УДК 004.04:004.09
DOI https://doi.org/10.15589/znp2024.1(494).15

A REVIEW OF SOFTWARE DEVELOPMENT EFFORT ESTIMATION METHODS

АНАЛІЗ СУЧАСНОГО СТАНУ МЕТОДІВ ОЦІНЮВАННЯ ТРУДОМІСТКОСТІ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Tetyana A. Farionova
tetyana.farionova@nuos.edu.ua
ORCID: 0000-0003-3384-4712
Oleksandr S. Oriekhov
oleksandr.oriekhov@nuos.edu.ua
ORCID: 0000-0002-0001-0140

Т. А. Фаріонова,
канд. техн. наук, доцент
О. С. Орєхов,
аспірант

Admiral Makarov National University of Shipbuilding, Mykolaiv
Національний університет кораблебудування імені адмірала Макарова, м. Миколаїв

Abstract. Software development effort estimation is an important indicator for budget forming and planning the development time of a software project. The choice of correct methods and models for software development effort estimation, depending on the development methodology, allows you to obtain reliable information about the planning of the software project.
The purpose of the work is the analysis of modern methods of software development effort estimation, determining their advantages and disadvantages, for further use in the creation of IT projects using certain frameworks, technologies, taking into account the purpose of software and development languages.
The object of the study is the process of estimating the complexity of software development.
The subject of the study are methods, approaches and models of software development effort estimation.
The work considers algorithmic and non-algorithmic methods for software development effort estimation. Evaluation by non-algorithmic methods, which are based on unstructured expert judgments, is unreliable. Algorithmic methods use parametric estimation and are built on the basement of math. Methods based on artificial intelligence and machine learning models use already existing information on projects and are built on the basis of certain data sets. The development of nonlinear regression models based on normalizing transformations makes it possible to improve the reliability of software development effort and duration estimation, taking into account the probabilistic nature of the distribution of empirical data.
In conclusion the existing approaches require improvement of modern methods of estimating the software development effort estimation in order to increase the accuracy of the corresponding estimate. It was determined that one of the main factors that affects the success of software development projects and allows such projects to be completed on time is the reliability of software development effort and duration estimation of these projects, which depend on models and evaluation methods. Prospects for further research are aimed at improving existing and developing new software development effort estimation models taking into account the features of Agile methodology, software categories, frameworks and programming languages.
Key words: software development effort estimation; estimation models; software code metrics; software; Agile; regression model.

Анотація. Оцінка трудомісткості розробки програмного забезпечення є важливим показником для формування бюджету та планування часу розробки програмного проєкту. Вибір коректних методів та моделей оцінки трудомісткості розробки ПЗ в залежності від методології розробки, дозволяє отримати достовірну інформацію щодо планування програмного проєкту.
Метою роботи є аналіз сучасних методів оцінювання трудомісткості розробки програмного забезпечення (ПЗ), визначення їх переваг та недоліків, для подальшого використання при створенні ІТ-проєктів із застосуванням певних фреймворків, технологій, враховуючи призначення ПЗ та мови розробки.
Об'єктом дослідження є процес оцінювання трудомісткості розробки ПЗ. *Предметом дослідження* є методи, підходи та моделі оцінювання трудомісткості розробки ПЗ.
В роботі розглядаються алгоритмічні та неалгоритмічні методи для оцінки трудомісткості розробки ПЗ. Оцінка неалгоритмічними методами, які базуються на неструктурованих експертних судженнях, є ненадійною. Алго-

ритмічні методи використовують параметричну оцінку та побудовані на основі математичного апарату. Методи на основі штучного інтелекту та моделей машинного навчання, використовують вже існуючу інформацію по проєктам та побудовані на основі певних наборів даних. Розробка нелінійних регресійних моделей на основі нормалізуючих перетворень дає змогу покращити достовірність оцінювання трудомісткості та тривалості розробки проєктів ПЗ, враховуючи ймовірнісний характер розподілу емпіричних даних.

У *висновку* зазначено, що існуючі підходи потребують удосконалення сучасних методів оцінки трудомісткості розробки ПЗ для підвищення точності відповідної оцінки. Визначено, що одним з основних факторів, який впливає на успішність виконання проєктів з розробки ПЗ та дозволяє вчасно завершувати такі проєкти, є достовірність оцінювання трудомісткості та тривалості цих проєктів, які залежать від моделей та методів оцінювання. Перспективи подальших досліджень спрямовані на вдосконалення існуючих та розробку нових моделей оцінювання розробки ПЗ з урахуванням особливостей Agile методології, категорій ПЗ, фреймворків і мов програмування.

Ключові слова: оцінка трудомісткості програмного забезпечення; моделі оцінювання; метрика програмного коду; програмне забезпечення; Agile методологія; регресійна модель.

ПОСТАНОВКА ЗАДАЧІ

Оцінка трудомісткості розробки програмного забезпечення є одним з ключових показників для бізнеса при формуванні бюджету та плануванні часу розробки. Ця задача є важливою для IT-компаній, оскільки дозволяє спрогнозувати кількісні показники ресурсів, які необхідно виділити на створення програмного проєкту, що в свою чергу допоможе враховувати ризики і ресурси, та дозволить підвищити ефективність процесу розробки. Підбір коректних методів та моделей оцінки трудомісткості розробки ПЗ в залежності від методології розробки, дозволить отримати достовірну інформацію щодо планування програмного проєкту.

Критерії «хорошої» оцінки ґрунтуються на здатності її підтримувати успіх проєкту. **Хороша оцінка** (за Стівом МакКоннеллом [1, с. 9]) – це оцінка, яка забезпечує достатньо чітке уявлення про реальність проєкту, щоб дозволити керівництву прийняти правильні рішення про те, як контролювати проєкт, щоб досягти поставлених цілей. Заниження або завищення оцінки трудомісткості може призвести до певних проблем.

Достовірна оцінка є джерелом інформації для формування реальних витрат по ресурсам проєкту, що дозволяє менеджерам проєкту прийняти правильні рішення та досягти поставлених цілей. Заниження (Underestimation) або завищення (Overestimation) оцінки може призвести до суттєвих проблем. Як показано на рис. 1, найкращі результати розробки ПЗ, досягаються завдяки обґрунтованим оцінкам, що можна отримати на ранніх стадіях проєктування. Якщо оцінка занадто низька, неефективність планування призведе до збільшення фактичної вартості та збільшення термінів розробки проєкту. У випадку, коли оцінка зависока, спрацьовує закон Паркінсона [1, с. 21–23].

Статистика успішності програмних проєктів, яка представлена компанією The Standish Group за період 25 років за показниками часу розробки (OnTime), дотриманням бюджету (OnBudget), повнотою реалізації функціоналу ПЗ (OnTarget), має помірну

Рис. 1. Порівняння вартості зусиль в залежності від недооцінки, точної оцінки та переоцінки [1]

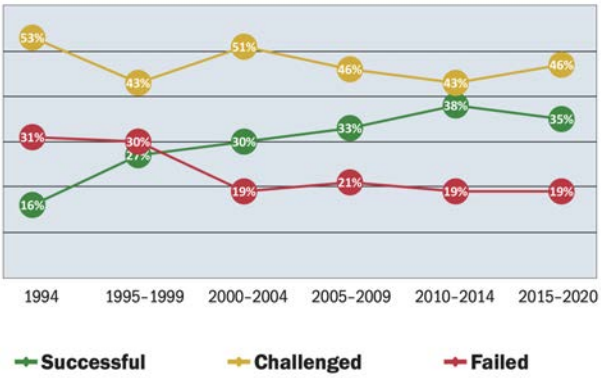

Рис. 2. Дослідження успішності реалізації IT проєктів за даними The Standish Group за період з 1994 року по 2020 рік [2], де Successful – успішні проєкти, Challenged – проєкти з певними проблемами при реалізації, Failed – проєкти, які провалились

позитивну динаміку в бік збільшення частки успішно реалізованих проєктів. Так, в 1994 частка успішно реалізованих проєктів складала 16%, а в 2020 році відсоток таких проєктів (виконані вчасно, в межах бюджету, реалізовано 100% вимог) вже становив 35%. Статистика провальних проєктів зменшилась з 31% до 19% за період з 1994 по 2020 рік. А частка проєктів, які зазнали проблеми (не вклалися в терміни, або вийшли за рамки бюджету, або функціонал не був повністю реалізований у відповідності до вимог), має незначну тенденцію до зменшення, біля 4% (див. рис. 2) [2].

Крім того дослідження The Standish group [3] показали, що чим більший і складніший проєкт, тим вище імовірність помилкової оцінки трудомісткості розробки програмного проєкту (рис. 3).

	SUCCESSFUL	CHALLENGED	FAILED	TOTAL
Grand	6%	51%	43%	100%
Large	11%	59%	30%	100%
Medium	12%	62%	26%	100%
Moderate	24%	64%	12%	100%
Small	61%	32%	7%	100%

Рис. 3. Статистика успішності розробки ПЗ в залежності від його розміру за звітом The Standish Group [3]

Аналіз успішності реалізації IoT (Internet-of-Things) проєктів, проведений у 2017 році відомою та авторитетною компанією CISCO, показав, що близько 60% таких проєктів зазнають труднощів на етапі розробки концепту, і тільки 26% – завершуються вчасно, вкладаються в бюджет та мають заявлений функціонал [4].

На реалізацію програмного проєкту суттєво впливає людський фактор: організаційна структура команди, корпоративна культура, рівень soft skills, відповідальність членів команди за прийняття рішень, ізольованість IT-архітектури тощо [5, с. 46–48]. В роботі [5] зазначено, що більшість IT проєктів може зазнати провалу через некваліфіковане виконання персоналом поставлених задач.

В індустрії IT з 1994 року відбулась зміна цілих епох щодо підходів та технологій розробки, збільшився розмір програмних продуктів. Відбулися зміни у побудові архітектури клієнтських і серверних застосунків. Активного поширення набули здорові практики написання коду, такі як CLEAN, DRY (don't repeat yourself), TDD (test driven development), DDD (data drive development), тощо.

Таким чином, якісне планування ресурсів і бюджету є надзвичайно важливим для успішної реалізації будь-якого проєкту, для будь-якої компанії з розробки ПЗ, оскільки воно дає уявлення про те чи здатна взагалі компанія взятися за проєкт і довести його до успішного запуску в межах очікуваного часу та бюджету. Для цього необхідно використовувати відповідні підходи, методи та моделі оцінювання часу, вартості та трудомісткості розробки ПЗ.

АНАЛІЗ ОСТАННІХ ДОСЛІДЖЕНЬ І ПУБЛІКАЦІЙ

Оцінюванню трудомісткості та вартості розробки ПЗ із застосуванням алгоритмічних і неалгоритмічних методів присвячені роботи вітчизняних і закордонних дослідників. В роботах [6; 7; 8; 9; 10; 11; 12; 13; 14; 15] наводиться огляд та аналіз найбільш ефективних методів оцінювання та порівняння їх між собою. В наукових працях [16; 17; 18] розглядаються підходи до побудови нелінійних регресійних моделей оцінювання трудомісткості розробки ПЗ для JAVA-застосунків на основі метрик програмного коду; роботи [19; 20; 21] присвячені оцінки трудомісткості розробки PHP-застосунків із використання алгоритмічних методів. В дослідженні [22] запропоновано математичні моделі оцінки відповідних метрик програмного коду для мобільних застосунків незалежно від цільової платформи, в роботі [23] розглядаються побудова регресійної моделей для оцінки трудомісткості розробки систем на базі Visual Basic. В [24] представлено аналіз використання метрик ПЗ, за даними ISBSG, для оцінки трудомісткості розробки ПЗ. В дослідженні [25] наведено огляд особливостей застосування моделі оцінки вартості програмного забезпечення SLIM на основі метрик програмного коду. В роботі [26] запропоновано моделі для оцінки трудомісткості Agile Web проєктів.

ВІДОКРЕМЛЕННЯ НЕВИРІШЕНИХ РАНІШЕ ЧАСТИН ЗАГАЛЬНОЇ ПРОБЛЕМИ

Отже, не дивлячись на стрімкий розвиток галузі інформаційних технологій, існують проблеми із достовірністю оцінювання трудомісткості та вартості розробки ПЗ. Аналіз останніх досліджень і публікацій показав, що більшість методів та моделей не враховують методології проєктування та технологічні аспекти розробки програмного забезпечення. Для отримання достовірних оцінок трудомісткості та вартості розробки ПЗ необхідно враховувати сучасні технологічні аспекти як розробки, так і управління програмними проєктами. Тому аналіз існуючих методів оцінки трудомісткості ПЗ є актуальною науково-практичною задачею.

МЕТА ДОСЛІДЖЕННЯ

Метою роботи є аналіз сучасних методів оцінювання трудомісткості розробки ПЗ, визначення їх переваг та недоліків, для подальшого використання при створенні IT-проєктів із застосуванням певних фреймворків, технологій, враховуючи призначення ПЗ та мови розробки.

МЕТОДИ, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Методи. Дослідження проводиться з використанням загальнонаукових методів дослідження, таких, як пошук, системний аналіз, синтез, порівняння, абдукція, ідеалізація, історичний і логічний методи.

Об'єктом дослідження є процес оцінювання трудомісткості розробки програмного забезпечення.

Предметом дослідження є методи, підходи та моделі оцінювання трудомісткості розробки ПЗ.

ОСНОВНИЙ МАТЕРІАЛ

Сучасні методи оцінювання трудомісткості розробки ПЗ можна умовно поділити на три категорії: **алгоритмічні, неалгоритмічні** та **моделі машинного навчання**. Оцінювання неалгоритмічними методами ґрунтується на думці експертів, які мають досвід подібних попередніх проєктів, тоді як алгоритмічні підходи використовують параметричну оцінку та побудовані на основі математичного апарату. Методи на основі штучного інтелекту (ШІ) та моделей машинного навчання використовують вже існуючу інформацію по проєктам та побудовані на основі певних наборів даних [6; 7].

Слід зазначити, що точність методів оцінювання трудомісткості розробки ПЗ залежить від використання певних методологій розробки.

Неалгоритмічні методи оцінювання.

Більшість неалгоритмічних методів оцінювання ПЗ базуються на експертних оцінках та аналітичному порівнянні з попередніми аналогічними проєктами. Найбільш популярними є методи експертного судження, аналогій, «top-down», «bottom-up», «price-to-win», широкосмуговий метод Delphi; planning poker, T-shirt size, що застосовуються для Agile підходу в розробці ПЗ [6; 7; 8].

Експертне оцінювання. Техніка експертного оцінювання є найбільш поширеним методом оцінювання трудомісткості розробки ПЗ. Цей підхід базується на досвіді керівника проєкту та експертів. Експертний метод корисний в тих ситуаціях, коли є обмеження в пошуку даних і зборі вимог. Судження експертів схильне до людських помилок і упередженості. Його успіх ґрунтується на експертній оцінці, але експертний досвід може відрізнятися від одного експерта до іншого, що призводить до різних оцінок для того самого типу проєкту. Однак такий підхід може бути корисним для оцінювання малих і середніх програмних проєктів, коли групи розробників і технологічний стек розробки ПЗ не зазнали значних змін порівняно з попередніми проєктами [6; 8].

Метод оцінки аналогій полягає у використанні досвіду з реалізації схожих програмних проєктів з метою прогнозування вартості нового проєкту. Оцінка вартості проєкту може бути зроблена на загальному рівні, або на рівні підсистем. Однак, при застосуванні цього підходу, не враховуються параметри попередніх проєктів, такі як середовище та функції, що можуть відрізнятися від відповідних параметрів нового проєкту. Крім того, потрібна інформація про вже реалізовані проєкти, тоді як у деяких ситуаціях вона може бути відсутня [6; 7; 9].

Так певні дослідження показуються, що на точність експертної оцінки в межах оцінювання проєктів середнього розміру не слід покладатися навіть в рамках однієї компанії. Оцінка витрат на основі неструктурованих експертних суджень є ненадійною. Низька достовірності оцінки має прямі економічні наслідки. З одного боку, компанія може втрачати бізнес можливості, якщо оцінки зависокі, а з іншого – гроші, якщо оцінки занизькі. Непослідовність суджень і надмірна самовпевненість експертів може стати проблемою. Це не означає, що експертне судження не слід використовувати як метод оцінки, а скоріше вказує на необхідність вдосконалення цього підходу [10].

Price-to-win. Метод оцінювання price-to-win базується на кількісних показниках бюджету клієнта, а не на параметрах або функціях програмного забезпечення. Приклад: клієнт готовий платити за 6 осіб на місяць, а оцінка проєкту становить 8 осіб на місяць, тоді оцінка виконується відповідно до платоспроможності клієнта. Цей метод допомагає отримати контракт, але зазвичай спричиняє перевитрати коштів і часу [6].

Bottom-up. Метод «bottom-up» окремо оцінює кожен компонент програмного забезпечення, а загальна оцінка для продукту є відповідною сумою. Це можливо лише тоді, коли вимоги та дизайн системи відомі на ранній стадії розробки програмного забезпечення [6; 7]. Метод забезпечує задовільну точність, але має і свої недоліки.

Top-down метод дозволяє оцінити загальну оцінку витрат для проєкту. Система підрозділяється на функціональні компоненти, які потім оцінюються на основі загальної оцінки як окремі складові. Для структуризації програмного забезпечення окремі компоненти, конструкція та вимоги до ПЗ мають бути чітко визначені. Метод, порівняно з «bottom-up», є менш детальним та не потребує багато часу і ресурсів. Однак, недоліком Top-down є те, що він може призвести до недооцінки витрат на окремі етапи розробки, оскільки вартість розраховується згори донизу, без детального аналізу кожного етапу, крім цього не враховуються проблеми низького рівня [6;7].

Широкосмуговий метод Delphi – це техніка оцінювання витрат, у якій параметри трудомісткості та витрати узгоджуються на основі консенсусу команди. Це робиться шляхом отримання порад від експертів, які мають досвід роботи в подібних проєктах. Широкосмугову техніку Delphi представили Баррі Боем і Джон Фаркуер у 1970-х роках. Метод використовує структуру розподілу робіт, як основу для оцінювання розміру проєкту, трудомісткості і вартості. Цей метод робить акцент на консультаціях, спілкуванні та взаємодії між учасниками (представниками клієнтів і членами технічної групи, які будуть залучені до розробки програмного продукту). Кожен з них оцінює окремі завдання, визначає зміни та припущення в структурі розподілу роботи. Експертів із високими та/або низькими оцінками просять обґрунтувати їх, а потім, за необхідності, ці оцінки переглядаються. Цикл повторюється, доки оцінювачі не дійдуть згоди щодо оцінок. Координатор збирає результати від

членів команди та формує список завдань. Достовірність широкосмугового методу Delphi залежить від досвіду членів команди та її злагодженості. Крім того, цей метод має достатню достовірність, коли вимоги чітко визначені. Він простий у застосуванні та підтримує оцінки на основі консенсусу, усі припущенні проходять через етапи обговорення та документуються. Вважається, що оцінка отримана за широкосмуговим методом Delphi є більш надійною та достовірною у порівнянні з індивідуальною оцінкою. Але незважаючи на те, що оцінки базуються на консенсусі, експерти можуть бути упередженими, оптимістичними чи песимістичними у своїх оцінках. Крім того цей метод часто потребує тісної співпраці з менеджментом, що не завжди можливо [6; 9]. Зазначений метод може застосовуватись як для оцінки всього проєкту, так й для окремих підсистем та функцій. Це надає можливість його використовувати для оцінювання проєктів ПЗ, створених, як за класичними методологіями, так і за методологіями Agile.

Poker Planning є методом оцінювання на основі експертних суджень, який використовується при Agile розробці ПЗ. Історії користувача визначаються product owner-ом (представником клієнта, відповідальним за підтримку та визначення пріоритетів історій користувача). Оцінка розробки виконується командою (групою розробників, відповідальною за впровадження історій користувачів), шляхом незалежного голосування в умовних Story Points. Вважається, що такий підхід надає кращі результати, ніж окремі експерти, забезпечуючи рівну участь усіх розробників у процесі оцінювання. Цей підхід далекий від точного оцінювання розробки в цілому програмного проєкту, він більше виражає відносні зусилля, для розробки окремих функціональних частин [6; 10; 11]. Для оцінки загальної трудомісткості розробки ПЗ необхідно з Planning Poker методом застосовувати ще додаткові емпіричні моделі оцінки трудомісткості ПЗ, орієнтовані на Agile методологію [11].

Оцінка на основі нечіткої логіки (Fuzzy Logic) або техніка м'яких обчислень. Метод нечіткої логіки може бути ефективним для оцінювання трудомісткості розробки ПЗ, за умов недостатньої інформації щодо програмного проєкту, при використанні гнучкої методології. При застосуванні цього підходу використовуються параметри швидкості розробки, розміру команди, відносної оцінки проєкту в Story Point. Перевагою методу є те, що він підходить для оцінювання проєктів, які створюються за гнучкою методологією та дозволяє враховувати нелінійні залежності між різними параметрами. До недоліків слід віднести складність у використанні та суб'єктивність результатів оцінки, яка залежить від суджень експертів [28].

Методи оцінки трудомісткості розробки ПЗ за допомогою ШІ та моделей машинного навчання.

Машинне навчання набуло популярності в моделях оцінки трудомісткості або витрат розробки ПЗ. Ці моделі можуть бути автономними моделями або моделями, які працюють у поєднанні з алгоритмічними моделями.

Метод оцінювання на основі нейронної мережі базується на навчанні мережі на основі історичних даних. Такий підхід може давати хороші результати оцінки трудомісткості чи вартості цілого проєкту, а значення параметрів налаштовані таким чином, щоб зменшити різницю між фактичними та прогнозованими оцінками. Нейронна мережа корисна тим, що її достатньо легко навчити, оскільки вона змінює ваги кожного разу, коли їх використовують. Це призводить до покрокового покращення оцінки від початкової стадії проєкту до його завершення. До недоліків слід віднести необхідність збору великої кількості навчальних даних для отримання якісного прогнозу [7; 8].

Існують різновиди моделей машинного навчання, які використовуються для оцінки трудомісткості розробки ПЗ на основі нейронних мереж: Multilayer Perceptron (MLP), General Regression Neural Network (GRNN), Radial Basis Function Neural Network (RBFNN) та Cascade Correlation Neural Network (CCNN). Основним обмеженням впровадження цих моделей є те, що немає консенсусу щодо того, яка модель є найкращою. Окремі моделі показують якісні оцінки для певних категорій проєктів, а для інших ні [13].

Дослідження [13] доводить, що моделі побудовані на основі нейронних мереж не завжди мають задовільну якість оцінювання. Модель CCNN надає кращі результати оцінки трудомісткості, але суттєво не відрізняється від інших моделей на основі нейронних мереж. Більшість моделей є чутливими до викидів вихідних даних та потребують дослідження, щодо нормальності розподілу метрик програмного забезпечення.

Алгоритмічні (або параметричні) методи оцінювання базуються на застосуванні математичних моделей та алгоритмів для визначення параметрів програмного проєкту таких, як трудомісткості, час та вартості розробки. Ці методи можуть бути використані як на етапі планування проєкту, так і на етапі розробки ПЗ. Для оцінювання трудомісткості розробки програмного забезпечення найбільш поширеними є моделі COCOMO та COCOMO II, Function Point Analysis (FPA), SLIM, ISBSG та інші. Для гнучких методологій розробки ПЗ може застосовуватись конструктивний Agile алгоритм оцінювання – AgileMOW.

Constructive Cost Model (COCOMO) – це одна з найпопулярніших алгоритмічних моделей для оцінювання трудомісткості розробки ПЗ, що була запропонована Баррі Боемом [8]. Параметри та рівняння, які використовуються в моделі, отримані на основі статистистичних даних за метриками (розмір коду *KLOC* (тисячі рядків коду), трудомісткість, *Effort* (людино-місяцях) вже реалізованих програмних проєктів. Згідно [8], існує три моделі COCOMO:

– Basic COCOMO – перша з набору моделей COCOMO.

$Effort=a*(KLOC)^b$,

де значення констант *a* і *b* залежить від типу проєкту. Модель цього рівня підходить для ранньої швидкої приблизної оцінки витрат, але точність її дуже низька;

– Intermediate COCOMO. На цьому рівні базова модель уточнена за рахунок введення додаткових 15-ти «Атрибутів вартості» (або факторів витрат) *EAF*, які згруповані за чотирма категоріями (Product Attributes, Hardware Attributes, Personnel Attributes, Project Attributes).

$Effort=a*(KLOC)^b*EAF$,

де значення констант *a* і *b* залежить від типу проєкту, але відрізняються від Basic COCOMO;

– Detailed COCOMO – крім параметрів Intermediate COCOMO, враховує повний набір факторів розробки програмного забезпечення, таких як складність проєкту, надійність ПЗ, розмір бази даних, середовище розробки, тощо.

$Effort = a\times(KLOC)^b\times(SF)^i\times(EM)^j\times(PM)^k\times(M)^l$,

де *SF* – коефіцієнт масштабу, *EM* – множник зусиль, *PM* – параметр персоналу, *M, a, b ,i, j, k, l* – параметри, які залежать від типу проєкту та вимог. Detailed COCOMO підходить для складних проєктів із детальними вимогами та значною кількістю змінних, що впливають на оцінку.

Перевагою моделей COCOMO є те, що оцінювання може бути виконано на ранніх стадіях проєктування програмного забезпечення. До недоліків слід віднести, що моделі були розроблені за даними проєктів, які є застарілими, тому для сучасних проєктів, що використовують нові технології розробки, оцінки за цією моделлю можуть бути не якісними [8].

У 1997 методика була вдосконалена і отримала назву COCOMO II [6]. При побудові COCOMO II для обробки статистичних даних використовується Байєсовський аналіз, який дає кращі результати для програмних проєктів, що мають певні фактори невизначеності. Також у COCOMO II допускається вимірювати розмір проєкту не тільки за допомогою метрики рядків коду, а й із використанням параметру функціональних точок. При розрахунку показників COCOMO II враховується рівень зрілості процесу розробки відповідно до моделей SEI CMM/CMMI.

До недоліків цієї моделі слід віднести неякісну оцінку тривалості розробки для малих проєктів, також COCOMO II орієнтована на використання для оцінювання ПЗ, що розроблені за каскадною моделлю [8], що призводить до недостатньої ефективності при застосуванні до проєктів створених за Agile методологією.

Модель оцінювання трудомісткості на основі аналізу функціональних точок була запропонована у 1979 році Аланон Альбрехтом. Метод призначений для оцінки (на основі логічної моделі) розміру програмного коду, де в якості виміру його застосовують функціональні точки.

Функціональна точка (Function Point – FP) – це параметр, що визначає кількість бізнес-функцій, які інформаційна система (як продукт) надає користувачеві. Функціональні точки використовуються для розрахунку вимірювання функціонального розміру (FSM) програмної системи [27, с. 178].

Даний підхід побудований на обчисленні основних програмних компонентів, які включають зовнішні вхідні дані, зовнішні вихідні дані, зовнішні запити, логічні внутрішні файли та зовнішні інтерфейси. Кожна функція оцінюється за коефіцієнтом складності в діапазоні від низького, середнього до високого. Функціональна точка може бути визначена на етапі специфікації вимог або проєктування. Перевагою цього методу є те, що він не залежить від мови програмування або методології, що використовується в розробці програмного забезпечення. Недоліки цієї моделі полягають у тому, що вона не може використовуватися в ситуаціях, коли вимоги неповні або змінюються в процесі розробки ПЗ, та потребує багато часу для визначення функціоналу в FP [6; 7; 8].

Відомі також підходи до оцінювання трудомісткості на основі функціональних точок такі, як ESTIMACS та SPQR/20. ESTIMACS це інкрементний підхід, який складається з трьох етапів введення даних і еволюція оцінки, підбиття оцінок, аналіз чутливості оцінок. Основна увага зосереджена на ключових бізнес-факторах і впровадження нових факторів удосконалення оцінки. Оцінка залежить від параметрів – часу роботи, кількості персоналу, вартість, апаратного ризику, тощо. SPQR/20 є однією з перших моделей, яка включала параметри програмного проєкту такі як рядки коду а також проєктну документацію. Ця модель дозволяє прогнозувати якість, надійність та витрати на обслуговування ПЗ протягом 5 років після випуску [15].

Модель ISBSG (International Software Benchmarking Standards Group) – нелінійна регресійна модель для оцінювання часу розробки проєктів програмного забезпечення. Як фактор в цих регресійних моделях використано метрику трудомісткості розробки ПЗ:

$D = 0.458 * E^{0.366}$ (для платформи mainframe);

$D = 0.548 * E^{0.360}$ (для платформи MIDRANGE);

$D = 1.936 * E^{0.201}$ (для PC),

де *D* – тривалість проєкту (в місяцях); *E* – трудомісткість (в людино-годинах).

ISBSG підтримує велику базу проєктів ПЗ на різних мовах програмування. Набір даних містить інформацію щодо розміру та тривалості розробки програм, типів застосунків, методологій проєктування, складу команди розробників тощо. Інформація з бази ISBSG є найбільшою із доступних, яка може бути використана для розробки моделей оцінювання

трудомісткості, як алгоритмічних так і моделей на основі ШІ та машинного навчання [24].

Модель життєвого циклу програмного забезпечення Путнема SLIM (Software Life Cycle Management). SLIM одна з перших алгоритмічних моделей, розроблена у 1970-х роках. Вона дозволяє оцінити трудомісткість розробки на основі кількості рядків програмного коду (SLOC) і рівняння Релея. Формула моделі SLIM має наступний вигляд:

$TPM=1/RDT^4*(SLOC/TC)^3$,

де *TPM* (Total person months) – трудомісткість проєкту (в людино-місяцях), *SLOC* (Source Line of Code) – кількість рядків коду, *RDT* (required development time) – час розробки (в місяцях), *TC* (Technical Constant) – параметр, який враховує особливості розробки, такі як фреймворк, апаратна платформа та методологія проєктування. Модель SLIM застосовується для програмного забезпечення розміром більш ніж 70 тисяч рядків коду та базується на припущенні, що витрати на розробку програмного забезпечення розподіляються згідно кривих Норден-Рейлі [29]. Перевага моделі SLIM полягає у використанні підходу лінійного програмування для визначення вартості і трудомісткості.

До недоліків моделі слід віднести [15; 25]:

– обмеженість неможливість її застосування для оцінювання програмних проєктів невеликого розміру;

– отримані оцінки надзвичайно чутливі до технологічного фактору (*TC*).

AgileMOW – це модель оцінювання трудомісткості та вартості розробки програмного забезпечення для проєктів, які створені із використанням Agile методології. AgileMOW використовує експертні судження та параметри проєкту, які були описані в COCOMO II і відповідають принципам гнучкої методології розробки ПЗ. Складовими цього підходу є комунікативні навички, особливості організації команди, зворотній зв'язок, тощо. Таким чином, цей підхід використовує як веб-компоненти для оцінки розміру веб-додатку, так і враховує людські фактори. Трудомісткість обчислюються на основі параметрів розміру веб-застосунку та факторів, пов'язаних з особливостями команди розробників і середовища. Головною перевагою AgileMOW є те, що цей механізм оцінювання відповідає принципам Agile методології, таким як комунікація та взаємодія всередині команди та між командами для досягненні максимального результату. Недоліком є те, що ця модель може бути використана тільки для оцінювання розробки веб-застосунків. AgileMOW не має на меті замінити оцінки експертів, а пропонує механізм для досягнення достовірної оцінки трудомісткості розробки ПЗ для Agile методології [26].

Розглянуті вище ймовірнісні моделі не завжди враховують негаусівський характер розподілу емпіричних даних за метриками проєктів з розробки програмного забезпечення, що може призводити до зниження достовірності оцінювання. Крім того, такі моделі не дозволяють визначити довірчі інтервалів прогнозування трудомісткості та тривалості розробки ПЗ. Для того щоб підвищити достовірність оцінювання потрібно використовувати моделі, які будуть враховувати характер розподілу емпіричних даних з метрик проєктів з розробки ПЗ.

У роботах [17; 18; 19; 20; 21; 22; 23] проведені дослідження у напрямку підвищення достовірності оцінювання за допомогою нелінійних регресійних моделей для різних категорій проєктів, з урахуванням мов програмування та набору бібліотек та фреймворків на основі нормалізуючих перетворень. Так роботи [19; 20; 21] пропонують моделі, які підвищують достовірність оцінювання проєктів з розробки ПЗ, що створені мовою програмування PHP. В роботах [17; 27] представлени регресійні моделі для оцінювання параметрів JAVA проєктів. В роботах [22; 23] пропонуються моделі для підвищення достовірності оцінювання трудомісткості розробки мобільних застосунків та інформаційних систем на базі Visual Basic.

Таким чином, використання нелінійних регресійних моделей на основі нормалізуючих перетворень дає змогу покращити достовірність оцінювання трудомісткості та тривалості розробки проєктів ПЗ, враховуючи ймовірнісний характер розподілу емпіричних даних.

ВИСНОВКИ

1. Дослідження сучасного стану успішності розробки ІТ проєктів показало, що близько 20% таких проєктів є провальними, а 46% зазнають труднощів при виході на ринок. 2. Проведено аналіз алгоритмічних та неалгоритмічних методів оцінки трудомісткості розробки ПЗ, виявлені їх переваги та недоліки. 3. Визначено, що одним з основних факторів, який впливає на успішність виконання проєктів з розробки програмного забезпечення та дозволяє вчасно завершувати такі проєкти, є достовірність оцінювання трудомісткості та тривалості створення цих проєктів, яка в значній мірі залежить від моделей та методів оцінювання. 4. Перспективи подальших досліджень спрямовані на вдосконалення існуючих та розробку нових моделей оцінювання розробки ПЗ з урахуванням особливостей Agile методології, категорій ПЗ, фреймворків і мов програмування.

REFERENCES

[1] McConnel, S. (2006). *Software Estimation: Demystifying the Black Art*. Redmond, Washington: Microsoft Press.

[2] Johnson, J., & Mulder, H. (2021). Endless Modernization: How Infinite Flow Keeps Software Fresh. *The Standish Group, Hans Mulder's Lab.* https://www.researchgate.net/publication/348849361_Endless_Modernization_How_Infinite_Flow_Keeps_Software_Fresh

[3] The Standish Group. (2015). Chaos report 2015. https://standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf

[4] CISCO. (2017, May 23). Cisco Survey Reveals Close to Three-Fourths of IoT Projects Are Failing. *The Internet of Things World Forum (IoTWF)*. https://newsroom.cisco.com/c/r/newsroom/en/us/a/y2017/m05/cisco-survey-reveals-close-to-three-fourths-of-iot-projects-are-failing.html

[5] Gladka, M. V. (2021). Modeli ta metody multyahentnoho rozpodilu trudovykh resursiv v IT proektakh v umovakh nevyznachenosti [Models and methods of multiagent distribution of labor resources in IT projects in the conditions of uncertainty]. *Candidate's thesis.* Kyiv: Taras Shevchenko National University of Kyiv [in Ukrainian].

[6] Munialo, S.W., & Muketha, G.M. (2016). A review of Agile Software effort estimation methods. *International Journal of Computer Applications Technology and Research,* 5 (9), 612–618. https://doi.org/10.7753/ijcatr0509.1009

[7] Zaffar Iqbal, S. (2017). Comparative analysis of common software cost estimation modeling techniques. *Mathematical Modelling and Applications,* 2 (3), 33–39. https://doi.org/10.11648/j.mma.20170203.12

[8] Shekhar, S., & Kumar, U. (2016). Review of various software cost estimation techniques. *International Journal of Computer Applications,* 141 (11), 31–34. https://doi.org/10.5120/ijca2016909867

[9] Javdani Gandomani, T., Koh, T. W., & Binhamid, A. (2014). A Case Study Research on Software Cost Estimation Using Experts'. Estimates, Wideband Delphi, and Planning Poker Technique. *International Journal of Software Engineering and its Applications,* 8 (11), 173-182. http://dx.doi.org/10.14257/ijseia.2014.8.11.16

[10] Faria, P., & Miranda, E. (2012). Expert judgment in software estimation during the bid phase of a Project -- An Exploratory Survey. *2012 Joint Conference of the 22nd International Workshop on Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement*, 126–131. https://doi.org/10.1109/iwsm-mensura.2012.27

[11] Zia, Z., Kamal, T., & Ziauddin, Z. (2012). An Effort Estimation Model for Agile Software Development. *Advances in computer science and its applications (ACSA)*, 2 (1), 314–324.

[12] Ziauddin, Z., Kamal, S.,Khan, S., & Nasir, J.A. (2013). A fuzzy logic based software cost estimation model. *International Journal of Software Engineering and its Applications.* 7 (2), 7–18.

[13] Nassif, A. B., Azzeh, M., Capretz, L. F., & Ho, D. (2016). Neural network models for software development effort estimation: A comparative study. *Neural Computing and Applications*, 27 (8), 2369–2381. https://doi.org/10.1007/s00521-015-2127-1

[14] Benediktsson, O., Dalcher, D., Reed, K., & Woodman, M. (2003). COCOMO – Based Effort Estimation for Iterative and Incremental Software Development. *Software Quality Journal*, 11 (4), 265–281. https://doi.org/10.1023/a:1025809010217

[15] Bali, R. (2018). How To Determine And Selecting Automated Estimating Tools For Information Systems Development Project. https://doi.org/10.13140/RG.2.2.20690.45767

[16] Prykhodko, S. B., & Farionova, T. A. (2021). Nonlinear regression equations for estimating the duration of Java-applications development for the midrange platform. *ITMAS – 2021: Information Technologies: Models, Algorithms, Systems*, 2, 20–23. Mykolaiv: NUOS.

[17] Prykhodko, S. B., Prykhodko, N. V., & Smykodub, T. G. (2020). Four-factor non-linear regression model to estimate the size of open source Java-based applications. *Scientific Notes of Taurida National V.I. Vernadsky University. Series: Technical Sciences*, 1 (2), 157–162. https://doi.org/10.32838/2663-5941/2020.2-1/25

[18] Prykhodko, S. B., & Reznychenko, M.V. (2022). A mathematical model for evaluating the complexity of object-oriented design due to the relationships between classes of open-source applications in Java. *ITMAS – 2022: Information Technologies: Models, Algorithms, Systems*, 3, 7–8. Mykolaiv: NUOS.

[19] Latanska, L., Makarova, L., Koltsov, A., & Davlatova, D. (2022). A nonlinear regression model for estimating the size of WEB applications created using SYMFONY framework. *Herald of Khmelnytskyi national university. Technical Sciences № 315*, 6 (1), 119–124. https://doi.org/10.31891/2307-5732-2022-315-6-119-124

[20] Vorona, M.V. (2021). Matematychni modeli ta informatsiina tekhnolohiia dlia otsiniuvannia rozmiru prohramnykh zastosunkiv z vidkrytym kodom na PHP [Mathematical models and information technology for estimating the size of open source PHP-based applications]. *PhD's thesis*. Mykolaiv: Admiral Makarov National University of Shipbuilding [in Ukrainian].

[21] Prykhodko, S.B., & Vorona, M. V. (2021). Estimating the size of open-source PHP-based apps by nonlinear regression models with various factors. *Collection of Scientific Publications NUOS*, (1), 92–98. https://doi.org/10.15589/znp2021.1(484).13

[22] Prykhodko, S., Prykhodko, N., & Knyrik, K. (2020). Estimating the efforts of mobile application development in the planning phase using nonlinear regression analysis. *Applied Computer Systems*, 25 (2), 172–179. https://doi.org/10.2478/acss-2020-0019

[23] Prykhodko, N.V., & Prykhodko, S.B. (2018). Non-linear regression model to estimate the software size of VB-based information systems. *Information Technology And Computer Engineering*, 43 (3), 37–42. https://doi.org/10.31649/1999-9941-2018-43-3-37-42

[24] González-Ladrón-de-Guevara, F., Fernández-Diego, M., & Lokan, C. (2016). The usage of ISBSG data fields in software effort estimation: A systematic mapping study. *Journal of Systems and Software*, 113, 188–215. https://doi.org/10.1016/j.jss.2015.11.040

[25] Ghafory, H., & Sahnosh, F.A. (2020). The review of software cost estimation model: SLIM. *International Journal of Advanced Academic Studies*, 2(4), 511–515. https://doi.org/10.33545/27068919.2020.v2.i4h.447

[26] Litoriya, R., & Kothari, A. (2013). An efficient approach for agile web based project estimation: AgileMOV. *Journal of Software Engineering and Applications*, 06 (06), 297–303. https://doi.org/10.4236/jsea.2013.66037

[27] Project Management Institute (2021). *A guide to the Project Management Body of Knowledge (PMBOK® Guide) 7th ed*., Project Management Institute.

[28] Raslan, A. E., Ramadan, N., & Hefny, H. A. (2015). Effort Estimation in Agile Software Projects using Fuzzy Logic and Story Points. *50th Annual Conference on statistics, computer sciences, and operation research*, Cairo, Egypt. https://www.researchgate.net/publication/288839279_Effort_Estimation_in_Agile_Software_Projects_using_Fuzzy_Logic_and_Story_Points.

[29] Pukhalevich, A. (2017). Modeli ta informatsiina tekhnolohiia pererobky informatsii dlia otsiniuvannia tryvalosti proektiv z rozrobky prohramnoho zabezpechennia [Models and information processing information technology for estimating the duration of software development projects]. *Candidate's thesis.* Mykolaiv: Admiral Makarov National University of Shipbuilding [in Ukrainian].

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

[1] McConnel, S. (2006). *Software Estimation: Demystifying the Black Art*. Redmond, Washington: Microsoft Press.

[2] Johnson, J., & Mulder, H. (2021). Endless Modernization: How Infinite Flow Keeps Software Fresh. *The Standish Group, Hans Mulder's Lab*. https://www.researchgate.net/publication/348849361_Endless_Modernization_How_Infinite_Flow_Keeps_Software_Fresh

[3] The Standish Group. (2015). Chaos report 2015. https://standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf

[4] CISCO. (2017, May 23). Cisco Survey Reveals Close to Three-Fourths of IoT Projects Are Failing. *The Internet of Things World Forum (IoTWF)*. https://newsroom.cisco.com/c/r/newsroom/en/us/a/y2017/m05/cisco-survey-reveals-close-to-three-fourths-of-iot-projects-are-failing.html

[5] Гладка М.В. (2021). Моделі та методи мультиагентного розподілу трудових ресурсів в IT проектах в умовах невизначеності : дис. … кандидата тех. наук : 05.13.22 / Гладка Мирослава Вікторівна. – Київ : Київський національний університет імені Тараса Шевченка, 2021. – 247 с.

[6] Munialo, S.W., & Muketha, G.M. (2016). A review of Agile Software effort estimation methods. *International Journal of Computer Applications Technology and Research,* 5 (9), 612–618. https://doi.org/10.7753/ijcatr0509.1009

[7] Zaffar Iqbal, S. (2017). Comparative analysis of common software cost estimation modeling techniques. *Mathematical Modelling and Applications,* 2 (3), 33–39. https://doi.org/10.11648/j.mma.20170203.12

[8] Shekhar, S., & Kumar, U. (2016). Review of various software cost estimation techniques. *International Journal of Computer Applications,* 141 (11), 31–34. https://doi.org/10.5120/ijca2016909867

[9] Javdani Gandomani, T., Koh, T. W., & Binhamid, A. (2014). A Case Study Research on Software Cost Estimation Using Experts'. Estimates, Wideband Delphi, and Planning Poker Technique. *International Journal of Software Engineering and its Applications,* 8 (11), 173-182. http://dx.doi.org/10.14257/ijseia.2014.8.11.16

[10] Faria, P., & Miranda, E. (2012). Expert judgment in software estimation during the bid phase of a Project -- An Exploratory Survey. *2012 Joint Conference of the 22nd International Workshop on Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement*, 126–131. https://doi.org/10.1109/iwsm-mensura.2012.27

[11] Zia, Z., Kamal, T., & Ziauddin, Z. (2012). An Effort Estimation Model for Agile Software Development. *Advances in computer science and its applications (ACSA)*, 2 (1), 314–324.

[12] Ziauddin, Z., Kamal, S.,Khan, S., & Nasir, J.A. (2013). A fuzzy logic based software cost estimation model. *International Journal of Software Engineering and its Applications.* 7 (2), 7–18.

[13] Nassif, A. B., Azzeh, M., Capretz, L. F., & Ho, D. (2016). Neural network models for software development effort estimation: A comparative study. *Neural Computing and Applications*, 27 (8), 2369–2381. https://doi.org/10.1007/s00521-015-2127-1

[14] Benediktsson, O., Dalcher, D., Reed, K., & Woodman, M. (2003). COCOMO – Based Effort Estimation for Iterative and Incremental Software Development. *Software Quality Journal*, 11 (4), 265–281. https://doi.org/10.1023/a:1025809010217

[15] Bali, R. (2018). How To Determine And Selecting Automated Estimating Tools For Information Systems Development Project. https://doi.org/10.13140/RG.2.2.20690.45767

[16] Приходько, С.Б., & Фаріонова, Т.А. (2021). Нелінійні регресійні рівняння для оцінювання тривалості розробки JAVA-застосунків для платформи MIDRANGE. *ITMAS – 2021. Інформаційні Технології: Моделі, Алгоритми, Системи,* 2, 20–23. Миколаїв: НУК ім. адм. Макарова

[17] Приходько, С.Б., Приходько, Н.В., & Смикодуб, Т.Г. (2020). Чотирьохфакторна нелінійна регресійна модель для оцінювання розміру JAVA-застосунків з відкритим кодом. *Вчені записки ТНУ імені В.І. Вернадського. Серія: технічні науки*, 1 (2), 157–162. Режим доступу – https://doi.org/10.32838/2663-5941/2020.2-1/25

[18] Приходько С. Б., & Резниченко М.В. (2022). Математична модель для оцінювання складності об'єктно-орієнтованого проектування через зв'язки між класами застосунків з відкритим кодом на JAVA. *ITMAS – 2022. Інформаційні Технології: Моделі, Алгоритми, Системи*, 3, 7–8. Миколаїв: НУК ім. адм. Макарова.

[19] Латанська, Л., Макарова, Л., Кольцов, А., & Давлатова, Д. (2022). Нелінійна регресійна модель для оцінювання розміру веб-застосунків, що створюються з використанням PHP фреймворку SYMFONY. *Вісник Хмельницького національного університету. Технічні науки №315*, 6 (1), 119–124. Режим доступу – https://doi.org/10.31891/2307-5732-2022-315-6-119-124

[20] Ворона, М.В. (2021). Математичні моделі та інформаційна технологія для оцінювання розміру програмних застосунків з відкритим кодом на PHP : дис. … доктора філософії (PhD) / Ворона Михайло Владиславович. – Миколаїв : Національний університет кораблебудування імені адмірала Макарова, 2021. – 247 с.

[21] Приходько, С.Б., & Ворона, М.В. (2021). Оцінювання розміру PHP-застосунків з відкритим кодом за нелінійними регресійними моделями з різними факторами. *Збірник наукових праць НУК*, (1), 92–98. Режим доступу – https://doi.org/10.15589/znp2021.1(484).13

[22] Prykhodko, S., Prykhodko, N., & Knyrik, K. (2020). Estimating the efforts of mobile application development in the planning phase using nonlinear regression analysis. *Applied Computer Systems*, 25 (2), 172–179. https://doi.org/10.2478/acss-2020-0019

[23] Приходько, Н. В., & Приходько, С. Б. (2018). Нелінійна регресійна модель для оцінювання розміру програмного забезпечення інформаційних систем на базі VB. *Інформаційні технології та комп'ютерна інженерія*, 43(3), 37–42. Режим доступу – https://doi.org/10.31649/1999-9941-2018-43-3-37-42

[24] González-Ladrón-de-Guevara, F., Fernández-Diego, M., & Lokan, C. (2016). The usage of ISBSG data fields in software effort estimation: A systematic mapping study. *Journal of Systems and Software*, 113, 188–215. https://doi.org/10.1016/j.jss.2015.11.040

[25] Ghafory, H., & Sahnosh, F.A. (2020). The review of software cost estimation model: SLIM. *International Journal of Advanced Academic Studies*, 2(4), 511–515. https://doi.org/10.33545/27068919.2020.v2.i4h.447

[26] Litoriya, R., & Kothari, A. (2013). An efficient approach for agile web based project estimation: AgileMOV. *Journal of Software Engineering and Applications*, 06 (06), 297–303. https://doi.org/10.4236/jsea.2013.66037

[27] Project Management Institute (2021). *A guide to the Project Management Body of Knowledge (PMBOK® Guide) 7th ed*., Project Management Institute.

[28] Raslan, A. E., Ramadan, N., & Hefny, H. A. (2015). Effort Estimation in Agile Software Projects using Fuzzy Logic and Story Points. *50th Annual Conference on statistics, computer sciences, and operation research*, Cairo, Egypt. https://www.researchgate.net/publication/288839279_Effort_Estimation_in_Agile_Software_Projects_using_Fuzzy_Logic_and_Story_Points.

[29] Пухалевич, А.В. (2017). Моделі та інформаційна технологія переробки інформації для оцінювання тривалості проектів з розробки програмного забезпечення : дис. … кандидата тех. наук : 05.13.06 / Пухалевич Андрій Володимирович. – Миколаїв : Національний університет кораблебудування імені адмірала Макарова, 2017. – 179 с.

Дата надходження статті до редакції: 14.02.2024
Дата затвердження статті до друку: 29.02.2024