УДК 004.4`6 DOI: 10.30837/0135-1710.2025.185.005

А.С. КРУГЛИК, В.М. ЛЕВИКІН, М.В. ЄВЛАНОВ, Б.І. МОРОЗ, Д.М. МОРОЗ

МОДИФІКАЦІЯ МОДЕЛІ ОДНОМІСНОГО КОВАРІАНТНОГО ФУНКТОРА ДЛЯ ПРОЦЕСУ КРОСПЛАТФОРМНОЇ МІГРАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Розглянуто основні особливості та недоліки існуючих рішень з формального опису процесу кросплатформної міграції програмного забезпечення інформаційних систем. Для формального опису цього процесу запропоновано модифікувати модель одномісного коваріантного функтора. Розроблено два основних варіанти модифікації цієї моделі з врахуванням можливості виконання процесу кросплатформної міграції як під час експлуатації ІС, так і під час валідації ІС до початку її експлуатації. Проведено експериментальну перевірку отриманих результатів.

1. Вступ

Переведення бізнесів у цифрову площину ставить виклики до ІТ-галузі. Зокрема, постійно зростає навантаження на ІТ-інфраструктуру підприємств та обсяги оброблюваної інформації. Залежно від різновиду бізнесу, його стратегії та інших факторів, перед ІТ-сектором ставляться завдання знаходити дешевші рішення, або ж більш захищенні, або надійніші рішення. Виходячи із сукупності цих факторів, ІТ-компанії можуть приймати рішення стосовно використання тієї чи іншої платформи для роботи інформаційних систем (ІС), які автоматизують бізнес-логіку підприємства, або навіть стосовно підтримки кількох платформ одночасно.

В основі рішень фахівців ІТ-компаній щодо кросплатформної міграції або кросплатформної підтримки ІС, як правило, знаходяться дві протилежні за сенсом концепції.

Перша концепція: розробити рішення, яке буде унікальним і апріорно орієнтованим на вирішення конкретної проблеми, яка виникає для конкретної ІС конкретного підприємства-замовника тільки один раз за весь життєвий цикл цієї ІС.

Друга концепція: розробити рішення, яке буде типовим (шаблонним) і апріорно орієнтованим на вирішення множини східних між собою проблем, які виникають багато разів для тієї самої ІС та її варіантів конфігурації.

Слід зазначити, що значну увагу в галузі міграції сучасні дослідження приділяють саме розробці рішень на основі другої концепції. Прикладами таких рішень є фреймворки хмарної міграції, в яких основну увагу приділено структуруванню та визначенню окремих етапів і їх організації в єдиний життєвий цикл міграції [1], [2]. Але такі рішення вимагають формальних описів ІС та її елементів. Дослідження, проведені у 2010-2018 рр., визнали як найрозповсюдженіші такі засоби формального опису ІС:

– структуровані текстові документи, які описують вихідний програмний код, додаткову контекстну інформацію (трасування виконання та початкову сутність вихідного коду, перевірену на наявність змін) та запити на зміни [3];

– профілі прототипів програмного забезпечення (ПЗ), які використовувалися для виявлення і документування функціональних вимог до створюваного ІТ-продукту [4];

– спеціальні структурні моделі [5].

Аналізуючи досвід цих досліджень та власні доробки у даному напрямі, автори [6], [7], визнали, що основним артефактом, який відображає проєктні рішення та вплив проєктних рішень на технічні артефакти та організаційні обов'язки, є архітектура ПЗ ІС. Але саме обмеження артефактами процесу розробки ПЗ автори [6] визнали одним з

головних недоліків застосування формальних описів для підтримки і супроводження ІС. Для подолання цього недоліку у [7] було представлено інструментальний підхід для оцінки поширення змін, спричинених запитом на зміну в бізнес-процесах або програмних системах, на основі архітектури ПЗ ІС та дизайну процесу. Але застосування запропонованого у [7] підходу та аналогічних йому для супроводження та управління змінами ІС під час експлуатації і, зокрема, під час кросплатформної міграції ПЗ ІС можливе тільки за дотримання таких умов:

– наявність множини адекватних моделей, які повністю описували б бізнес-процеси підприємства, ІС та її ПЗ;

– необхідність зберігання структурних та поведінкових особливостей бізнес-процесів підприємства, ІС та її ПЗ під час трансформації моделей одна в іншу, а також моделей у відповідні фрагменти програмного коду або команди.

Дотримання цих умов вимагає існування формального механізму, який би забезпечував існування узгоджених між собою моделей бізнес-процесів підприємства, ІС та її ПЗ, а також підтримував би цілісність структурних та поведінкових особливостей цих моделей під час їх взаємної трансформації. Необхідність існування і застосування такого формального механізму непрямо підтверджується дослідженням можливості використання засобів штучного інтелекту для вирішення задачі класифікації та пріоритизації запитів на зміну у ІТ-проєкті [8]. Якщо для повних та адекватних моделей запитів на зміну та ІС застосування методу Random Forest дозволило передбачити можливу зміну пріоритетності запитів на зміну з точністю 94 % і визначити збільшення або зменшення пріоритетності цієї зміни з точністю 93 %, то в ситуаціях із спотвореними даними, на основі яких формувалися моделі запитів на зміну, точність моделі становила 68 %, що, за словами авторів [8], відповідає опублікованим результатам сучасних досліджень. Це означає, що спроби замінити цей формальний механізм методами і засобами штучного інтелекту не гарантують рішення кращого, ніж ручна обробка таких моделей кваліфікованим аналітиком. Саме тому проведення досліджень з розробки і вдосконалення такого формального механізму є актуальним з теоретичної та прикладної точок зору.

2. Аналіз літературних даних і постановка проблеми дослідження

Досвід, накопичений дослідниками та фахівцями ІТ-компаній, дозволяє серед напрямів розвитку засобів формального механізму управління підтримкою та супроводженням ПЗ особливо виділити множину семантичних методів. Серед цієї множини методів найчастіше виділяють [9]:

– методи структурної операційної семантики (описують поведінку програми у вигляді змін станів, спричинених виконанням елементарних кроків);

– методи денотаційної семантики (визначають зміни станів функціями).

Кожна з цих груп методів має власні переваги та недоліки, які ускладнюють їх прикладне застосування. Тому у [9] було запропоновано новий підхід до управління семантикою ПЗ: поведінка програм, тобто зміни станів, моделюється як категорії цих станів. При цьому категорія морфізмів виражає множину елементарних кроків виконання, а виконання конкретної програми є орієнтованим шляхом у категорії, тобто композицією морфізмів [9]. Для розвитку підходу було використано можливість зв'язування між собою категорних моделей системою функторів, які у запропонованому підході реалізовано у вигляді окремих процедур. Така реалізація дозволила формально описати повторний виклик процедур, вкладення викликів процедур та рекурсивні виклики. Хоча запропонований у [9] підхід достатньо простий і точний, він має суттєві обмеження, тому що його формальна база побудована виключно для простої процедурної мови, яка містить усі основні конструкції ван Дейкстри.

Використання математичного апарату теорії категорій як формальної основи механізму забезпечення існування та підтримки цілісності узгоджених між собою моделей бізнес-процесів підприємства, ІС та її ПЗ залишається одним з основних способів створення такого механізму. Дана точка зору обумовлена у [10] такими причинами:

– теорія категорій пропонує інтегроване бачення концепцій моделі, а також надає механізми для об'єднання моделей, механізми для міграції між моделями та механізми для побудови мостів між моделями;

– категорія, як і модель, вважається сумішшю графічної інформації та алгебраїчних операцій, тому мова категорій здається найзагальнішою для опису моделей;

– теорія категорій дозволяє спростити опис того, як людина мислить та використовує окремі моделі.

Але прикладне застосування теоретико-категорних моделей в ІТ-галузі продовжує викликати значні проблеми. Так, у [11] для створення моделі лінійки програмних продуктів з оновленнями було запропоновано розглянути системи умовних переходів в коалгебраїчному середовищі, елементами яких є еквівалентні категорії Клейслі, в яких знаходяться ці коалгебри. В цих системах функтори використано для формального опису типу розгалуження коалгебри програмного продукту. Одним з результатів використання таких категорно-коалгебраїчних систем для моделювання лінійки програмних продуктів з оновленнями є можливість створення екземпляру існуючого алгоритму мінімізації коалгебри для виведення поведінкових еквівалентностей у цьому середовищі [11]. Такий екземпляр алгоритму дозволяє забезпечити поведінкову еквівалентність оновленої версії програмного продукту із мінімальними змінами коалгебри операцій оновленої версії. Але запропоновані у [11] рішення мають два значні недоліки, які обмежують їх прикладне застосування, а саме:

– необхідність наявності у розробників формального механізму забезпечення існування та підтримки цілісності узгоджених між собою моделей бізнес-процесів підприємства, ІС та її ПЗ значного обсягу теоретичних знань в галузі сучасної загальної алгебри (теорія категорій, теорія представлень, універсальна алгебра тощо);

– перед початком програмної реалізації запропонованих систем умовних переходів необхідно довести існування адекватної реалізації розробленої формальної бази таких систем засобами конкретної мови або середовища програмування.

Крім того, запропоновані у [11] системи категорно-коалгебраїчних моделей описують виключно артефакти процесу супроводження ПЗ, що, як показано у розділі 1, є одним з головних недоліків застосування формальних описів для підтримки і супроводження ІС.

Для подолання цих недоліків у [12] було запропоновано нову формульну рамку опису моделі архітектури ПЗ, в якій теорія категорій, алгебраїчні специфікації та алгебра процесів були поєднані разом:

– компоненти ПЗ були описані за допомогою об'єктів категорій;

– різні типи морфізмів зображували семантику, що міститься у зв'язках компонентів;

– функтори теорії категорій описували зв'язки між моделями архітектур, представленими типізованими діаграмами категорій.

Отримані у [12] теоретичні результати були орієнтовані на визначення того, чи відповідає перетворення певним характеристикам або обмеженням, та реалізовані у вигляді системи спільної розробки ПЗ. Встановлено, що такий підхід розширює можливості семантичного опису моделі архітектури та може бути використаний як ефективне доповнення до існуючого методу моделювання ПЗ [12]. Але подібне рішення не враховує зміни, які можуть виникати у ПЗ внаслідок змін бізнес-процесів та ІТ-інфраструктури, в межах якої експлуатується це ПЗ.

Розвитком результатів, отриманих у [12], є розроблена у [13] єдина основна структура семантичного опису, призначена для опису та перетворення компонентно-орієнтованих моделей ПЗ, а також для підтримки та перевірки семантичних властивостей у процесі перетворення моделі. Формальною основою цієї структури семантичного опису є розширення теорії типізованих категорій до алгебри процесів. Крім цього, отримані у [13] результати орієнтовані на опис не стільки ПЗ, скільки його моделей, а саме:

– діаграма категорій використовується для опису семантики архітектурної моделі ПЗ;

– типізований морфізм описує зв'язок залежності між компонентними об'єктами категорії;

– типізований функтор використовується для опису механізму відображення до та після перетворення моделі.

Прикладні дослідження показують, що отримані результати добре відповідають сутності та вимогам процесу модельно-орієнтованої розробки та забезпечують нову основну структуру для розуміння, когнітивного навчання та просування досліджень у галузі розробки ПЗ на основі модельно-орієнтованого підходу [13].

Новою у результатах, отриманих у [13], є їхня орієнтація на формальний опис не компонентів ПЗ (фрагментів програмного коду), а моделей цих компонентів, для опису яких застосовуються згадувані у розділі 1 засоби формального опису ІС. Але при цьому залишаються малодослідженими або зовсім не дослідженими дуже багато питань, пов’язаних із прикладними аспектами застосування запропонованої у [13] структури семантичного опису. Зокрема, залишається незрозумілим спосіб забезпечення еквівалентності категорного-функторного представлення моделей ПЗ, самих моделей та їх програмної реалізації з врахуванням обмежень середовищ розробки та експлуатації ПЗ. Знову залишається невирішеною проблема відображення поточних варіантів та можливих змін у бізнес-процесах підприємства у категорно-функторні формальні описи семантики ПЗ, яке експлуатується в межах цих бізнес-процесів.

Один з поглядів на загальну перспективу застосування категорно-функторного апарату для семантичного моделювання ПЗ розглянуто у [14]. Ця перспектива має такі особливості [14]:

– моделювання денотації в межах категорій, враховуючи вхідні/вихідні значення та динаміку станів компонентів ПЗ;

– зосередження уваги на з'ясуванні зв'язків між об'єктами, що представляють стани пам'яті як денотаційні функції;

– цілеспрямоване використання коалгебр для вираження поведінки програми, особливо щодо конфігурацій пам'яті;

– представлення динаміки виконання через налаштований поліноміальний ендофунктор, що враховує ряд типів переходів.

Очікуваним результатом використання цієї перспективи є отримання категорії, адаптованої для конфігурацій пам'яті, яка ефективно відображає переходи програми, тобто встановлення переходу між описом ПЗ та пам’яттю як елементом ІТ-інфраструктури, в межах якої планується експлуатація цього ПЗ [14]. Але дослідники розуміють, що навіть така перспектива хоча й підкреслює незмінну актуальність теорії категорій у динамічному ландшафті семантичного моделювання, пропонує розуміння лише потенційних практичних застосувань у програмуванні та суміжних галузях.

Ще одним напрямом сучасних досліджень способів формального опису процесів супроводження та міграції ПЗ є використання положень теорії категорій для опису окремих випадків та різновидів трансформації ПЗ. Цей напрям, по суті, є спробою формально узагальнити прикладний досвід розробки та супроводження ПЗ різного

призначення і розглядає категорно-функторний апарат як своєрідні формальні патерни, які встановлюють загальні особливості виконання тих чи інших операцій. Як приклади робіт за цим напрямом можна вказати такі дослідження:

– розробка теоретичної бази, яка забезпечує об'єднувальну мову для різних існуючих симпліціальних фільтрацій, а також є механізмом для генерації довільно великих сімейств нових функторів фільтрації з контролем залежності/незалежності від базової точки, а також локальності фільтрації [15];

– розробка так званих функторів Меркла як складних дерев з багаторазових будівельних блоків, що включають суми, добутки та функціональні простори та є замкнутими відносно композиції та найменших точок виправлення, та застосування цих функторів як важливого елементу формальної основи протоколу Canton та для перевірки цілісності й гарантій безпеки цього протоколу, а також тестування нових фільтрацій на наборах даних у блокчейнах та інших технологіях розподілених реєстрів [16];

– використання функторів C++ як механізмів адаптації загальних алгоритмів (інтегрування, інтерполяція, мінімізація та процедури пошуку коренів) до різних типів даних та/або типів функцій у програмному пакеті C++ для розрахунку паралельних розподілів імпульсів (включаючи механізм відриву та механізм дифракційної дисоціації) важкого залишку (ядра) в реакціях нокауту окремих нуклонів, викликаних пучками стабільних та радіоактивних атомних ядер середньої енергії [17];

– використання теорії категорій для моделювання та управління потоками даних та залежностями в системі, зокрема за допомогою функторів та монад, в процесах створення адаптивних та масштабованих архітектур ігрових додатків, враховуючи специфіку компонентно-орієнтованого та сервісно-орієнтованого архітектурних стилів в умовах зміни вимог та умов експлуатації [18];

– використання категорно-функторного апарату для формального опису патернів проєктування вимог до ІС, які визначають моделі основних процесів інженерії вимог та синтезу опису архітектури створюваної ІС [19].

Проведений аналіз сучасних досліджень дозволяє зробити такі висновки:

– проблема формального опису механізму управління підтримкою та супроводженням ПЗ ІС ще досить далека від повного вирішення;

– застосування апарату функторів спрямовано, переважно, на формальний опис елементів механізму управління підтримкою та супроводженням ПЗ ІС, які забезпечують зберігання структурних особливостей програмних систем та продуктів, а також їхніх компонентів, під час трансформацій;

– існуючі категорно-функторні моделі орієнтовано, головним чином, на опис ПЗ та його окремих компонентів, як перспектива – на опис переходу між описом ПЗ та пам’яттю як елементом ІТ-інфраструктури.

– проблеми формального опису трансформації ПЗ базової (типової) ІС до особливостей ІТ-інфраструктури конкретних підприємств, які є споживачами різноманітних варіантів конфігурації цієї ІС, а також впливу на архітектуру ПЗ запитів на зміни, які виникають у бізнес-процесах підприємств, залишаються майже недослідженими.

Виходячи з цих висновків, проблему даного дослідження запропоновано сформулювати як проблему модифікації категорно-функторних моделей з врахуванням особливостей опису трансформації ПЗ ІС в процесі кросплатформної міграції, яка виникає за вимогами адаптації типової ІС до особливостей ІТ-інфраструктури конкретних підприємств-споживачів цієї ІС. Вирішення цієї проблеми дозволить отримати результати, що будуть викликати інтерес як у науковців в галузі комп’ютерних наук, так і у фахівців-практиків з міграції ПЗ ІС.

3. Мета і задачі дослідження

Метою даного дослідження є модифікація моделі функтора для опису трансформаціїї програмного забезпечення ІС в межах її кросплатформної міграції. Досягнення цієї мети дозволить визначити формальний опис елементів процесу кросплатформної міграції програмного забезпечення ІС на рівні її семантичного опису, що дасть змогу зменшити витрати ресурсів та часу на виконання цього процесу на вимоги значної кількості підприємств-споживачів ІС.

Для досягнення цієї мети треба вирішити такі задачі:

– модифікація елементів моделі одномісного коваріантного функтора для опису особливостей кросплатформної міграції програмного забезпечення ІС;

– експериментальна перевірка модифікованої моделі функтора в процесі рефакторингу вихідного програмного коду мігруючого ПЗ ІС.

4. Моделі і технології, які використовуються у дослідженні

Об'єктом даного дослідження є процес кросплатформної міграції ПЗ ІС управління підприємством.

Основною гіпотезою даного дослідження є гіпотеза про можливість застосування моделі функтора, який зв'язує дві категорні моделі ПЗ ІС, для опису елементів процесу кросплатформної міграції цього ПЗ.

Ідея використання функторів як засобів формалізованого опису елементів механізму управління підтримкою та супроводженням ПЗ ІС, які забезпечують зберігання структурних особливостей категорної моделі ПЗ ІС під час її трансформації, поставила питання про можливість розробки єдиного підходу до формалізованого опису функторів. В даному дослідженні запропоновано в загальному випадку використовувати модель одномісного коваріантного функтора. Під час побудови цієї моделі було прийнято позначення похідної категорної моделі, яка є початком функтора, як категорії A, а кінцевої категорної моделі, яка є кінцем функтора, як категорії B. З врахуванням цих позначень узагальнена модель одномісного коваріантного функтора має такий вигляд [19]-[21]:

$$\Phi_B^A = (Sc_{Tez(A)}, Ob^A, Mor^A, Sc_{Tez(B)}, Ob^B, Mor^B, \Phi_{Ob^A}^{Ob^B}, \Phi_{Mor^A}^{Mor^B}), \quad (1)$$

де $Sc_{Tez(A)}$ – рівень представлення фрагмента тезаурусу, що використовується похідною категорією A; Ob^A – клас об'єктів похідної категорії A, які описують елементи класу об'єктів похідної категорної моделі; Mor^A – клас морфізмів похідної категорії A, які описують зв'язки елементів класу об'єктів похідної категорної моделі; $Sc_{Tez(B)}$ – рівень представлення фрагмента тезаурусу, що використовується кінцевою категорією B; Ob^B – клас об'єктів кінцевої категорії B, які описують елементи класу об'єктів кінцевої категорної моделі; Mor^B – клас морфізмів кінцевої категорії B, які описують зв'язки елементів класу об'єктів кінцевої категорної моделі; $\Phi_{Ob^A}^{Ob^B}$ – система правил трансформації елементів класу об'єктів похідної категорії A в елементи класу об'єктів кінцевої категорії B; $\Phi_{Mor^A}^{Mor^B}$ – система правил трансформації елементів класу морфізмів похідної категорії A в елементи класу морфізмів кінцевої категорії B.

Модель (1) може існувати тільки при дотриманні таких умов [22]:

$$\forall a \in Ob^A \quad \exists \Phi_B^A(a) \in Ob^B \; ; \quad (2)$$

$$\forall \alpha \in H_A(a_i, a_j) \subseteq Mor^A \ \exists \Phi_B^A(\alpha) \in H_B(\Phi_B^A(a_i), \Phi_B^A(a_j)) \in Mor^B \ ; \quad (3)$$

$$\forall 1_a \in Mor^A \ \exists \Phi_B^A(1_a) = 1_{\Phi_B^A(a)} \in Mor^B \ ; \quad (4)$$

$$\forall \alpha \in H_A(a_i, a_j) \in Mor^A, \beta \in H_A(a_j, a_k) \in Mor^A \ \exists \Phi_B^A(\alpha\beta) = \Phi_B^A(\alpha)\Phi_B^A(\beta) \in Mor^B, \quad (5)$$

де a – будь-який об'єкт, що належить до класу об'єктів Ob^A похідної категорії A; $H_A(a_i, a_j)$ – множина морфізмів, визначених в похідній категорії A для об'єкта a_i як початку морфізму та об'єкта a_j як кінця морфізму, при цьому $i \neq j$ є ідентифікаторами об'єктів похідної категорії A; α – морфізм, який є елементом множини $H_A(a_i, a_j)$; $H_B(\Phi_B^A(a_i), \Phi_B^A(a_j))$ – множина морфізмів, визначених у кінцевій категорії B для об'єкта $\Phi_B^A(a_i)$ як початку морфізму та об'єкта $\Phi_B^A(a_j)$ як кінця морфізму; 1_a – одиничний морфізм, визначений для об'єкта a похідної категорії A; $1_{\Phi_B^A(a)}$ – одиничний морфізм, визначений для відповідного об'єкта кінцевої категорії B; $\alpha\beta$ – часткова бінарна операція здобутку морфізмів похідної категорії A.

Крім умов (2)-(5), обумовлених особливостями формального апарату теорії категорій, на модель (1) накладено додаткові умови існування, які мають вигляд:

$$Ob^A \subseteq \Phi_{Ob^B}^{Ob^A}(Ob^A) \subseteq Ob^B, \quad Mor^A \subseteq \Phi_{Mor^B}^{Mor^A}(Mor^A) \subseteq Mor^B. \quad (6)$$

Умову (6) для моделей, які описуються категоріями A та B, слід інтерпретувати таким чином: фрагмент тезаурусу, що утворює похідну модель, яку описано категорією A, не повинен перевищувати за розмірами та складністю фрагмент тезаурусу, що утворює кінцеву модель, яку описано категорією B [19]. Але ця умова повинна виконуватися тільки у випадках, коли рівні тезаурусів $Sc_{Tez(A)}$ та $Sc_{Tez(B)}$ категорій будуть еквівалентні один одному.

Модель (1) дозволяє зробити висновок про існування двох основних способів реалізації запропонованого формалізованого опису одномісного коваріантного функтора. Перший спосіб передбачає пошук таких систем правил, які однозначно визначали б відповідність заздалегідь заданих елементів похідної і кінцевої категорій. Другий спосіб передбачає формування таких об'єктів і морфізмів категорії B, які задовольняли б заздалегідь заданим системам правил трансформації, об'єктам і морфізмам похідної категорії A, а також враховували б різницю рівнів подання тезаурусів моделей $Sc_{Tez(A)}$ і $Sc_{Tez(B)}$. З прикладної точки зору, перший спосіб реалізації функтора передбачає формування та реалізацію якогось загального алгоритму або бази правил перетворення похідного представлення ПЗ в кінцеве представлення цього ж ПЗ або множини можливих результатів його трансформації. При цьому передбачається, що текстові конструкції або елементи візуальних моделей, що описують це ПЗ, апріорно визначені як елементи такого алгоритму або бази правил і практично не змінюються з часом. Другий спосіб реалізації функторів передбачає створення та постійний розвиток деякої множини алгоритмів або якоїсь системи правил перетворення похідного представлення ПЗ в кінцеве представлення цього ж ПЗ або множини можливих результатів його трансформації. При цьому темпи зміни цієї множини алгоритмів або систем правил визначаються темпами появи нових або

зміни існуючих понять тезаурусу похідного представлення ПЗ. Поява кожного нового, раніше невідомого поняття у такому представленні ПЗ вимагає доповнення існуючого варіанту реалізації функтора новими алгоритмами чи правилами, які встановлюють спосіб і форму опису такого нового поняття у кінцевому представленні цього ж ПЗ або множини можливих результатів його трансформації.

Для проведення експериментальних досліджень було запропоновано обрати ПЗ спеціалізованої інформаційно-аналітичної системи медичного закладу (ІАСМЗ). Головною метою створення ІАСМЗ була автоматизація процесів обліку, контролю, аналізу та прийняття рішення за результатами здійснення клінічних лабораторних аналізів. Для досягнення цієї мети в ІАСМЗ реалізовано такі функціональні модулі:

– «Облік результатів клінічних аналізів пацієнта, введених оператором з власного робочого місця»;

– «Облік результатів клінічних аналізів пацієнта, сформованих системою управління приладом, який виконував аналіз, і переданих до ІАСМЗ за допомогою спеціального бізнес-сервісу ІАСМЗ»;

– «Контроль результатів клінічних аналізів пацієнта, введених до ІАСМЗ»;

– «Аналіз введених до ІАСМЗ результатів клінічних аналізів пацієнта»;

– «Розрахунки та публікація калькуляції результатів клінічних аналізів пацієнта»;

– «Обґрунтування та підтримка прийняття рішення за результатами дослідження клінічних аналізів пацієнта».

Під час розробки ПЗ ІАСМЗ було прийнято рішення використати клієнт-серверну архітектуру із застосуванням для клієнтської та серверної частин тієї самої мови програмування. Як базову мову програмування було обрано C/C++. Головне призначення серверної частини ПЗ – реалізація бізнес-логіки функцій ІАСМЗ. Головне призначення клієнтської частини ПЗ – реалізація користувацького інтерфейсу з використанням GUI-фреймворків (QT framework, MFC framework тощо).

Під час експлуатації ІАСМЗ виникла задача підтримки та супроводження її ПЗ для платформ Linux та AIX від IBM. Основну увагу під час даного дослідження було зосереджено на вирішенні питань кросплатформної міграції та підтримки серверної частини ПЗ ІАСМЗ.

5. Вирішення задачі модифікації моделі одномісного коваріантного функтора

5.1. Результати модифікації елементів моделі одномісного коваріантного функтора

В процесі дослідження було встановлено, що хід та результати модифікації моделі одномісного коваріантного функтора з врахуванням особливостей процесу кросплатформної міграції ПЗ ІС значною мірою залежать від бажаного для виконавців робіт варіанта реалізації цієї моделі в межах відповідної інформаційної технології. При цьому уся множина можливих варіантів реалізації може бути зведена до двох базових варіантів:

Перший варіант – використання першого способу реалізації моделі одномісного коваріантного функтора для кожного окремого випадку кросплатформної міграції (результатом є база правил, кожен окремий фрагмент якої відповідає за кросплатформну міграцію ПЗ до відповідної платформи).

Другий варіант – використання другого способу реалізації моделі одномісного коваріантного функтора для усіх підтримуваних випадків кросплатформної міграції (результатом є система правил, кожен окремий фрагмент якої відповідає за формування та підтримку бази правил кросплатформної міграції ПЗ до відповідної платформи).

На формальному рівні перший варіант запропоновано представити конусом одномісних коваріантних функторів, кожен з яких описує набір правил кросплатформної

міграції з платформи *A* до платформи *B*. Цей конус має такий вигляд:

$$Sw_{op}^{i} \xrightarrow{F_j} Sw_{op}^{j}, j = 1,...,n, \quad (7)$$

де Sw_{op}^{i} – опис похідної реалізації ПЗ експлуатованої ІС, виконаної в межах платформи Pl_i, i=1,2,…; Sw_{op}^{j} – опис кінцевої реалізації ПЗ експлуатованої ІС, виконаної в межах платформи Pl_j; F_j – позначення одномісного коваріантного функтора; n – кількість платформ, для яких слід підтримувати процес кросплатформної міграції ПЗ ІС.

В цьому варіанті модель одномісного коваріантного функтора (1)-(6) повинна враховувати умову, за якою рівні тезаурусів $Sc_{Tez(A)}$ та $Sc_{Tez(B)}$ категорій еквівалентні один одному. Але при цьому неможливо або недоцільно встановити наявність якогось вищого рівня тезаурусу $Sc_{MTez(A)} \xleftarrow{iso} Sc_{MTez(B)}$, який був би єдиним для обох рівнів тезаурусів $Sc_{Tez(A)}$ та $Sc_{Tez(B)}$ категорій, що описують ПЗ ІС для платформ *A* та *B* відповідно. Тому модель одномісного коваріантного функтора запропоновано модифікувати для першого варіанту шляхом зміни умов (6) таким чином:

$$Sc_{Tez(A)} \equiv Sc_{Tez(B)},\ Ob^A \equiv \Phi_{Ob^B}^{Ob^A}(Ob^A) \equiv Ob^B,\ Mor^A \equiv \Phi_{Mor^B}^{Mor^A}(Mor^A) \equiv Mor^B. \quad (8)$$

Дотримання еквівалентності в умовах (8) дозволяє отримати для платформи *B*, де планується експлуатувати мігроване ПЗ ІС, ті самі рішення, які було отримано та перевірено під час експлуатації похідного ПЗ цієї ж ІС на платформі *A*.

Отриманий результат дозволив описати формування моделі одномісного коваріантного функтора для першого варіанту кросплатформної міграції як метод, що складається з таких етапів.

Етап 1. Визначення аналітиком множин тезаурусів $Sc_{Tez(A)}$ та $Sc_{Tez(B)}$ і перевірка умови $Sc_{Tez(A)} \equiv Sc_{Tez(B)}$. Якщо умова не виконується, то визначення неможливості виконання обраної дії з кросплатформної міграції і завершення використання методу.

Етап 2. Визначення на основі тезаурусу $Sc_{Tez(A)}$ елементів множин Ob^A та Mor^A для похідного варіанту ПЗ.

Етап 3. Визначення на основі тезаурусу $Sc_{Tez(B)}$ елементів множин Ob^B та Mor^B для кінцевого варіанту ПЗ.

Етап 4. Визначення множини правил $\Phi_{Ob^B}^{Ob^A}(Ob^A)$ та $\Phi_{Mor^B}^{Mor^A}(Mor^A)$, для яких виконуються умови $Ob^A \equiv \Phi_{Ob^B}^{Ob^A}(Ob^A) \equiv Ob^B$ та $Mor^A \equiv \Phi_{Mor^B}^{Mor^A}(Mor^A) \equiv Mor^B$ відповідно.

Етап 5. Формування опису одномісного коваріантного функтора (1) з врахуванням результатів виконання Етапів 2-4.

Етап 6. Перевірка результату виконання Етапу 5 на відповідність умовам (2)-(5). Завершення використання методу.

Опис одномісного коваріантного функтора (1), отриманий в результаті використання запропонованого методу, можна багаторазово використовувати для кросплатформної міграції інших фрагментів ПЗ ІС або нових версій даного ПЗ без додаткових витрат на оплату праці фахівця з міграції ПЗ.

Другий варіант на формальному рівні запропоновано представити як сукупність таких одномісних коваріантних функторів:

– коконус функторів G_i, кожен з яких визначає базу правил модифікації ПЗ ІС до особливостей платформи Pl_i;

– ізоморфний функтор, який встановлює взаємну відповідність між визначеною системою правил кросплатформної міграції та описами цих правил для варіанту ПЗ, що є похідним для процесу кросплатформної міграції;

– конус функторів F_i, кожен з яких визначає базу правил міграції похідного варіанту ПЗ ІС у варіант ПЗ ІС, який планується експлуатувати в межах платформи Pl_i.

Цю сукупність функторів запропоновано представити діаграмою, що має вигляд:

$$\begin{array}{ccc} K_{op} & \xleftrightarrow{Iso} & Sw_{op} \\ \uparrow G_i & & \downarrow F_i \\ K_{op}^i & & Sw_{op}^i \end{array}, \quad i = 1,...,n, \tag{9}$$

де K_{op}^i – база правил, яка визначає особливості міграції похідного ПЗ до платформи Pl_i; n – кількість платформ, на яких здійснюється чи планується експлуатація ПЗ ІС; K_{op} – система правил кросплатформної міграції, яка є загальною для усіх підтримуваних платформ, на яких здійснюється чи планується здійснювати експлуатацію ПЗ ІС; Sw_{op} – варіант ПЗ, що є похідним для процесу кросплатформної міграції; Sw_{op}^i – варіант ПЗ, який є результатом процесу кросплатформної міграції для платформи Pl_i.

Роботу з формування баз правил K_{op} та K_{op}^i запропоновано формально описати такою комутативною діаграмою:

$$\begin{array}{ccc} Sc_{Tez}^i & \xrightarrow{O_i} & Sc_{MTez} \\ \downarrow \Phi_{K_{op}^i}^{Sc_{Tez}^i} & & \downarrow \Phi_{K_{op}}^{Sc_{Mtez}} \\ K_{op}^i & \xrightarrow{G_i} & K_{op} \end{array}, \quad i = 1,...,n, \tag{10}$$

де Sc_{Tez}^i – фрагмент тезаурусу, який визначає особливості кросплатформної міграції похідного варіанту ПЗ ІС до платформи Pl_i; Sc_{MTez} – тезаурус, який визначає загальні особливості міграції ПЗ ІС до платформи, яку обрано як похідну базову платформу для експлуатації ІС; O_i – коконус функторів, кожен з яких визначає базу правил модифікації тезаурусу платформи Pl_i до особливостей похідної базової платформи для експлуатації ПЗ ІС; $\Phi_{K_{op}^i}^{Sc_{Tez}^i}$ – функтор, який визначає особливості перетворення фрагменту тезауруса на множину правил кросплатформної міграції похідного варіанту ПЗ ІС до платформи Pl_i; $\Phi_{K_{op}}^{Sc_{MTez}}$ – функтор, який визначає особливості перетворення тезауруса на множину правил міграції ПЗ ІС до платформи, яку обрано як похідну базову платформу для експлуатації ІС.

В цьому варіанті модель одномісного коваріантного функтора (1)-(6) повинна враховувати існування тезаурусу, рівень якого перевищує рівні фрагментів тезаурусів окремих платформ. Такий тезаурус повинен базуватися на фрагменті тезауруса платформи,

яку обрано як похідну базову платформу для експлуатації ІАСМЗ, і розглядатися як результат сукупності бієктивних відображень елементів тезаурусів окремих платформ Pl_i. Тому для опису другого варіанту було запропоновано модифікувати опис одномісного коваріантного функтора (1) таким чином

$$\Phi_B^A = (Sc_{MTez}, Sc_{Tez(A)} \subseteq Sc_{MTez}, Ob^A, Mor^A, Sc_{Tez(B)} \subseteq Sc_{MTez}, Ob^B, Mor^B, \Phi_{Ob^A}^{Ob^B}, \Phi_{Mor^A}^{Mor^B}). \quad (11)$$

Запропонований модифікований опис одномісного коваріантного функтора (11) є узагальненим. У випадках застосування моделі (11) для формального опису окремих конусів та коконусів функторів, визначених на діаграмах (9) та (10), один з елементів ($Sc_{Tez(A)}$ або $Sc_{Tez(B)}$) буде набувати такого вигляду:

$$Sc_{Tez(A)} = Sc_{MTez}; \ \ Sc_{Tez(B)} = Sc_{MTez} \ . \quad (12)$$

Умови (6) для другого варіанту було запропоновано змінити таким чином:

$$Ob^A \subseteq \Phi_{Ob^B}^{Ob^A}(Ob^A) \equiv Ob^B, \ Mor^A \subseteq \Phi_{Mor^B}^{Mor^A}(Mor^A) \equiv Mor^B. \quad (13)$$

Застосування модифікованої таким чином моделі одномісного коваріантного функтора в процесі кросплатформної міграції ПЗ ІС було запропоновано представити як дві послідовно виконувані роботи цього процесу: Робота 1 (підготовча) та Робота 2 (здійснення міграції). Суть Роботи 1 полягає у формуванні баз правил K_{op}^i та системи правил K_{op} на основі результатів аналізу тезаурусів Sc_{Tez}^i та визначення тезауруса Sc_{MTez}. Узагальнений сценарій виконання Роботи 1 було запропоновано представити як метод, що складається з таких етапів.

Етап 1. Визначення аналітиком множини платформ (Pl_i), на яких планується здійснювати експлуатацію ПЗ ІС, та платформи, яку обрано як похідну базову платформу для експлуатації ІС. Формування множини тезаурусів Sc_{Tez}^i і тезауруса Sc_{MTez}.

Етап 2. Для кожного з тезаурусів множини Sc_{Tez}^i визначення на його основі елементів множин Ob^A та Mor^A для моделей функтора коконуса O_i. Для тезауруса Sc_{MTez} визначення на його основі елементів множин Ob^B та Mor^B для моделей функтора коконуса O_i.

Етап 3. Для моделей функтора коконуса O_i визначення множин правил $\Phi_{Ob^B}^{Ob^A}(Ob^A)$ та $\Phi_{Mor^B}^{Mor^A}(Mor^A)$, для яких виконуються умови (13). Формування опису функтора коконуса O_i на основі моделі (11) з врахуванням отриманих результатів. Перевірка сформованого опису на відповідність умовам (2)-(5).

Етап 4. Формування множини моделей функторів $\Phi_{K_{op}^i}^{Sc_{Tez}^i}$ та моделі функтора $\Phi_{K_{op}}^{Sc_{MTez}}$. Синтез баз правил K_{op}^i та системи правил K_{op}.

Етап 5. Синтез моделей функторів коконуса G_i. Завершення використання методу.

Суть Роботи 2 полягає у здійсненні процесів кросплатформної міграції похідного варіанту ПЗ ІС Sw_{op} до варіантів ПЗ ІС Sw_{op}^i, які повинні експлуатуватися в межах

платформ Pl_i, на основі сформованих в результаті виконання Роботи 1 баз правил K_{op}^i та системи правил K_{op}. Узагальнений сценарій виконання Роботи 2 було запропоновано представити як метод, що складається з таких етапів.

Етап 1. Перевірка наявності оновленої версії похідного варіанту ПЗ ІС Sw_{op}, для якого треба виконати процеси кросплатформної міграції. Якщо такої версії немає, то визначення неможливості виконання процесів кросплатформної міграції і завершення використання методу.

Етап 2. Визначення на основі системи правил K_{op} елементів множин Ob^A та Mor^A для похідного варіанту ПЗ.

Етап 3. Визначення на основі баз правил K_{op}^i елементів множин Ob^B та Mor^B для кінцевого варіанту ПЗ.

Етап 4. Визначення множини правил $\Phi_{Ob^B}^{Ob^A}(Ob^A)$ та $\Phi_{Mor^B}^{Mor^A}(Mor^A)$, для яких виконуються умови $Ob^A \subseteq \Phi_{Ob^B}^{Ob^A}(Ob^A) \equiv Ob^B$ та $Mor^A \subseteq \Phi_{Mor^B}^{Mor^A}(Mor^A) \equiv Mor^B$ відповідно.

Етап 5. Формування опису одномісного коваріантного функтора (11) з врахуванням результатів виконання Етапів 2-4.

Етап 6. Перевірка результату виконання Етапу 5 на відповідність умовам (2)-(5). Завершення використання методу.

Опис одномісного коваріантного функтора (11), отриманий в результаті використання запропонованих методів, можна багаторазово використовувати для кросплатформної міграції нових версій ПЗ ІС у варіанти ПЗ цієї ж ІС, які повинні експлуатуватися в межах платформ Pl_i, i=1,2,…, без додаткових витрат на оплату праці фахівця з міграції ПЗ.

Основна відмінність між першим та другим способом здійснення кросплатформної міграції за умови використання запропонованих модифікованих моделей одномісного коваріантного функтора полягає у такому. Перший спосіб дозволяє здійснити процес кросплатформної міграції безпосередньо під час експлуатації поточної чи оновленої похідної версії ПЗ ІС. При цьому під час експлуатації похідної версії ПЗ ІС можуть бути виявлені та виправлені помилки, які не були знайдені під час розробки, реалізації та верифікації цієї версії ПЗ ІС. Такий спосіб є найкращим для ІС, які експлуатуються обмеженою кількістю підприємств-замовників в межах невеликої кількості платформ.

Другий спосіб орієнтовано на здійснення процесу кросплатформної міграції після завершення верифікації ПЗ ІС (до початку або в процесі валідації цього ПЗ). При цьому для кожної платформи з множини платформ (Pl_i), на яких планується експлуатація ПЗ ІС, процес кросплатформної міграції буде здійснюватися автоматично за умови можливості формування баз правил міграції K_{op}^i та загальної системи правил міграції K_{op}. Такий спосіб є найкращим для до ІС, що експлуатуються великою кількістю підприємств-замовників в межах значного різноманіття платформ, в межах яких здійснюється експлуатація цих ІС та їхніх ПЗ.

5.2. Експериментальна перевірка результатів модифікації моделі одномісного коваріантного функтора

Під час експериментальної перевірки результатів модифікації моделі одномісного коваріантного функтора було розглянуто особливості виконання процесу кросплатформної міграції ПЗ ІАСМЗ з платформи Linux до платформи AIX від IBM. Для

цієї міграції першою роботою процесу є робота з кросплатформної міграції результатів рефакторингу вихідного коду. Необхідність цієї роботи обумовлена такими причинами. Для ПЗ ІАСМЗ певна частина програмного коду, що реалізує бізнес-логіку функцій, повинна бути присутня як на клієнтській, так і на серверній частині (наприклад, бізнес-логіка функцій калькуляції). Але решта програмного коду повинна бути присутня лише в одній частині ПЗ – або клієнтській, або серверній. Тому слід виділити в ПЗ ІАСМЗ фрагменти ПЗ, які повинні виконуватися виключно на сервері.

Оскільки ПЗ ІАСМЗ вже протягом значного часу експлуатується на платформі Linux, для здійснення процесу кросплатформної міграції було вирішено використати перший спосіб. Тому виконання роботи з кросплатформної міграції результатів рефакторингу вихідного коду здійснювалося за визначеними етапами відповідного методу.

Під час виконання Етапу 1 було встановлено, що застосування мови програмування C/C++ дало для платформи Linux можливість реалізувати виділення серверної частини ПЗ ІАСМЗ за допомогою директиви препроцесора, наведеної на рис. 1.

```
#if defined __linux__
//source code that related to server side only will be included from
this section in compilation time
#endif
//all source code from this section will be included in compilation
time
```

Рис. 1. Директива препроцесора, яка виділяє серверну частину програмного забезпечення для платформи Linux

Таке рішення забезпечило для платформи Linux виділення в серверну частину лише того коду ПЗ ІАСМЗ, який відповідає за реалізацію операцій, що складають частини сценаріїв виконання функцій ІАС, призначених для виконання в межах серверної частини. Застосування цього рішення на етапі компіляції повинно було зменшити загальний розмір модулів ПЗ, що, в свою чергу, повинно було позитивно вплинути на загальну продуктивність усієї ІАСМЗ в межах платформи Linux.

Але застосувати наведену на рис. 1 директиву «__linux__» в межах платформи AIX неможливо, тому що ця платформа не гарантує присутність запропонованої директиви. Якщо ж здійснити міграцію ПЗ ІАСМЗ до платформи AIX, не змінюючи директиву «__linux__», усі фрагменти ПЗ ІАСМЗ, які були виділені цією директивою, будуть додані до клієнтської частини, а не до серверної. Тому для забезпечення виконання умови $Sc_{Tez(A)} \equiv Sc_{Tez(B)}$ було запропоновано для платформи AIX використати директиву препроцесора, яка розпізнається AIX-компілятором (див. рис. 2).

```
#if defined (_AIX) || defined (__linux__)
//source code that related to server side only will be included from
this section in compilation time
#endif
//all source code from this section will be included in compilation
time
```

Рис. 2. Директива препроцесора, яка виділяє серверну частину програмного забезпечення для платформ Linux та AIX

Під час виконання Етапу 2 було встановлено, що наведена на рис. 1 директива є

єдиним елементом множини Ob^A для похідного варіанту ПЗ ІАСМЗ, який експлуатується на платформі Linux. Оскільки таке виділення не змінює зв'язки між модулями ПЗ, встановлені під час проєктування та розробки ПЗ ІАСМЗ, множина Mor^A була визначена як пуста множина.

Під час виконання Етапу 3 було встановлено, що наведена на рис. 2 директива є єдиним елементом множини Ob^B для похідного варіанту ПЗ ІАСМЗ, який експлуатується на платформі Linux. Оскільки таке виділення не змінює зв'язки між модулями ПЗ, встановлені під час проєктування та розробки ПЗ ІАСМЗ, множину Mor^B було визначено як пусту множину.

Під час виконання Етапу 4 було визначено, що множину правил $\Phi_{Ob^B}^{Ob^A}(Ob^A)$ повинно складати одне правило. Загальний вигляд функції «directive_server», яка реалізує правило кросплатформної міграції з платформи Linux до платформи AIX директиви препроцесора з виділення серверної частини ПЗ наведено на рис. 3.

```
string directive_server (directive_x)
{
 if (directive_x == "#if defined __linux__")
 {
     return "#if defined (_AIX) || defined (__linux__)"
 }
}
```

Рис. 3. Загальний вигляд функції «directive_server»

Використання цього правила дозволяє стверджувати, що умову $Ob^A \equiv \Phi_{Ob^B}^{Ob^A}(Ob^A) \equiv Ob^B$ для даного одномісного коваріантного функтора виконано (наведене на рис. 3 правило $\Phi_{Ob^B}^{Ob^A}(Ob^A)$ містить частини, еквівалентні як Ob^A, так і Ob^B).

Множина правил $\Phi_{Mor^B}^{Mor^A}(Mor^A)$, виходячи з того, що $Mor^A = \varnothing$ та $Mor^B = \varnothing$, теж була визначена як пуста множина. Тому умову $Mor^A \equiv \Phi_{Mor^B}^{Mor^A}(Mor^A) \equiv Mor^B$ слід вважати виконаною.

В результаті виконання Етапу 5 було отримано екземпляр моделі одномісного коваріантного функтора, який має вигляд:

$$\begin{aligned}\Phi_{AIX}^{Linux} = (&"\_\_linux\_\_","\#if\ defined\ \_\_linux\_\_",\varnothing,\\ &\_AIX,"\#if\ defined(\_AIX)||defined(\_\_linux\_\_)",\varnothing,.\\ &"string\ directive\_server(directive\_x)",\varnothing)\end{aligned} \quad (14)$$

В результаті виконання Етапу 6 було встановлено, що отримана модель (14) відповідає умовам (2)-(5).

Сформована модель одномісного коваріантного функтора (14), яка описує роботу з кросплатформної міграції результатів рефакторингу вихідного коду ПЗ ІАСМЗ, дозволяє автоматизувати виконання цієї роботи для усіх подальших версій цього ПЗ. Таке рішення забезпечить коректне лінкування з API платформи, і дозволить звести до мінімуму витрати

людино-годин на міграцію ПЗ ІАСМЗ після внесення змін до похідного варіанту цього ПЗ.

6. Обговорення результатів дослідження

Основними результатами дослідження є розроблені дві модифікації моделі одномісного коваріантного функтора. Перша з цих модифікацій дозволяє описати роботи процесу кросплатформної міграції як конуси одномісних коваріантних функторів, кожен з яких встановлює правило виконання цієї роботи для похідної та кінцевої платформ. Суть цього варіанту модифікації полягає в уточненні умови (6) існування одномісного коваріантного функтора, яка, на відміну від базового варіанту, встановлює наявність еквівалентних описів елементів множин об'єктів та морфізмів похідної та кінцевої категорій та множин відповідних правил функтора. Друга з цих модифікацій дозволяє описати роботи процесу кросплатформної міграції як конуси одномісних коваріантних функторів, кожен з яких визначає базу правил міграції похідного варіанту ПЗ ІС у варіант ПЗ ІС, який планується експлуатувати в межах платформи Pl_i . Але, на відміну від першого варіанту, конус функторів у другому варіанті є результатом формування коконуса функторів, кожен з яких визначає базу правил модифікації тезаурусу платформи Pl_i до особливостей похідної базової платформи для експлуатації ПЗ ІС, та коконуса функторів, кожен з яких визначає базу правил модифікації ПЗ ІС до особливостей платформи Pl_i . При цьому отримані результати формування цих коконусів узгоджуються між собою за рахунок існування ізоморфних одномісних коваріантних функторів, які встановлюють взаємно-однозначне відображення цих результатів. Суть цього варіанту модифікації полягає в уточненні не тільки умови (6), а й базового формального опису одномісного коваріантного функтора (1).

Запропоновані модифікації моделі одномісного коваріантного функтора дозволяють, на відміну від базової моделі цього функтора (1)-(6), краще пристосувати цю модель до особливостей процесу кросплатформної міграції ПЗ ІС різного призначення. Використання модифікованих моделей функтора для формального опису робіт та діяльностей процесу кросплатформної міграції ПЗ ІС дозволяють встановити основні вимоги до інформаційних технологій, що дозволять автоматизувати виконання цього процесу. Така автоматизація дозволить скоротити витрати часу та ресурсів на виконання процесу кросплатформної міграції ПЗ ІС під час впровадження та експлуатації даного ПЗ на платформах підприємств-замовників цих ІС.

Основні характеристики запропонованих модифікацій моделі одномісного коваріантного функтора та їх значення наведено у табл. 1.

На відміну від наведених у [18] рішень, запропоновані модифіковані моделі одномісного коваріантного функтора орієнтовано на опис процесу кросплатформної міграції ПЗ саме ІС як окремого класу ІТ-продуктів. На відміну від рішень, розроблених у [19], розроблені моделі спрямовано на опис робіт та діяльностей процесів валідації, функціонування та супроводження ІС.

Головним обмеженням застосування отриманих результатів даного дослідження є спрямованість модифікованих моделей одномісного коваріантного функтора саме на опис процесу кросплатформної міграції ПЗ ІС. Тому застосування отриманих моделей для формального опису інших процесів життєвого циклу ІС вимагає проведення додаткових досліджень з модифікації або уточнення отриманих моделей до особливостей цих процесів. Ще одним досить значним недоліком отриманих результатів є необхідність додаткових витрат зусиль та часу аналітика на встановлення фактів еквівалентності елементів тезаурусів платформ, між якими повинна відбуватися міграція ПЗ ІС. Намагання ліквідувати цей недолік приводить до необхідності вирішення задачі оцінювання

Таблиця 1

Порівняльні характеристики запропонованих модифікацій моделі одномісного коваріантного функтора

Найменування характеристики	Перший варіант модифікації	Другий варіант модифікації
Формальне представлення	Конус одномісних коваріантних функторів, кожен з яких описує набір правил кросплатформної міграції ПЗ ІС з платформи A до платформи B	– коконус функторів, кожен з яких визначає базу правил модифікації тезаурусу платформи Pl_i до особливостей похідної базової платформи для експлуатації ПЗ ІС; – коконус функторів, кожен з яких визначає базу правил модифікації ПЗ ІС до особливостей платформи Pl_i ; – ізоморфний функтор, який встановлює взаємну відповідність між визначеною системою правил кросплатформної міграції та описами цих правил для варіанту ПЗ, що є похідним для процесу кросплатформної міграції; – конус функторів, кожен з яких визначає базу правил міграції похідного варіанту ПЗ ІС у варіант ПЗ ІС, який планується експлуатувати в межах платформи Pl_i .
Концепція міграції	Концепція «точка – точка» (міграція відбувається між двома апріорно обраними платформами)	Концепція «загальносистемна шина» (міграція відбувається між базовою платформою та множиною платформ (Pl_i), які апріорно обрані для експлуатації ПЗ ІС)
Необхідність централізованої системи / бази правил міграції	Ні	Так
Обробка тезаурусів платформ	Здійснюється аналітиком вручну	Може бути автоматизована
Технологічний рівень реалізації	Може бути реалізована вручну або шляхом створення інформаційної технології	Потребує спеціальної інформаційної технології
Потреба в персоналі	Потрібен один фахівець, добре знайомий з тезаурусами обох платформ	Потрібна команда фахівців, знайомих з тезаурусами усіх запропонованих для міграції платформ Pl_i
Потреба в часі для налаштування моделі	Може бути налаштована за один робочий день	Потребує виділення для налаштування проміжку часу, не більше однієї ітерації ІТ-проєкту

Кінець таблиці 1

Найменування характеристики	Перший варіант модифікації	Другий варіант модифікації
Можливість повторного використання	Тільки для апріорно обраної пари платформ	Тільки для апріорно обраної множини платформ (Pl_i)
Стадія та процеси життєвого циклу ІС, в межах якої слід застосовувати модифікацію	Стадія експлуатації ІС (процеси функціонування та супроводження ІС)	Стадія впровадження ІС (процеси інтеграції та валідації ІС)
Відповідальні за застосування модифікації	Фахівці з супроводження ПЗ ІС	Фахівці з інтеграції та/або валідації ПЗ ІС на платформах підприємств-замовників ІС
Кількість впроваджень ІС, для яких доцільне застосування модифікації	Невелика (до 10-15 впроваджень)	Практично необмежена

ефективності застосування інформаційних технологій автоматизованої кросплатформної міграції ПЗ ІС, які базуються на категорно-функторному апараті, і проведення значної кількості подальших досліджень в даному напрямі.

Ці та інші обмеження і недоліки отриманих результатів з модифікації моделі одномісного коваріантного функтора визначили такі напрями подальших досліджень в цій галузі:

– дослідження технологічних і прикладних рішень з реалізації модифікованих моделей одномісного коваріантного функтора;

– дослідження варіантів модифікації моделі одномісного коваріантного функтора для формального опису інших процесів життєвого циклу ІС;

– дослідження з формального категорно-функторного опису та розробки архітектури автоматизованої інтелектуальної системи управління життєвим циклом ІС різного призначення.

7. Висновки

У ході даного дослідження було вирішено задачу модифікації моделі функтора для опису трансформації ПЗ ІС в межах її кросплатформної міграції. Під час вирішення цієї задачі було здійснено:

– модифікацію моделі одномісного коваріантного функтора (7) шляхом уточнення умов (8) існування такого функтора в процесі кросплатформної міграції ПЗ ІС під час експлуатації цієї ІС;

– модифікацію моделі одномісного коваріантного функтора (11) шляхом уточнення опису цього функтора з врахуванням особливостей процесу кросплатформної міграції ПЗ ІС під час валідації цієї ІС;

– експериментальну перевірку модифікованих моделей одномісного коваріантного функтора під час здійснення процесу кросплатформної міграції ПЗ ІАСМЗ і, зокрема, роботи з кросплатформної міграції результатів рефакторингу вихідного коду даного ПЗ.

Отримані результати підтверджують можливість автоматизованого виконання процесу кросплатформної міграції ПЗ ІС шляхом створення і використання відповідних інформаційних технологій.

Основним напрямом подальших досліджень запропоновано вважати дослідження особливостей технологічних і прикладних рішень з реалізації модифікованих моделей одномісного коваріантного функтора та оцінювання ефективності застосування подібних рішень під час функціонування та супроводження ІС різного призначення.

Перелік посилань:

1. Awasthi, Y., Zengeni, T. M., & Makambwa, T. A Framework for Cloud Migration in Academic Institutions. ResearchGate. 2024. URL: https://www.researchgate.net/publication/381952876_A_Framework_for_Cloud_Migration_in_Academic_Institutions

2. Emmanuel Cadet, Olajide Soji Osundare, Harrison Oke Ekpobimi, Zein Samira & Yodit Wondaferew Weldegeorgise. Cloud migration and microservices optimization framework for large-scale enterprises. Open Access *Research Journal of Engineering and Technology.* 2024. Vol. 7(2). P. 046–059. URL: https://doi.org/10.53022/oarjet.2024.7.2.0059

3. Gethers, M., Dit, B., Kagdi, H., Poshyvanyk, D. Integrated impact analysis for managing software changes. *34th International Conference on Software Engineering (ICSE)*. Zurich, Switzerland. 2012. P. 430-440. URL: doi: https://doi.org/10.1109/ICSE.2012.6227172

4. Manapian, A., Prompoon, N. Software time estimation model for requirements change based on software prototype profiles using an analogy estimation method. *2014 International Computer Science and Engineering Conference (ICSEC)*. Khon Kaen. Thailand. 2014. P. 366-371. URL: doi: https://doi.org/10.1109/ICSEC.2014.6978224

5. Miguel, M.A., Araújo, M.A.P., David, J.M.N., Braga, R. A framework to support effort estimation on software maintenance and evolution activities. *12th Brazilian Symposium on Information Systems: Information Systems in the Cloud Computing Era*, Proceedings. Florianopolis - Santa Catarina. 2016. P. 232-239.

6. Rostami, K., Stammel, J., Heinrich, R., Reussner, R. Change Impact Analysis by Architecture-based Assessment and Planning. *Lecture Notes in Informatics*, Proceedings - Series of the Gesellschaft fur Informatik. Hannover. 2017. Vol. P267, P. 69-70.

7. Rostami, K., Heinrich, R., Busch, A., Reussner, R. Architecture-Based Change Impact Analysis in Information Systems and Business Processes. *2017 IEEE International Conference on Software Architecture (ICSA)*. Gothenburg, Sweden. 2017. P. 179-188. URL: https://doi.org/10.1109/ICSA.2017.17.

8. Dugar, M. How Machine Learning Can Help Developers. *24th International Arab Conference on Information Technology*, ACIT 2023. Ajman. 2023. Code 198200. URL: https://doi.org/10.1109/ACIT58888.2023.10453880

9. Steingartner, W., Novitzká, V., Bačíková, M., Korečko, Š. New approach to categorical semantics for procedural languages. *Computing and Informatics*, 2017. Vol. 36, Iss. 6. P. 1385-1414. URL: https://doi.org/10.4149/cai_2017_6_1385.

10. Crăciunean, D.-C. Categorical modeling method, proof of concept for the petri net language. *Proceedings of the 7th International Conference on Model-Driven Engineering and Software Development*, 2019. P. 283-291. URL: https://doi.org/10.5220/0007360602830291

11. Beohar, H., König, B., Küpper, S., Silva, A., Wissmann, T. A coalgebraic treatment of conditional transition systems with upgrades. *Logical Methods in Computer Science*, 2018. Vol. 14, Iss. 128, 19. URL: https://doi.org/10.23638/LMCS-14(1:19)2018.

12. Hou, J., Zhang, Y., Rana, A.D. Describing Approach for Model-Driven Collaborative Application Development. *International Conference on Artificial Intelligence and Advanced Manufacturing*, AIAM 2019, Proceedings. 2019. P. 336-343. URL: https://doi.org/10.1109/AIAM48774.2019.00073.

13. Hou, J., Xu, C., Zhang, Y. Architecture-Based Semantic Description Framework for Model Transformation. *5th International Conference on Natural Language Processing and Information Retrieval*, NLPIR 2021, ACM International Conference Proceeding Series, 2021. P. 73-80. URL: https://doi.org/10.1145/3508230.3508241

14. Steingartner, W. Perspectives of semantic modeling in categories. *Journal of King Saud University - Computer and Information Sciences.* 2025. Vol. 37. Iss. 3. № 19. URL: https://doi.org/10.1007/s44443-025-00010-9.

15. Chowdhury, S., Clause, N., Mémoli, F., Sánchez, J.Á., Wellner, Z. New families of stable simplicial filtration functors. *Topology and its Application.* 2020. Vol. 279. 107254. URL: https://doi.org/10.1016/j.topol.2020.107254.

16. Lochbihler, A., Marić, O. Authenticated Data Structures as Functors in Isabelle/HOL. *2nd Workshop on Formal Methods for Blockchains*, FMBC 2020, Open Access Series in Informatics, 2020, Vol. 84, 6. URL: https://doi.org/10.4230/OASIcs.FMBC.2020.6.

17. Sun Y.Z., Wang S.T. CNOK: A C++ Glauber model code for single-nucleon knockout reactions. *Computer Physics Communications*, 2023, Vol. 288, № 108726. URL: https://doi.org/10.1016/j.cpc.2023.108726.

18. Bezditnyi, V., Chebanyuk, O. Software Engineering Fundamentals to Design Application for Modern Game Engines. *2024 14th International Scientific and Practical Programming Conference*, UkrPROG 2024, CEUR Workshop Proceedings, Vol. 3806, P. 89–99.

19. Левикін В.М., Євланов М.В., Керносов М.А. Патерни проєктування вимог до інформаційної системи:

моделювання та застосування: монографія. (рос.) Харків: ТОВ «Компанія СМІТ», 2014. 320 с.

20. Левикін В.М., Євланов М.В. Задача визначення функторів між категорними моделями інформаційної системи. (рос.) *Проблеми біоніки*. 2003. Вип. 58. С. 62-67.

21. Євланов М.В. Формалізація взаємних відображень моделей інформаційних систем. (рос.) *Materialy IV Miedzynarodowej naukowi-praktycznej konferencji Nowoczesnych naukowych osiagniec - 2008*. Т. 13. Matematyka. Fizyka. Nowoczesne informacyjne technologie. Przemysl: Nauka i studia. P. 82-85.

22. Awodey S. Category Theory. 2nd ed. NY.: OXFORD UNIVERSITY PRESS, 2010. XVI+311 p.

Надійшла до редколегії 12.06.2025 р.

Круглик Андрій Сергійович, аспірант кафедри програмного забезпечення комп'ютерних систем, факультет інформаційних технологій, НТУ «Дніпровська Політехніка», м. Дніпро, Україна, e-mail: swatkrim@gmail.com.

Левикін Віктор Макарович, доктор технічних наук, професор, професор кафедри ІУС ХНУРЕ, м. Харків, Україна, e-mail: viktor.levykin@nure.ua, ORCID: https://orcid.org/0000-0002-7929-515X

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

Мороз Борис Іванович, доктор технічних наук, професор, професор кафедри програмного забезпечення комп'ютерних систем НТУ «Дніпровська політехніка», м. Дніпро, Україна, e-mail: moroz.b.i@nmu.one, ORCID: https://orcid.org/0000-0002-5625-0864

Мороз Дмитро Максимович, доктор філософії, доцент кафедри програмного забезпечення комп'ютерних систем НТУ «Дніпровська політехніка», м. Дніпро, Україна, e-mail: moroz.d.m@nmu.one, ORCID: https://orcid.org/0000-0003-2577-3352

УДК 004.932.2 DOI: 10.30837/0135-1710.2025.185.023

П.Е. СИТНІКОВА, М.О. ГРИЦАЙ

МОДЕЛЮВАННЯ ТА ДОСЛІДЖЕННЯ МЕТОДІВ ДЕТЕКЦІЇ ПОГЛЯДУ КОРИСТУВАЧА У СИСТЕМАХ ЛЮДИНО-КОМП'ЮТЕРНОЇ ВЗАЄМОДІЇ

Реалізовано систему зорового введення, що дозволяє керувати курсором на екрані за допомогою рухів очей користувача. Рішення базується на використанні нейронної мережі для прогнозування напрямку погляду та враховує просторову орієнтацію голови, отриману за допомогою оцінки пози. Для підвищення стабільності результатів застосовано методи фільтрації шуму та згладжування координат. Програмний модуль виконує нормалізацію зображення обличчя, трансформує дані у тривимірному просторі та обчислює точку перетину вектора погляду з площиною екрана, що дає змогу точно відображати фокус уваги користувача у вигляді візуального маркера. Результати підтвердили працездатність системи в режимі реального часу.

1. Вступ

Сучасні технології взаємодії людини з комп'ютером стрімко розвиваються, наслідком чого є зростання інтуїтивності та зручності процесу керування пристроями. Одним із перспективних напрямів є зорове введення, що дозволяє користувачам керувати системами за допомогою рухів очей, зменшуючи необхідність фізичних маніпуляцій. Це особливо важливо для людей з обмеженими можливостями, а також у середовищах, де традиційні методи введення можуть бути незручними або неефективними.

Основною проблемою при розробці таких систем є точність визначення погляду, обробка великого обсягу даних у реальному часі та інтеграція цих технологій у існуючі програмні та апаратні комплекси. Сучасні методи комп'ютерного зору та машинного