *k*-го прецеденту використання.

На основі цих історичних параметрів формується прогностична оцінка трудовитрат для поточного прецеденту використання:

$$\hat{E}_{avg,i,q} = FUSP_i * \bar{\Gamma}_{i,q}. \quad (14)$$

Варіанти оптимістичного та песимістичного прогнозу у цій роботі не розглядаються.

Розробка розширеної моделі розрахунку Unadjusted Use-Case Size Points на основі базового методу

Розширена модель розрахунку Unadjusted Use-Case Size Points (далі xUUSP) орієнтована на вирішення проблем точності оцінки складності прецеденту шляхом введення додаткових параметрів.

$$xUUSP_i = \sum_{k=1}^{s} \left(b_k \cdot w_b + p_k \cdot w_p + r_{s,k} \cdot w_{s,r} + r_{a,k} \cdot w_{a,r} + r_{c,k} \cdot w_{c,r} + d_k \cdot w_d \right), \quad (15)$$

де:

s – кількість слотів у фреймі прецеденту використання;

w_b – вага функціонального кроку (b_k = 3.0 для пар condition-action, b_k = 1.0 – для звичайних кроків, b_k = 0 – для порожніх слотів та у випадку виконання умови $((r_{s,k} > 0) \vee (r_{a,k} > 0) \vee (r_{c,k} > 0))$.

w_p – вага інформаційної місткості (*pk* – кількість полів/властивостей у *k*-му слоті);

$w_{s,r}$ та $r_{s,k}$ – вага і кількість простих правил, $w_{a,r}$ та $r_{a,k}$ – вага і кількість правил середньої складності, $w_{c,r}$ та $r_{c,k}$ – вага і кількість складних правил;

w_d – вага залежностей (d_k – кількість залежних сутностей у слоті).

Розробка методу автоматизованого калібрування вагових параметрів моделі.

Для автоматизації визначення вагових коефіцієнтів w_b, $w_{s,r}$, w_d, $w_{a,r}$, $w_{c,r}$ були використані два методи: GridSearch та модифікація методу градієнтного спуску. Метою було отримати перший набір параметрів, який задовольняє умову PRED(0.25). Для досягнення цієї мети у спеціалізованому програмному забезпеченні xArchEvaluator (Lytvynov, 2026) було розроблено гібридний алгоритм.

Таким чином, задачу калібрування було визначено як мінімізацію MMRE, що розглядається як функція втрат (Loss Function) $L(w)$ у заданому діапазоні ваг:

$$L(\vec{w}) = min\left(MMRE\left(w_{s,r},\ w_{a,r},\ w_{c,r}, w_d\right)\right), \tag{16}$$

де $\vec{w} = \left\{w_{s,r},\ w_{a,r},\ w_{c,r}, w_d\right\}$ – вектор ваг архітектурної складності.

Для прискорення пошуку повільного та малоефективного методу GridSearch було реалізовано гібридний алгоритм оптимізації. Метод поєднує градієнтний спуск із моментом та евристичний пошук із бектрекінгом. Такий підхід був зумовлений нелінійною, дискретною та «сходинкоподібною» природою поверхні архітектурних метрик.

Калібрування було визначено як багатокритеріальну задачу оптимізації. MMRE використовувалася як основна функція втрат (Loss Function) $L(\vec{w})$ для обчислення градієнта, тоді як максимізація PRED(0.25) виступала критерієм функціональної придатності. Алгоритм ітеративно оновлював вектор ваг у напрямку антиградієнта функції втрат:

$$\vec{v}_t = \gamma \vec{v}_{t-1} - \eta \nabla L\left(\vec{w}_{t-1}\right), \tag{17}$$

$$\vec{w}_t = \vec{w}_{t-1} + \vec{v}_t, \tag{18}$$

де:

$\vec{w}_t$ – вектор ваг $\left\{w_{s,r}, w_{a,r}, w_{c,r}, w_d\right\}$ на ітерації t; $\vec{v}_t$ – вектор швидкості (momentum), що накопичує напрямок і величину попередніх оновлень; η – який визначає вплив поточного градієнта на оновлення − для стабілізації пошуку на пізніх ітераціях η не є фіксованою константою, а лінійно затухає, тобто розмір кроку на ітерації t η_t розраховується як η_0-$(1-t/T)$, де η_0 – базовий розмір кроку,

T – параметр згасання (у експериментах T=40), що забезпечує поступове згасання кроку та стабілізацію ваг на завершальних ітераціях пошуку.; ∇L – вектор чисельних часткових похідних функції втрат (Loss Function) MMRE, обчислений методом центральних різниць із кроком $\Delta = 0.5$.

Через квантування операційних даних градієнт часто зникає ($\nabla L \approx 0$) на пласких ділянках поверхні метрик(плато). Для розв'язання цієї проблеми алгоритм виконує дискретні фіксовані стрибки, найпростіший – у напрямку зростання ваги:

$$w_i = w_i + \delta, if \left| \frac{\partial L}{\partial w_i} \right| < m. \quad (19)$$

Для забезпечення стабільності моделі та запобігання деградації результатів у окремих контекстах було використано механізм глобальної пам'яті найкращих станів (Global Best–State Memory), що базується на такій специфіці:

– Пам'ять найгіршого прогнозу (Record Keeping). Алгоритм відстежує $PRED_{min}$(найгіршу точність прогнозування) серед усіх контекстів домену.

– Повертання (Backtracking). Якщо нова ітерація призводить до зменшення $PRED_{min}$, або тривалого застою без покращення, система повертається до найкращої конфігурації ваг $\vec{w}_{best}$.

– Ортогональне дослідження (Orthogonal Exploration). Після повернення до «рекордної» точки алгоритм виконує пошук в альтернативних напрямках уздовж ортогональних осей простору параметрів, а в разі вичерпання всіх напрямків робиться випадкове зміщення в розширеному діапазоні навколо найкращої точки (аналог м'якого перезапуску пошуку) для знаходження перспективнішої траєкторії спуску.

Метрики оцінки точності прогнозування

Для оцінювання точності прогнозування трудовитрат використовувалися метрики середнє значення відносної помилки (MMRE) та відсоток прогнозування в межах x% (PRED(x)) (Nhung et al., 2019), задано формулами (20)–(22), які є двома стандартними метриками, що використовуються в

програмній інженерії. Обидва параметри базуються на величині відносної помилки (MRE). Стандартне відхилення похибки прогнозування (20) у цій роботі використовується як допоміжна метрика. На відміну від MMRE, яка відображає середню величину похибки, σ_δ дозволяє оцінити стабільність прогнозування.

Стандартне відхилення (StdDev) похибки прогнозування

$$\sigma_\delta = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(\delta_{avg,i} - \mu_\delta\right)^2} \ , \tag{20}$$

де $\mu_\delta = \frac{1}{n}\sum_{i=1}^{n}\delta_{avg,i}$ – середня величина.

Середнє відносне відхилення (Mean of Magnitude of Relative Error (MMRE))

$$MMRE_\delta = \frac{1}{n}\sum_{i=1}^{n}\left|\delta_{avg,i}\right| \ , \tag{21}$$

$$PRED(x) = \frac{1}{n}\sum_{i=1}^{n}\begin{cases}1, if \left|\delta_{avg,i}\right| \le x \\ 0, otherwise\end{cases} . \tag{22}$$

Згідно з (Dolado, 2001) , точна модель оцінювання трудовитрат повинна задовольняти умовам MMRE ≤ 0.25 (середня похибка оцінювання не перевищує 25%) та PRED(0.25) ≥ 0.75 (щонайменше 75% прогнозованих значень мають MRE < 0.25).

Методологія експерименту

Порівняльна оцінка ефективності моделей ADM та RDM досягається шляхом квантування та декомпозиції процесів міграції між двома ізольованими обмеженими контекстами (далі Bounded Contexts) у межах цієї системи. Оскільки сучасні корпоративні архітектури складаються з множини автономних контекстів, які взаємодіють через механізми інтеграційних подій та API, у роботі застосовується покрокова ітераційна стратегія міграції, яка складається із шести послідовних етапів, серед яких не підлягають оцінюванню: початкова підготовка інфраструктури (етап 1); процеси вибору набору функцій (етап 2); визначення обмеженого контексту (етап 3), а також безпосереднього вибору функцій із декомпозицією на конкретні прецеденти й блоки логіки (етап 4). На етапі 5, що

охоплює міграцію вибраних функцій виконується оцінювання складності на базі моделі оцінки операційної складності міграції бізнес-логіки з застосуванням розширеної моделі розрахунку xUUSP, методів GridSearch та модифікації методу градієнтного спуску для автоматизованого калібрування вагових параметрів моделі та метрик оцінки точності прогнозування. Фінальний етап 6 (масштабування та перенесення решти функцій за сформованими шаблонами) передбачає аналогічний до п'ятого етапу підхід до оцінювання.

У даному дослідженні Representative Test Project (RTP) включав 89 прецедентів використання системи Technical Station Management System, розподілених між двома обмеженими контекстами: Service та Inventory. Дослідження обмежене двома обмеженими контекстами, оскільки логіка їхньої взаємодії залишається незмінною для обох архітектурних варіантів. Логіка запитів (Get) виключена з оцінювання, оскільки read-only операції зазвичай не потребують складних перетворень. Основна увага була приділена прецедентам, орієнтованим на створення, зміну та видалення об'єктів, зокрема, прецедентам класу Modify, які стали базою для проведення експерименту. Розподіл прецедентів використання наведено в табл. 4. Структура доменного шару з визначенням кардинальності зв’язків та типів сутностей, які використовуються як основа для класифікації прецедентів представлена на рис. 1. Саму класифікацію наведено в табл. 5. Повний перелік 53 прецедентів типу Modify репрезентативного тестового проєкту надано у репозиторії (Lytvynov, 2026).

Таблиця 4

Розподіл прецедентів використання

Обмежений контекст	Всього	Add	Remove	Modify
Inventory	45	7	7	31
Service	44	11	11	22
Total	89	18	18	53

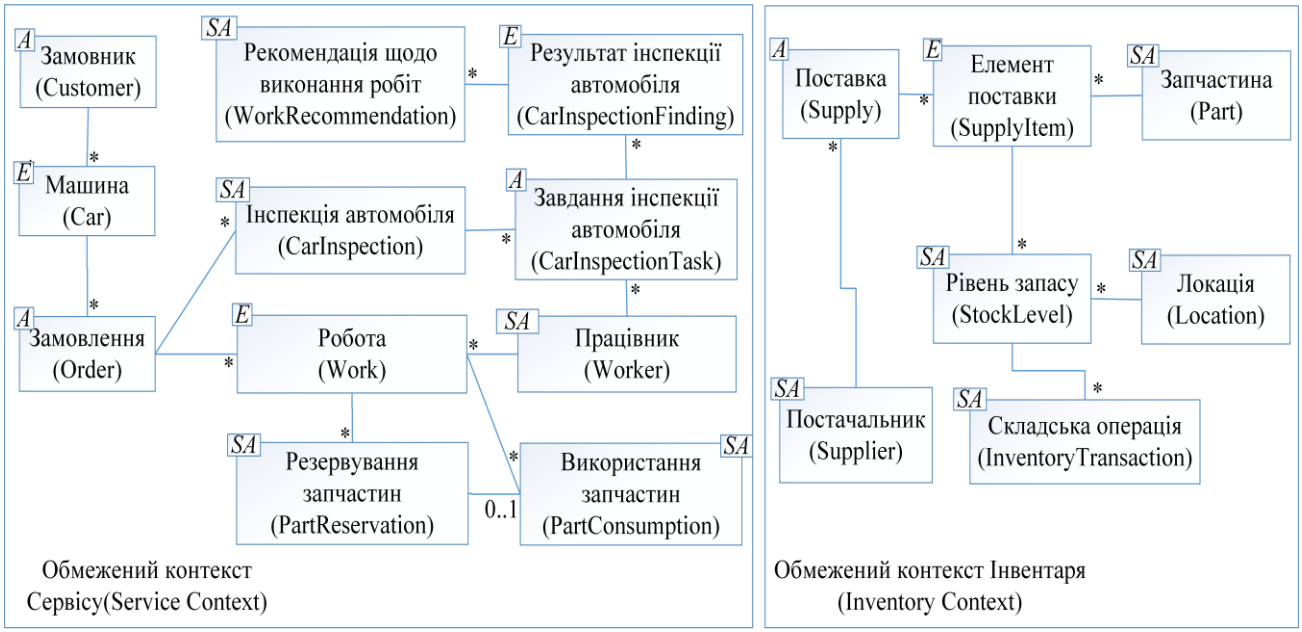

Рис. 1. Структура шару Domain

Таблиця 5

Сутності обмеженого контексту

BC	Клас сутності	Всього	Опис
Service	A	3	Customer – Car, Order – Work, CarInspectionTask – CarInspectionFinding
	SA	5	CarInspection, WorkRecommendation, PartReservation, PartConsumption, Worker
	E	3	Car, Work, CarInspectionFinding
Inventory	A	1	Supply-SupplyItem
	SA	5	Supplier, StockLevel, Part, Location, InventoryTransaction
	E	1	SupplyItem

Модель операційної оцінки, модифікація базового методу, зокрема використання розширеної моделі розрахунку Unadjusted Use-Case Size Points (xUUSP) і класифікація прецедентів RTP для прецедентів використання класу *Modify* виступили базисом для перевірки наступних сценаріїв (S1-S4) та гіпотез (H0-H3) щодо оцінки міграції системи.

S1.Базове цілісне оцінювання. Розрахунок операційної складності виконувався на основі розробленої моделі с використанням методу (Lytvynov et al., 2026) без архітектурної класифікації прецедентів.

S2. Оцінювання з застосуванням базового методу з класифікацією прецедентів. Цей сценарій є розширенням S1 шляхом попереднього розділення прецедентів використання за класами доменних сутностей (E, SA, A).

S3. Цілісне оцінювання з застосуванням модифікації базового методу. Розрахунок виконувався без архітектурної класифікації, але з впровадженням розширеної моделі розрахунку Unadjusted Use-Case Size Points, що враховує додаткові вагові параметри структурних елементів прецедентів.

S4. Комплексне оцінювання з застосуванням модифікації базового методу та використанням класифікації. Даний сценарій поєднує переваги архітектурної класифікації сутностей (як у S2) та обчислювальні можливості розширеної моделі розрахунку Unadjusted Use-Case Size Points (як у S3).

Першою ціллю сценаріїв було визначити як класифікація впливає на точність прогнозування та чи дозволяє розширений метод обчислення xUUSP підвищити точність оцінювання у порівнянні з методом (Lytvynov et al., 2026). Другою було оцінити, наскільки історія прогнозів, що отримана під час міграції одного обмеженого контексту, може покращити прогнозування для іншого.

H0 (ефективність розширеної моделі розрахунку UUSP): Розширена модель (сценарії S3 та S4) має забезпечувати вищу деталізацію структурної складності, що призводить до покращення точності оцінювання порівняно з базовим методом (сценарії S1 та S2).

H1 (вплив класифікації): класифікація прецедентів використання за класами сутностей (сценарії S2 та S4) має підвищувати точність прогнозування порівняно з некласифікованим підходом (сценарії S1 та S3). Це пояснюється тим, що вона враховує різну операційну щільність архітектурних об’єктів.

H2 (вплив історії): Міжконтекстне перенесення знань має зменшувати варіативність прогнозування на ранніх етапах. Таким чином, це має покращити прогнозування (порівнюються результати сценаріїв S1-S4 з урахуванням і без урахування історії прогнозів).

H3 (ефект поєднання підходів): Поєднання класифікації сутностей та зважених параметрів розширеної моделі розрахунку Unadjusted Use-Case Size

Points (сценарій S4) має забезпечувати найвищу передбачуваність і найменше відносне відхилення (MMRE) серед усіх протестованих наборів даних міграції.

Відмінності від базового методу

На відміну від (Lytvynov et al., 2026), де оцінюються трудовитрати в людино-годинах для варіацій у межах анемічної моделі (ADM), у цій роботі вирішується операційно-орієнтована оцінка міграції від анемічної до збагаченої доменної моделі (RDM), що вимагає розрізнення операцій перенесення (M) і створення нової логіки (N). Безпосередньо з (Lytvynov et al., 2026) успадковано лише правила формалізації прецедентів, структуру фреймів-класів і механізм фазифікації FUSP, який було розширено. Самостійними результатами цієї роботи є: формальна модель операційної складності у вигляді кортежу OM = ⟨O, L, ET, W⟩ з проекціями за класами операцій, архітектурними шарами та типами сутностей, відсутня в (Lytvynov et al., 2026), де відповідний опис має описовий, неформалізований характер; розширена модель обчислення xUUSP із вагами структурних елементів кроків (кількість полів, простих, середніх і складних правил, залежностей), яка розширює, а не повторює підрахунок UUSP з (Lytvynov et al., 2026); автоматизоване калібрування вагових коефіцієнтів гібридним методом градієнтного спуску з моментом і backtracking відсутній в (Lytvynov et al., 2026); розширена експериментальна вибірка (89 прецедентів використання, 53 класу Modify у двох обмежених контекстах – Service та Inventory) проти 20 прецедентів, 5 класу Modify в одному обмеженому контексті у (Lytvynov et al., 2026).

Результати досліджень операційно-орієнтованого методу оцінювання трудомісткості архітектурної міграції в контексті предметно-орієнтованого проєктування

Оцінка застосування базового методу Fuzzy Use Case Size Point

Репрезентативну вибірку основного набору даних для сценарію S1, зосередженого на прецедентах використання класу Modify в обмеженому

контексті Service, та приклад прогнозування для операцій типу N а також повні дані розрахунків доступні для перегляду у відкритому репозиторії за посиланням (Lytvynov, 2026). Підсумкові результати експериментів S1 та S2 наведено в таблиці 6.

Таблиця 6

Підсумкові результати експериментів S1 та S2

Обмеження	N/M	Type	Mean	StdDev	MMRE	PRED	Sample
Service Context	N	U	44.06%	59.66%	0.6495	14.3%	21
	M		23.94%	33.48%	0.3413	42.9%	
Inventory Context	N		28.43%	41.10%	0.4364	13.3%	30
	M		13.52%	26.36%	0.2559	66.7%	
Service Context	N	A	20.3%	38.39%	0.373	17%	6
	M		27.8%	21.81%	0.320	33%	
Inventory Context	N		88.5%	52.82%	0.885	0.0%	5
	M		46.3%	31.14%	0.463	40.0%	
Service Context	N	SA	22.6%	36.15%	0.369	18%	11
	M		19.6%	26.15%	0.268	45%	
Inventory Context	N		21.8%	37.37%	0.346	26%	23
	M		11.1%	23.06%	0.210	78%	
Service Context	N	E	0%	0.00%	0	100.0%	2
	M		0%	40.45%	0.286	0%	

Без класифікації Entity модель демонструє значні відхилення, особливо для обмеженого контексту Service, де *MMRE* = 0.6495 та *PRED*(0.25) становить лише 14.3%. Це підтверджує, що універсальний підхід до оцінювання прецедентів використання не враховує структурні особливості системи. Перехід до осі M покращує MMRE до 0.3413 та 0.2559 в обох обмежених контекстах, однак показник надійності (*PRED*) усе ще нижчий за бажаний поріг 75%.

Коли дані декомпозуються за типами Entity (A, SA, E), діагностична здатність моделі UUSP зростає (для SA PRED знаходиться в межах від 18% до 78%), однак для класу A результати залишаються нестабільними. У підсумку можна стверджувати, що гіпотеза H1 частково підтверджується експериментами, але твердження про можливість точного прогнозування трудовитрат за допомогою класичного методу UUSP не підтвердилося.

Дослідження модифікації базового методу та автоматизованого нелінійного калібрування вагових параметрів

Наступну серію експериментів присвячено сценаріям S3 та S4.

Автоматичний підбір коефіцієнтів здійснювався наступним чином. Було використано стабільне значення $w_b = w_p = 0.1$ для базових операцій, тоді як інші коефіцієнти змінювалися в діапазонах $w_{s,r} = 2..4$, $w_d = w_{a,r} = 8..12$ (для методу GridSearch) з кроком 1.0, при цьому $w_{c,r}$ не задіяний у даній серії експериментів за відсутністю таких правил. Під час експериментів було виявлено, що точність моделі нелінійно залежить від різних компонент ваг – як для некласифікованих і класифікованих варіантів, так і для моделей операцій типів M та N.

Перехід від базового сценарію (S2) до каліброваного стану (S4) фактично відповідав дискретному градієнтному спуску (discrete gradient descent). Оптимальні трудовитрати міграції можна інтерпретувати як градієнт на поверхні архітектурних метрик (gradient on a surface of architectural metrics).

Експерименти підтвердили, що часткова похідна функції похибки за структурними залежностями ($\partial L/\partial w_{a,r}$) є більшою, ніж для простих правил ($\partial L/\partial w_{s,r}$) і значно більшою, ніж для базових функціональних кроків ($\partial L/\partial w_b$) для певних класів прецедентів.

Grid Search (з кроком 1.0) показав такі результати. Для класифікованих прецедентів використання типу *N* найкращими виявилися ваги $w_{s,r} = 2$, $w_d = w_{a,r} = 8$, для операцій M $w_{s,r} = 4$, $w_d = w_{a,r} = 8$. Для некласифікованих прецедентів використання – $w_{s,r} = 2$, $w_d = w_{a,r} = 12$ (табл. 7).

Таблиця 7

Підсумки результатів експериментів S3 та S4

Обмежений контекст	N/M	Type	Mean	StdDev	MMRE	PRED	Num
$w_{s,r} = 2$, $w_d = w_{a,r} = w_{c,r} = 12$ найкраще для Unclassified							
Service Context	N	U	2.08%	24.68%	0.1826	81.0%	21
	M		–5.60%	23.81%	0.1903	81.0%	
Inventory Context	N		1.08%	11.66%	0.0863	96.7%	30
	M		–6.45%	17.65%	0.1513	86.7%	
$w_{s,r} = 2$, $w_d = w_{a,r} = w_{c,r} = 8$ найкраще для N–операцій							
Service Context	N	A	–6.50%	16.88%	0.1236	83.3%	6

Inventory Context	N		16.21%	6.11%	0.1621	100.0%	5
Service Context	N		–2.48%	12.17%	0.0834	90.9%	11
Inventory Context	N	SA	9.01%	14.06%	0.1415	91.3%	23
Service Context	N	E	1.13%	7.43%	0.0525	100.0%	2
$w_{s,r} = 4$, $w_d = w_{a,r} = 8$ найкраще для M–операцій							
Service Context	M	A	–2.42%	11.00%	0.0726	100.0%	6
Inventory Context	M		10.65%	13.98%	0.1343	80.0%	5
Service Context	M	SA	–2.69%	15.86%	0.1337	81.8%	11
Inventory Context	M		7.07%	17.24%	0.1513	87.0%	23
Service Context	M	E	–0.37%	33.43%	0.2364	100.0%	2

Поряд із Grid Search для підвищення швидкості було реалізовано гібридний алгоритм оптимізації на основі модифікованого методу градієнтного спуску, описаного формулами (16)–(19). Результати наведені в табл. 8. Коефіцієнти змінювалися в діапазонах $w_{s,r} \in [0.5, 15.0]$, $w_d = w_{a,r} \in [1.0, 35.0]$ з кроком 1.0.

Таблиця 8

Підсумкові результати експериментів S3 та S4 із використанням методу гібридного градієнтного спуску з бектрекінгом (Hybrid Gradient Descent with Backtracking)

Обмежений контекст	N/M	Type	Mean	StdDev	MMRE	PRED	Num
$w_{s,r} = 2.48$, $w_d = w_{a,r} = w_{c,r} = 12$ найкраще для Unclassified							
Service Context	M	All	4.60%	26.95%	0.2196	76.2%	21
Service Context	M	All	-4.21%	22.42%	0.1696	81.0%	21
Inventory Context	M	All	3.11%	13.68%	0.1105	96.7%	30
Inventory Context	M	All	-5.09%	15.60%	0.1246	90.0%	30
$w_{s,r} = 1$, $w_d = w_{a,r} = 12$ для *N*–операцій							
Service Context	N	A	-12.08%	7.10%	0.1208	100.0%	6
Service Context	N	SA	-10.21%	11.63%	0.1441	100.0%	11
Service Context	N	E	5.80%	25.35%	0.1793	100.0%	2
Inventory Context	N	A	-19.88%	4.70%	0.1988	80.0%	5
Inventory Context	N	SA	2.23%	6.52%	0.0330	95.7%	23
$w_{s,r} = 2.78$, $w_d = w_{a,r} = w_{c,r} = 6.77$ для *M*–операцій							
Service Context	M	A	7.88%	14.73%	0.1258	83.3%	6
Service Context	M	SA	3.99%	17.67%	0.1435	81.8%	11
Service Context	M	E	0.32%	23.55%	0.1665	100.0%	2
Inventory Context	M	A	4.45%	12.16%	0.0982	100.0%	5
Inventory Context	M	SA	6.93%	14.33%	0.1344	87.0%	23

Аналіз отриманих експериментальних даних (табл. 7, 8) дозволяє стверджувати, що модифікація методу градієнтного спуску не надає таких

результатів, як Grid Search. Проте вона є більш ефективною, оскільки дозволяє замінити тривалий та ресурсномісткий повний перебір параметрів адаптивним аналітичним пошуком в означених рамках точності прогнозування (PRED(0.25)). Це робить метод придатним для інтеграції в автоматизовані системи управління програмними проєктами.

Висновки і перспективи подальших досліджень

Розроблено формальну модель оцінки операційної складності міграції бізнес-логіки, яка включає класифікацію операцій, з урахуванням їх розподілу між двома вимірами-категоріями: M-перенесенням вже існуючої логіки, N – побудовою нової. Також за даною моделлю було запропоновано фіксувати шар бізнес-логіки, якого торкається модифікація та ввести класифікацію прецедентів за пов'язаними з ними типами сутностей, виділивши три базові класи.

Дана модель стала основою методу кількісної оцінки трудовитрат розподілених по категоріям (M/N), класам сутностей та системним шарам.

На базі експериментів було підтверджено, що базовий метод з використанням розробленої моделі операційної складності не може застосовуватися для прогнозування операційної складності міграції. Без класифікації модель демонструє значні відхилення, особливо для обмеженого контексту Service, де *MMRE* = 0.6495 та *PRED* (0.25) становить лише 14.3%. Коли прецеденти було декомпозовано за типами сутностей, діагностична здатність моделі UUSP зросла (напр., для Singular Aggregates PRED знаходиться в межах від 18% до 78%), однак для класу *A* результати залишилися нестабільними.

Розроблено метод для операційного оцінювання міграції варіацій в рамках підходу DDD та методи автоматизованого калібрування вагових коефіцієнтів моделі. Даний метод відрізняється від базового новою моделлю оцінки кроку прецеденту, класифікацією рівнів складності для фазифікації оцінки; автоматизацією визначення вагових коефіцієнтів на базі модифікації методу градієнтного спуску із моментом та евристичного пошуку із бектрекінгом.

На базі проведених експериментів можна стверджувати, що на відміну від застосування базового методу розроблений метод операційного оцінювання значно покращив оцінку для розглянутого типу прецедентів. Завдяки введенню каліброваних ваг окремо для класів операцій *N* та *M*, модель успішно усунула розрив між функціональними вимогами та технічною реальністю. Зокрема, вимога PRED(0.25) виконується для усіх контекстів (мінімальне значення 80% для GridSearch та 76.2% для методу на базі градієнтного спуску, тоді як для базового методу воно складає 0%). Серед переваг запропонованого підходу можна відмітити стабілізацію стандартного відхилення помилки прогнозу з 59.66% (S1, без калібрування) до діапазону 4.70–26.95% (S4, після калібрування).

Стосовно обмежень цього дослідження слід зазначити наступне. Дослідження було спрямоване на еволюційну міграцію в рамках архітектури DDD з відносно простими прецедентами, що можна обґрунтувати типовою структурою даного класу проєктів. Це надало змогу класифікувати прецеденти і вплинуло на отримані результати. Цей метод не може бути безпосередньо застосований за межами варіацій підходу DDD, наприклад, у контексті міграцій старих систем або монолітів, розроблених без урахування принципів DDD на архітектуру DDD.

Розвиток дослідження полягає в поступовому знятті вказаних обмежень: розгляді інших класів проєктів, врахуванні специфіки міграції старих систем на базу DDD, міграцій монолітних проєктів, розроблених на базі DDD на мікросервісну архітектуру.

Література

1. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software. Boston : Addison-Wesley Professional, 2004. 560 p. ISBN 978-0-321-12521-7.

2. Vernon V. Implementing Domain-Driven Design. Boston : Addison-Wesley, 2013. 656 p. ISBN 978-0-321-83457-7.

3. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston : Prentice Hall, 2017. 432 p. ISBN 978-0-13-449416-6.

4. Ford N., Parsons R., Kua P., Sadalage P. Building Evolutionary Architectures. 2nd ed. Sebastopol : O'Reilly Media, 2022. 438 p. ISBN 978-1-4920-9754-9.

5. Kamimura M., Matsuo A., Maeda Y. Measuring Business Logic Complexity in Software Systems. *2015 Asia-Pacific Software Engineering Conference (APSEC)*. 2015. P. 370–376. DOI: https://doi.org/10.1109/APSEC.2015.26.

6. Maeda Y., Kamimura M., Yano K. An Approach to Transforming Systems for Execution on Cloud Computing Systems. *FUJITSU Scientific & Technical Journal*. 2017. Vol. 53, no. 5. P. 71–77. URL: https://dl.ndl.go.jp/pid/11613544.

7. Lytvynov O. A., Khandetskyi V. S., Lytvynov M. O. Estimation of effort of migration among domain-driven design architectural variations. *Radio Electronics, Computer Science, Control*. 2026. No. 1. P. 159–175. DOI: https://doi.org/10.15588/1607-3274-2026-1-14.

8. Khononov V. Learning Domain-Driven Design. Sebastopol : O'Reilly Media, 2021. 340 p. ISBN 978-1-098-10013-1.

9. Millett S., Tune N. Patterns, Principles, and Practices of Domain-Driven Design. Indianapolis : Wrox, 2015. 800 p. ISBN 978-1-118-71470-6.

10. Bajusova D., Silhavy P., Silhavy R. Enhancing Software Effort Estimation With Self-Organizing Migration Algorithm: A Comparative Analysis of COCOMO Models. *IEEE Access*. 2024. Vol. 12. P. 67170–67188. DOI: https://doi.org/10.1109/ACCESS.2024.3399060.

11. Hjalmarsson A., Korman M., Lagerström R. Software Migration Project Cost Estimation using COCOMO II and Enterprise Architecture Modeling. *Proceedings of the Practice of Enterprise Modeling (PoEM 2013) Short Papers*. 2013. P. 39–48. URL: https://ceur-ws.org/Vol-1023/paper4.pdf.

12. Silhavy R., Silhavy P., Prokopova Z. Using Actors and Use Cases for Software Size Estimation. *Electronics*. 2021. Vol. 10, no. 5. P. 592. DOI: https://doi.org/10.3390/electronics10050592.

13. Azzeh M., Nassif A. B., Martín C. L. Empirical analysis on productivity prediction and locality for use case points method. *Software Quality Journal*. 2021. Vol. 29. P. 309–336. DOI: https://doi.org/10.1007/s11219-021-09547-0.

14. Haryono A. F., Farhan A., Khairani D., Razak S., Mintarsih F. Use Case Point Activity-Based Costing and Adjusted Function Point for Software Cost Estimation. *Jurnal Teknik Informatika*. 2025. Vol. 18, no. 2. P. 316–326. DOI: https://doi.org/10.15408/jti.v18i2.46669.

15. Nhung H. L. T. K., Hoc H. T., Van Hai V. A Review of Use Case-Based Development Effort Estimation Methods in the System Development Context. *Software Engineering and Computer Systems*. Cham : Springer, 2019. P. 484–499. DOI: https://doi.org/10.1007/978-3-030-30329-7_44.

16. Lytvynov M. Working Materials for the Article "Operation-oriented model and method for effort estimation of architectural migration among Domain-Driven Design variations" [Dataset]. Zenodo. 2026. DOI: https://doi.org/10.5281/zenodo.21383199.

17. Dolado J. J. On the Problem of the Software Cost Function. *Information and Software Technology*. 2001. Vol. 43, no. 1. P. 61–72. DOI: https://doi.org/10.1016/S0950-5849(00)00137-3.

References

1. Evans, E. (2004). *Domain-driven design: Tackling complexity in the heart of software*. Addison-Wesley Professional.

2. Vernon, V. (2013). *Implementing domain-driven design*. Addison-Wesley.

3. Martin, R. C. (2017). *Clean architecture: A craftsman's guide to software structure and design*. Prentice Hall.

4. Ford, N., Parsons, R., Kua, P., & Sadalage, P. (2022). *Building evolutionary architectures* (2nd ed.). O'Reilly Media.

5. Kamimura, M., Matsuo, A., & Maeda, Y. (2015). Measuring business logic complexity in software systems. In *2015 Asia-Pacific Software Engineering Conference (APSEC)* (pp. 370–376). IEEE. https://doi.org/10.1109/APSEC.2015.26

6. Maeda, Y., Kamimura, M., & Yano, K. (2017). An approach to transforming systems for execution on cloud computing systems. *FUJITSU Scientific & Technical Journal*, *53*(5), 71–77. https://dl.ndl.go.jp/pid/11613544

7. Lytvynov, O. A., Khandetskyi, V. S., & Lytvynov, M. O. (2026). Estimation of effort of migration among domain-driven design architectural variations. *Radio Electronics, Computer Science, Control*, (1), 159–175. https://doi.org/10.15588/1607-3274-2026-1-14

8. Khononov, V. (2021). *Learning domain-driven design*. O'Reilly Media.

9. Millett, S., & Tune, N. (2015). *Patterns, principles, and practices of domain-driven design*. Wrox.

10. Bajusova, D., Silhavy, P., & Silhavy, R. (2024). Enhancing software effort estimation with self-organizing migration algorithm: A comparative analysis of COCOMO models. *IEEE Access*, *12*, 67170–67188. https://doi.org/10.1109/ACCESS.2024.3399060

11. Hjalmarsson, A., Korman, M., & Lagerström, R. (2013). Software migration project cost estimation using COCOMO ii and enterprise architecture modeling. In *Proceedings of the Practice of Enterprise Modeling (PoEM 2013) Short Papers* (pp. 39–48). CEUR-WS. https://ceur-ws.org/Vol-1023/paper4.pdf

12. Silhavy, R., Silhavy, P., & Prokopova, Z. (2021). Using actors and use cases for software size estimation. *Electronics*, *10*(5), 592. https://doi.org/10.3390/electronics10050592

13. Azzeh, M., Nassif, A. B., & Martín, C. L. (2021). Empirical analysis on productivity prediction and locality for use case points method. *Software Quality Journal*, *29*, 309–336. https://doi.org/10.1007/s11219-021-09547-0

14. Haryono, A. F., Farhan, A., Khairani, D., Razak, S., & Mintarsih, F. (2025). Use case point activity-based costing and adjusted function point for software cost estimation. *Jurnal Teknik Informatika*, *18*(2), 316–326. https://doi.org/10.15408/jti.v18i2.46669

15. Nhung, H. L. T. K., Hoc, H. T., & Van Hai, V. (2019). A review of use case-based development effort estimation methods in the system development context. In *Software Engineering and Computer Systems* (pp. 484–499). Springer. https://doi.org/10.1007/978-3-030-30329-7_44

16. Lytvynov, M. (2026). Working Materials for the Article "Operation-oriented model and method for effort estimation of architectural migration among Domain-Driven Design variations" [Dataset]. Zenodo. https://doi.org/10.5281/zenodo.21383199

17. Dolado, J. J. (2001). On the problem of the software cost function. *Information and Software Technology*, *43*(1), 61–72. https://doi.org/10.1016/S0950-5849(00)00137-3