

DOI: <https://doi.org/10.15276/ict.03.2026.45>

УДК 004.942:004.415

Метод ієрархічного узагальнення процесної моделі програмного продукту

Галушко Дмитро Олександрович

ORCID: <https://orcid.org/0000-0002-1980-7387>; halushko.dmytro@edu.kpi.ua

Київський політехнічний інститут імені Ігоря Сікорського, Берестейський просп., 37
Київ, 03056, Україна

АНОТАЦІЯ

У роботі запропоновано метод ієрархічного узагальнення процесної моделі програмного продукту, сформованої на основі документальних артефактів. Вихідну модель отримано з наскрізної специфікації процесу, двох операційних регламентів, матриці ролей і систем та сценаріїв приймальної перевірки. Документи сегментовано на логічні фрагменти, з яких виділено кандидатів на дії, умови, ролі, системи та переходи; до графа включено лише елементи, підтверджені або відредаговані бізнес-аналітиком. Модель формалізовано як типізований орієнтований граф із посиланнями на джерела. Для побудови рівнів деталізації кандидатні групи формуються в локальному графовому оточенні та перевіряються на цілісність альтернативних гілок. Структурно допустимі фрагменти оцінюються інтегральним показником, що поєднує текстову, контекстну й структурну складові. Під час узагальнення зберігаються керувальний потік, цілісність розгалужень і простежуваність агрегованих елементів до вихідної документації. Запропонований підхід поєднує семантичні характеристики процесних елементів, контекст їх виконання та структуру керувального потоку, що дає змогу формувати узгоджені рівні подання без порушення логіки альтернативних гілок. Відтворюваність побудови забезпечується фіксованими правилами формування кандидатних груп, збереженням зв'язків із вихідними документами та послідовним розкриттям узагальнених елементів до атомарного рівня. Контрольну перевірку проведено на тестовому графі. Після узагальнення кількість видимих вузлів та переходів скоротилася. Кандидат із достатнім числовим показником було відхилено через неповне охоплення альтернативної гілки. Метод формує компактне багаторівневе подання без втрати зв'язку з атомарними елементами та документацією, на підставі якої їх створено. Запропонований підхід може застосовуватися для багаторівневого подання процесних моделей програмних продуктів.

Ключові слова: процесна модель; програмний продукт; документація; ієрархічне узагальнення; простежуваність; орієнтований граф; процесна абстракція; керувальний потік

Для цитування: Галушко Д. О. “Метод ієрархічного узагальнення процесної моделі програмного продукту”. *Інформатика. Культура. Техніка*. 2026; Vol 3 № 2 (4): 599–612. DOI: <https://doi.org/10.15276/ict.03.2026.45>

Актуальність. Документація програмного продукту містить описи вимог, функціональних сценаріїв, бізнес-правил, ролей, систем та умов виконання операцій. Після вилучення цих відомостей і підтвердження аналітиком вони можуть бути подані як процесна модель. Однак зі зростанням продукту така модель швидко ускладнюється: збільшується кількість вузлів і переходів, а функціонально завершені етапи стають менш помітними.

Механічне вилучення деталей не розв'язує проблему, оскільки може приховати умови, альтернативні гілки та джерело походження елементів. Тому потрібне багаторівневе представлення, у якому користувач переходить від макроетапів до атомарних операцій, а кожен узагальнений блок зберігає зв'язок із первинним графом і документаційними фрагментами.

Дослідження процесної абстракції охоплюють трансформацію між рівнями деталізації, ієрархізацію подій і групування вузлів графа [1], [2], [3], [4], [5]. Семантичну близькість назв активностей оцінюють методами обробки природної мови та векторними представленнями [6], [7], проте така близькість сама по собі не відображає положення операцій у керувальному потоці. Окремо досліджуються зрозумілість, модульність і когнітивне навантаження процесних моделей [8], [9], [10], [11], [12], [13], [14], а також автоматизований контроль структурних помилок Business Process Model and Notation (BPMN) [15].

Водночас недостатньо формалізовано узагальнити процесну модель, із одночасним урахуванням семантики елементів, їх контексту, топології графа та простежуваності до джерел.

Запропонований метод спрямовано на усунення цієї прогалини: вихідний граф формується з підтверджених елементів документації, а його подальше узагальнення виконується без зміни первинної процесної логіки.

У роботі розмежовано два пов'язані етапи. Спочатку з документації формується перевірений граф G^0 , а потім для нього будується ієрархія подань. Таке розмежування потрібне, оскільки узагальнення не може виправити пропущений перехід, хибну умову або неправильно визначеного виконавця. Якість вищих рівнів безпосередньо залежить від повноти й структурної коректності вихідної моделі.

Метою дослідження є розроблення методу ієрархічного узагальнення процесної моделі програмного продукту, який формує взаємоузгоджені рівні деталізації, зберігає структурно значущі розгалуження та простежуваність до вихідної документації.

Для досягнення мети поставлено такі завдання:

- визначити вимоги до вихідного графа, сформованого з документації, та формалізувати зв'язок між рівнями його подання;
- розробити критерій оцінювання кандидатних фрагментів на основі текстової, контекстної та структурної складових;
- виконати контрольну перевірку методу та кількісно оцінити зміну структури подання на різних рівнях узагальнення.

Метод працює з процесною моделлю, відтвореною з документації, а не з журналом подій. Документи можуть мати різну структуру й рівень деталізації, тому автоматизований аналіз використовується для підготовки кандидатів, а не для безумовного наповнення графа. Бізнес-аналітик перевіряє зміст кандидата, його місце в керувальному потоці та посилання на джерело. Лише після цього елемент набуває статусу підтвердженого або зміненого й може входити до G^0 .

Множину документальних артефактів $D = \{d_1, d_2, \dots, d_n\}$ поділяють на логічні фрагменти: абзаци, пункти, рядки таблиць, кроки сценаріїв і окремі правила. Межі фрагмента мають бути достатніми для відновлення змісту пропозиції та водночас достатньо точними для посилання на джерело. Для кожного фрагмента зберігаються код документа, локальний ідентифікатор, тип блока та текстове представлення. Так формується основа для простежуваності майбутніх вузлів і переходів – набір карток.

Після сегментації виконується лінгвістична обробка, застосовуються правила для конструкцій умови й послідовності, а терміни зіставляються з доменним словником. На цій основі утворюються кандидати на дії, умови, ролі, бізнес-об'єкти, зовнішні системи та зв'язки. Кандидати з різних документів порівнюються за нормалізованою назвою, типом і контекстом. Збіг не спричиняє автоматичного об'єднання: система лише готує картку пропозиції з фрагментом-джерелом, способом вилучення та попередньою оцінкою впевненості.

Для картки бізнес-аналітик обирає одне з практичних рішень: підтвердити, відредагувати, пов'язати з наявним елементом, відхилити або повернути на уточнення. До процесної моделі потрапляють лише підтверджені й відредаговані елементи. Якщо відповідний вузол уже існує, до нього додається нове джерело замість створення дублікату. Відхилені гіпотези зберігаються в історії перевірки, але не впливають на топологію G^0 .

Структурні межі документа використовуються як початкова підказка: заголовок задає контекст, абзац містить завершене твердження, пункт списку або рядок таблиці - окрему вимогу чи операцію. Під час нормалізації усуваються повторні пробіли та суто оформлювальні відмінності, але зберігаються заперечення, модальні слова, технічні ідентифікатори й посилання між розділами. Довгий фрагмент поділяється лише тоді, коли умова, дія та результат залишаються пов'язаними через спільний ідентифікатор. Надмірне дроблення є небезпечним: відокремлена від наслідку умова може перетворитися на самостійний вузол без коректного вихідного переходу.

Лінгвістична анотація визначає предикат, виконавця, об'єкт і залежні частини висловлювання. Правила окремо обробляють активні та пасивні конструкції, умови, часові маркери й послідовність. Доменний словник узгоджує варіанти назв ролей, систем і процесних об'єктів, наприклад IAM, зовнішній IdP або картку інциденту. Кандидат зберігає не лише нормалізовану назву, а й текстовий доказ та правило, яке його сформувало. Якщо у

фрагменті немає явного виконавця або дія сформульована нечітко, кандидат не відкидається автоматично, але отримує нижчу впевненість і ознаку потреби в перевірці.

Напрямок зв'язку визначається явними маркерами «після», «до», «якщо», «у разі» та структурою сценарію. Для табличного регламенту джерелом можуть бути окремі колонки початкового вузла, умови й наступної дії. Сусідство двох речень саме по собі не вважається достатнім доказом послідовності. У контрольному наборі [16] `article_control_18_nodes` основні переходи взято з D1, а D2 і D3 уточнюють скорочені маршрути всередині альтернативних гілок. Якщо різні документи задають несумісний напрям або різні умови того самого переходу, система створює картку конфлікту замість довільного вибору. Після сегментації аналітик перевіряв гіпотези щодо дій, умов, ролей, систем і переходів: усунув дублікати й непроцесні фрагменти, уточнив неоднозначні зв'язки та підтвердив скорочені маршрути. Тому вхідний граф G^0 містив 18 узгоджених вузлів і 21 перехід, а не всі автоматично виділені картки. Ієрархічне узагальнення вже перевіреного графа виконувалося автоматично.

Кандидати порівнюються за типом, нормалізованою назвою, роллю, системою, об'єктом і положенням у потоці. Семантична подібність використовується як підказка, а не як остаточне рішення. Після підтвердження збігу наявний елемент одержує додаткове посилання на джерело та рішення аналітика. Якщо схожі назви стосуються різних операцій, рішення про відмінність зберігається й у наступних ітераціях перешкоджає повторній пропозиції того самого об'єднання. Завдяки цьому модель не накопичує дублікати, але й не втрачає різні за змістом дії через поверхневу схожість формулювань.

Таблиця 1. Документаційні артефакти для побудови вихідної моделі

Код	Артефакт	Відомості для моделі	Використання
D1	Наскрізна специфікація	Вузли v1-v18, основні переходи й умови	Топологія G^0 та межі процесу
D2	Регламент ідентифікації	Операції v6-v9 і скорочений перехід v7→v9	Перша діагностична гілка
D3	Регламент платформи	Операції v10-v13 і скорочений перехід v11→v13	Друга діагностична гілка
D4	Матриця ролей і систем	Роль, система та процесний об'єкт кожного вузла	Атрибути вузлів і S_{ctx}
D5	Сценарії приймальної перевірки	Маршрути T1-T5 та очікувані результати	Контроль переходів і розгалужень

Джерело: розроблено автором на основі відкритого набору даних [16]

D1 задає наскрізну послідовність, D2 і D3 деталізують альтернативи, D4 уточнює контекст, а D5 перевіряє допустимі маршрути. До графа не потрібно переносити весь текст регламентів. Передумови, аудиторські поля, технічні докази та пояснення залишаються атрибутами або посиланнями на джерело, якщо вони не змінюють керувальний потік. Це зменшує кількість зайвих вузлів ще до застосування ієрархічного узагальнення.

Ієрархічне узагальнення застосовується після формування структурно узгодженого графа і не змінює правил видалення елементів із документації. Воно створює взаємоузгоджені стани подання тієї самої моделі: початковий рівень містить атомарні елементи, перший рівень – функціональні блоки, а наступні – ширші макроетапи. Узагальнений вузол не замінює первинні дані, а посилається на відповідний підграф і його документаційні джерела.

Перед побудовою ієрархії перевіряються унікальність ідентифікаторів, наявність кінцевих вузлів для кожного переходу, умови виходу з вузлів розгалуження, точки злиття альтернатив і джерела походження елементів. Маршрути приймальної перевірки мають виконуватися без змішування частин різних гілок. Якщо ці вимоги порушено, модель

повертається на етап уточнення документації; ієрархічне узагальнення не використовується як спосіб приховати структурну помилку.

Кожен вузол, крім початкового, повинен мати принаймні один допустимий вхід, а кожен незавершальний вузол – вихід, що відповідає описаній умові або послідовності. Для вузлів вибору перевіряються всі вихідні альтернативи та подальше злиття. Окремо контролюється, що сценарії D5 досягають завершення й не використовують переходів, відсутніх у переліку ребер. Ця перевірка не доводить повноту документації в широкому розумінні, але виявляє висячі елементи, незавершені гілки та суперечності між табличним описом і графом.

Якщо D1 задає один порядок операцій, а регламент гілки – інший, жоден варіант не потрапляє до підтвердженої моделі без рішення аналітика. Суперечність прив'язується до обох фрагментів, після чого можна уточнити умову, змінити напрям переходу або зафіксувати окремі варіанти для різних контекстів. Аналогічно обробляється відсутня роль, невідома система або неоднозначна точка злиття. Невизначеність залишається локальною в картці пропозиції й не поширюється на ієрархію, доки відповідний елемент не одержить перевірений статус.

Вихідну процесну модель подано як типізований орієнтований граф:

$$G^0 = (V, E, \tau, A), \quad (1)$$

де V – множина вузлів; E – множина орієнтованих переходів; $\tau(v)$ – тип вузла; $A(v)$ – множина його атрибутів.

Типами вузлів є дії та умови. Ролі, бізнес-об'єкти, зовнішні системи, назви й посилання на документаційні фрагменти зберігаються як атрибути. Ролі й системи не визначають межі групи самостійно, оскільки можуть бути спільними для різних етапів процесу.

У контрольній конфігурації множина V містить 16 дій і два вузли умов v_5 та v_{15} . Для вузла зберігаються нормалізована назва, роль, інформаційна система, бізнес-об'єкт і код документа-джерела. Ребро містить початковий і кінцевий вузли, умову переходу та посилання на документ. Операційні перевірки з D2 і D3 залишаються атрибутами відповідної дії, якщо не створюють окремої точки вибору або нового порядку виконання.

Послідовність графів $G^0, G^1, \dots, G^k$ відповідає рівням $L_0, L_1, \dots, L_k$, а G^0 містить атомарні елементи підтвердженої моделі; G^1 – функціонально цілісні блоки; наступні графи – макроетапи. Кількість рівнів наперед не задається: побудова завершується, коли немає структурно допустимих груп, що досягають порога узгодженості та зменшують кількість видимих вузлів.

Зв'язок сусідніх рівнів визначає відображення $\mu_k : V_{k-1} \rightarrow V_k$. Кожен вузол попереднього рівня належить рівно одному агрегату. Якщо вузол не входить до прийнятої групи, він переноситься як одноелементний агрегат. Отже, прообрази вузлів V_k утворюють неперетинне розбиття V_{k-1} .

Композиція відображень $M_k = \mu_k \circ \dots \circ \mu_0$ забезпечує простежуваність до атомарних вузлів G^0 . Оскільки кожен атомарний вузол містить посилання на фрагмент-джерело, простежуваність поширюється й на документацію. Зміна рівня деталізації тому змінює лише одиницю подання, а не процесну логіку.

Для кожного агрегату зберігають атомарний склад і об'єднання його джерел. Тому зміна документаційного фрагмента дає змогу через зв'язки простежуваності визначити залежні вузли та локально переобчислити ієрархію.

Під час переходу між рівнями виконуються чотири інваріанти: агрегати покривають усі вузли попереднього рівня; їхні прообрази не перетинаються; кожне міжгрупове ребро має принаймні один атомарний прообраз; композиція відображень повертає макровузол до точного набору елементів G^0 і джерел. Це запобігає втраті та дублюванню вузлів і появі нових переходів.

Рівні $L_0, L_1, \dots, L_k$, є узгодженими поданнями однієї моделі, а не результатами повторного аналізу тексту. Вищі рівні не додають процесних фактів, а лише задають межі підграфів і короткі назви. Побудова припиняється, коли наступне узагальнення неможливе.

Ребро між агрегатами створюється, якщо між їхніми прообразами є перехід попереднього рівня. Умови первинних переходів зберігаються в атрибутах агрегованого ребра, а внутрішні ребра залишаються доступними після розкриття до G^0 .

Алгоритм побудови чергового рівня послідовно виконує п'ять операцій: генерує зв'язні кандидатні фрагменти; відхиляє групи, що порушують цілісність розгалуження; обчислює компоненти S_{txt} – текстову узгодженість, S_{ctx} – контекстну однорідність та S_{flow} – структурну узгодженість; ранжує допустимі кандидати та приймає неперетинні групи; формує нові вузли й переходи, а непогруповані вузли переносять як одноелементні агрегати. Після цього процедура повторюється для отриманого графа.

Повний перебір підмножин V є надлишковим і створює багато фрагментів, що не відповідають зв'язним ділянкам процесу. Тому пошук обмежується локальним графовим оточенням: кандидат утворюють суміжні вузли або вузли одного зв'язного фрагмента з малою графовою відстанню. На рівні L_0 кандидат складається з атомарних вузлів. На наступних рівнях кандидат формують із агрегатів попереднього рівня, але S_{txt} , S_{ctx} і S_{flow} обчислюють за множиною їх атомарних прообразів у G^0 . Завдяки цьому значення показників на різних рівнях мають однаковий зміст. Семантична подібність використовується для оцінювання, але не може сама створити групу з віддалених ділянок графа.

До числового оцінювання кандидат перевіряється на цілісність розгалуження. Зв'язний фрагмент усередині однієї альтернативної гілки можна узагальнювати самостійно, якщо він не охоплює частини іншої гілки. Кандидат, що поєднує кілька альтернатив, є допустимим лише тоді, коли кожен залучену гілку включено повністю, а вузли розгалуження та злиття або обидва залишаються за межами кандидата, або обидва входять до нього. Часткове включення альтернативи чи включення лише одного з двох шлюзів спричиняє відхилення кандидата. Це жорстке обмеження має пріоритет над значенням $Q(C)$.

Для кожного структурно допустимого кандидата C обчислюється інтегральний показник узгодженості:

$$Q(C) = 0,2S_{txt}(C) + 0,4S_{ctx}(C) + 0,4S_{flow}(C), \quad (2)$$

значення ваг задає бізнес-аналітик з урахуванням структури, повноти та стилю документації; обґрунтування контрольної конфігурації та процедуру її налаштування наведено нижче.

S_{txt} визначається як середня косинусна подібність усіх неупорядкованих пар нормалізованих назв у групі. Для контрольного прикладу використано TF-IDF-подання символічних n -грам довжиною 3-5 з L2-нормалізацією. Використання NLP для зіставлення назв активностей розглянуто в [6], [7].

S_{ctx} характеризує однорідність фрагмента за ролями та інформаційними системами. Для кожної ознаки обчислюється частка найпоширенішого значення, після чого отримані частки усереднюються. Різні ролі або системи зменшують оцінку, але не забороняють узагальнення.

Структурну складову визначено так:

$$S_{flow}(C) = \frac{2m_C}{2m_C + b_C}, \quad (3)$$

де m_C – кількість переходів, обидва кінці яких належать групі; b_C – кількість переходів, що перетинають її межу. Високе значення S_{flow} відповідає фрагменту з переважанням внутрішніх зв'язків і чіткою зовнішньою межею, але не замінює перевірки цілісності розгалуження.

Вагові коефіцієнти можуть задаватися бізнес-аналітиком з урахуванням структури, повноти та стилю документації. Для контрольного експерименту використано наведені значення. Для документів D1-D5 текстова складова отримала меншу вагу рівну 0.2, оскільки назви пов'язаних операцій могли відрізнятися за формулюванням. Контекстний і структурний складовим надано ваги 0.4, оскільки відомості про ролі, інформаційні системи та переходи

були більш стійкими ознаками належності операцій до одного етапу. Значення $Q_{min}=0,65$ для переходу $L_0 \rightarrow L_1$ обмежувало утворення недостатньо однорідних первинних груп, а нижчий поріг $Q_{min}=0,50$ для переходу $L_1 \rightarrow L_2$ допускав формування узагальнених макроетапів. Для інших наборів документації параметри можуть налаштовуватися шляхом порівняння отриманого розбиття з еталонним групуванням, сформованим аналітиком. Аналіз чутливості при цьому виконується повторним розрахунком ієрархії для сусідніх значень ваг і порогів із подальшим порівнянням кількості та складу сформованих груп. У межах цього експерименту параметри було підібрано експертно для контрольного набору, тому їх не слід вважати статистично оптимальними або універсальними.

Кандидати з $Q(C) < Q_{min}$ відкидаються. Решта впорядковується за спаданням $Q(C)$, далі – за $S_{flow}(C)$, величиною скорочення та стабільним порядком ідентифікаторів. Послідовно приймаються неперетинні кандидати, а вузли, що не ввійшли до жодної групи, переносяться на наступний рівень окремо. Радіус локального оточення і максимальний розмір кандидата обмежують кількість перевірок. Побудова завершується, якщо на новому рівні немає допустимих груп або кількість видимих вузлів не зменшується. За фіксованих параметрів і порядку ідентифікаторів результат є відтворюваним.

Нехай максимальний розмір кандидата обмежено значенням s . Для групи з p вузлів текстова складова потребує $p(p-1)/2$ попарних порівнянь назв, контекстна складова – одного проходу за ролями й системами, а структурна – підрахунку внутрішніх і граничних ребер. Оскільки $p \leq s$, вартість оцінювання окремої групи є обмеженою; основні витрати визначає кількість кандидатів у локальних оточеннях. Перевірка цілісності розгалуження виконується раніше за числове оцінювання, тому процесно некоректні групи не потребують обчислення всіх компонентів Q .

Поріг Q_{min} керує ступенем стискання: надто високе значення залишає багато одноелементних агрегатів, а надто низьке збільшує ризик об'єднання слабо пов'язаних операцій. Вага S_{ctx} посилює вплив схожості назв, S_{ctx} – спільності ролей і систем, S_{flow} – внутрішньої зв'язності фрагмента. У контрольній конфігурації перевагу надано контексту й потоку, оскільки функціонально пов'язані операції часто мають різні дієслова та об'єкти. Для іншої предметної області ваги й пороги мають калібруватися за еталонними рішеннями експертів, а не переноситися без перевірки.

Значення Q не є ймовірністю правильності або заміною експертного рішення. Воно потрібне для порівняння структурно допустимих кандидатів за однієї конфігурації. Жорсткі обмеження, зокрема повне охоплення split-join-регіону, застосовуються незалежно від порога й мають вищий пріоритет.

У D2 зазначено, що за федеративного входу після перевірки політики автентифікації обов'язково перевіряється зовнішній провайдер ідентифікації. Із цього фрагмента формуються кандидат на дію v_8 та переходи $v_7 \rightarrow v_8$ і $v_8 \rightarrow v_9$. D4 уточнює для v_8 роль Identity Analyst, систему External IdP і процесний об'єкт Identity Provider, а сценарій T2 у D5 підтверджує виконуванисть маршруту. Після рішення аналітика вузол входить до G^0 з посиланнями на джерела. На рівні L_1 він належить агрегату C_2 , а на L_2 – агрегату C_6 . Розкриття C_6 послідовно повертає C_2 , v_8 і фрагменти документів, на підставі яких визначено дію, контекст і маршрут.

Для перевірки методу сформовано граф процесу відновлення доступу користувача до програмного продукту з 18 вузлів і 21 переходу. Вихідну модель напівавтоматично отримано з п'яти документаційних артефактів контрольного прикладу: наскрізної специфікації процесу, двох операційних регламентів альтернативних гілок, матриці ролей і систем та сценаріїв приймального тестування. Документи було сегментовано; з фрагментів виділено кандидатів на дії, умови, ролі, системи й переходи; дублікати зіставлено; результат підтверджено аналітиком. Лише після цього граф G^0 використано як вхід для ієрархічного узагальнення.

Вузол v_5 , що відповідає визначенню характеру несправності, утворює дві альтернативні гілки. Перша (v_6-v_9) охоплює перевірку облікового запису, політики автентифікації,

зовнішнього провайдера ідентифікації та відновлення доступу. Друга ($v_{10}-v_{13}$) містить перевірку доступності сервісу, стану застосунку й залежностей та відновлення платформи. Гілки зливаються у v_{14} , після чого перевіряється результат і завершується інцидент.

Початкові й завершальні операції пов'язані з роллю Service Desk і системою ITSM. Першу гілку виконує Identity Analyst із використанням IAM та зовнішнього IdP, другу – Platform Engineer із системами Monitoring та Orchestrator. Ці атрибути використовуються для S_{ctx} , а посилання на фрагменти п'яти документів зберігаються в $A(v)$.

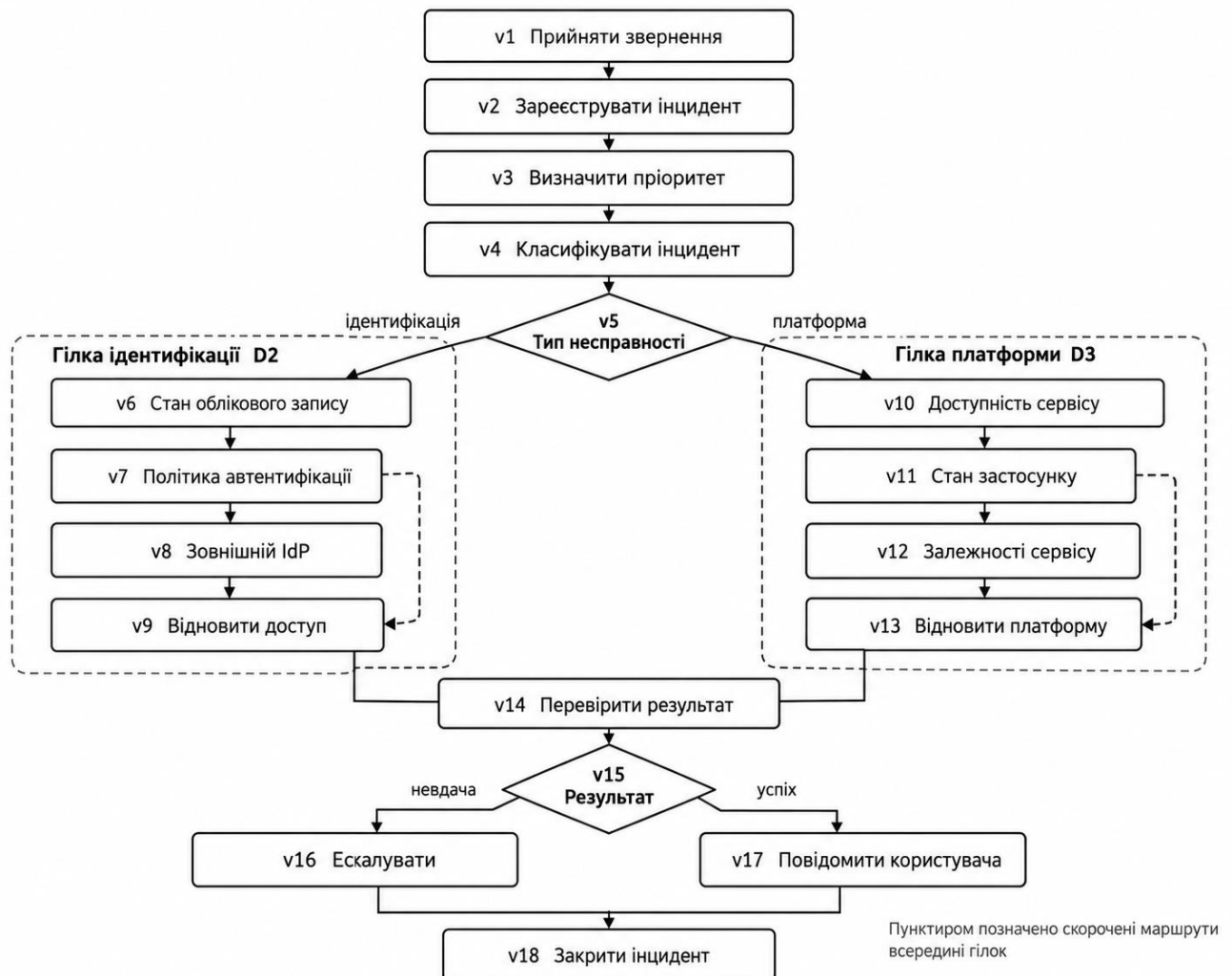


Рис. 1. Структура вихідного графа G^0 контрольного прикладу

Джерело: розроблено автором на основі відкритого набору даних [16]

Перед обчисленням кандидатів вихідний граф перевірено за сценаріями D5. Вони не оцінюють інтерфейс програмного продукту, а контролюють виконуваність переходів, скорочені маршрути всередині гілок і завершення негативного результату через ескалацію.

Коротку характеристику сценаріїв наведено в Таблиці 2.

Сценарії T1 і T3 проходять скороченими шляхами, а T2 і T4 охоплюють усі перевірки відповідної діагностичної гілки. T5 додає негативний результат після спільного вузла v_{15} . Сукупність маршрутів підтверджує, що обидві гілки після v_5 повертаються до v_{14} , а після v_{15} виконується рівно один варіант завершення. Лише після цієї перевірки граф G^0 використовується як вхід для узагальнення.

Оцінки кандидатів, повторно обчислені зі службових файлів, наведено в Таблиці 3.

Таблиця 2. Сценарії структурної перевірки вихідного графа

ID	Сценарій	Відмінна ділянка маршруту	Що перевіряється
T1	Локальна помилка IAM	$v7 \rightarrow v9$	Маршрут без зовнішнього IdP
T2	Недоступність зовнішнього IdP	$v7 \rightarrow v8 \rightarrow v9$	Повний маршрут гілки ідентифікації
T3	Локальна несправність застосунку	$v11 \rightarrow v13$	Скорочений маршрут без $v12$
T4	Несправність залежності	$v11 \rightarrow v12 \rightarrow v13$	Повний маршрут гілки платформи
T5	Невдале відновлення	$v15 \rightarrow v16 \rightarrow v17 \rightarrow v18$	Ескалація, повідомлення і закриття

Джерело: розроблено автором на основі відкритого набору даних [16]

Таблиця 3. Оцінювання кандидатних процесних фрагментів

Фрагмент	S_{txt}	S_{ctx}	S_{flow}	Q	Рішення
$C_1: v_1 - v_4$, приймання і класифікація	0,120	1,000	0,857	0,767	$L_0 \rightarrow L_1$
$C_2: v_6 - v_9$, відновлення доступу	0,104	0,875	0,800	0,691	$L_0 \rightarrow L_1$
$C_3: v_{10} - v_{13}$, відновлення платформи	0,114	0,875	0,800	0,693	$L_0 \rightarrow L_1$
$C_4: v_{14} - v_{18}$, перевірка і завершення	0,142	1,000	0,833	0,762	$L_0 \rightarrow L_1$
$C_5: v_1 - v_5$, кваліфікація інциденту	0,094	1,000	0,800	0,739	$L_1 \rightarrow L_2$
$C_6: v_6 - v_{13}$, діагностика і відновлення	0,120	0,438	0,800	0,519	$L_1 \rightarrow L_2$
$C_x: v_5, v_6, v_7, v_8, v_9, v_{10}$	0,104	0,583	0,800	0,574	відхилено

Джерело: розроблено автором

Усі чотири кандидати переходу $L_0 \rightarrow L_1$ перевищили поріг 0,65: для C_1 отримано $Q = 0,767$, для $C_2 - 0,691$, для $C_3 - 0,693$, для $C_4 - 0,762$. На переході $L_1 \rightarrow L_2$ кандидати C_5 (0,739) і C_6 (0,519) перевищили поріг 0,50. Отже, всі шість робочих кандидатів було прийнято на відповідних переходах між рівнями.

У C_1 і C_4 контекстна однорідність дорівнює 1,000, оскільки операції виконує Service Desk у межах ITSM. Для C_2 і C_3 значення $S_{ctx} = 0,875$: основна роль у межах кожної гілки стала, але одна операція використовує іншу систему. У ширшого кандидата C_6 контекстна складова зменшується до 0,438 через поєднання двох ролей і кількох систем, проте структурна складова 0,800 та поріг другого рівня дають $Q=0,519$. Низька текстова подібність не заважає об'єднанню, якщо група утворює цілісний процесний фрагмент.

Діагностичний кандидат $C_x = \{v_5, v_6, v_7, v_8, v_9, v_{10}\}$ отримав $S_{flow} = 0,800$ та $Q = 0,574$, тобто формально перевищив поріг 0,50. Проте він охопив повну гілку v_6-v_9 і лише початковий вузол альтернативної гілки $v_{10}-v_{13}$. Тому C_x відхилено жорсткою перевіркою цілісності розгалуження незалежно від числового результату.

Отже, метод спочатку відхиляє процесно некоректні фрагменти, а вже потім ранжує структурно допустимі кандидати за інтегральним показником.

На рівні L_1 початкові 18 вузлів подано п'ятьма видимими елементами: чотирма агрегатами C_1-C_4 та окремим вузлом розгалуження v_5 . Між ними збережено п'ять переходів, тому обидві альтернативи залишаються явно видимими й можуть бути розкриті окремо.

На рівні L_2 агрегат C_5 поєднує C_1 з v_5 , C_6 охоплює обидві повні діагностичні гілки, а C_4 залишається завершальним блоком. Подання містить три вузли й два переходи. Кожен агрегат зберігає склад групи, відповідність атомарним вузлам і через них - джерельним фрагментам документації.

Узагальнений вузол має стабільний ідентифікатор C_i та змістову назву. На рівні L_1 такий вузол називається функціональним блоком, оскільки об'єднує атомарні операції зі спільним локальним результатом; на рівні L_2 і вище – макроетапом, що поєднує кілька блоків у ширшу процесну мету. Назви агрегованих блоків формуються незалежно від інтегрального показника Q , який використовується лише для оцінювання та прийняття кандидатної групи. На рівні L_1 агрегат, що відповідає окремому підпроцесу, успадковує назву документа після вилучення службових слів, зокрема Procedure, Runbook, Guide, Standard і Workflow. Для фрагмента основного документа використовується назва відповідного розділу. Якщо агрегат охоплює кілька послідовних розділів, його назва задається діапазоном «перша назва → остання назва». На рівні L_2 назва макрофази аналогічно формується з назв першого й останнього блоків L_1 , що входять до неї. Такий спосіб не створює нових понять, відсутніх у документації, і забезпечує відтворюваність найменування без використання мовної моделі.

Перевірка простежуваності показала, що агрегат C_6 розкривається до вузлів v_6-v_{13} та джерел D2 і D3, а його зовнішні переходи додатково посиляються на D1. Завершальний блок C_4 пов'язаний із D1 і D5, оскільки сценарії приймання підтверджують перевірку результату, ескаляцію, повідомлення та закриття. Отже, на макрорівні зберігається не загальна назва без доказів, а точний перелік атомарних елементів і документів.

Локальне розкриття агрегату замінює його підграфом попереднього рівня та відновлює пов'язані умови й переходи. Решта моделі може залишатися узагальненою. Таке семантичне масштабування змінює лише склад видимих елементів, не змінюючи вихідного графа G^0 .

Назва агрегату є міткою подання, а не заміною його вмісту. Незалежно від назви зберігаються ідентифікатор C_i , склад вузлів, граничні точки, переходи, умови, атрибути й посилання на джерела. Композиція відображень повертає макровузол до його атомарних вузлів G^0 , тому локальне розкриття відновлює відповідний підграф. Отже, скорочення стосується лише кількості видимих елементів; правило найменування можна змінювати незалежно від структури агрегату без зміни складу агрегату або простежуваності.

Повторний запуск сценарію за файлами вузлів, переходів і кандидатів відтворив усі значення Таблиці 3 та показники рівнів із Таблиці 4 з точністю до наведеного округлення. Така перевірка усуває помилки ручного перенесення чисел, але не є незалежною апробацією методу: вона підтверджує лише узгодженість даних, реалізації формул і результатів контрольного прикладу.

Ступінь скорочення кількості видимих вузлів і переходів визначено так:

$$R_V^{(l)} = 1 - \frac{|V^{(l)}|}{|V^{(0)}|}, \quad R_E^{(l)} = 1 - \frac{|E^{(l)}|}{|E^{(0)}|}. \quad (4)$$

Результати кількісного оцінювання сформованих рівнів наведено в Таблиці 4.

Таблиця 4. Характеристики рівнів представлення процесної моделі

Рівень	Вузли	Переходи	RV (%)	RE (%)
L_0	18	21	0	0
L_1	5	5	72,2	76,2
L_2	3	2	83,3	90,5

Джерело: рохроблено автором

Перехід до L_1 скоротив кількість видимих вузлів на 72,2 %, а переходів – на 76,2 %. Для L_2 відповідні значення становили 83,3 % і 90,5 %. Послідовне зменшення обох показників підтверджує, що метод формує компактніші рівні подання.

Скорочення кількості видимих вузлів і переходів характеризує компактність ієрархічного представлення, але не є безпосереднім показником його зрозумілості для користувача. У межах контрольного експерименту коректність узагальнення оцінювалася за структурними інваріантами: кожен вузол попереднього рівня належить рівно одному агрегату або переноситься як окремий елемент; групи одного рівня не перетинаються; умови,

альтернативні гілки та переходи не вилучаються з первинного графа G^0 , а залишаються доступними під час розкриття агрегованого блока.

Отримане скорочення структурної складності узгоджується з підходами до процесної абстракції та ієрархічного подання [1], [2], [3], [4], [5]. Водночас контрольний приклад показав, що семантична й контекстна подібність не можуть замінити перевірку керувального потоку: високий Q не виправдовує часткового приховування альтернативної гілки.

Практично метод дає змогу підтримувати одну процесну модель для різних інформаційних потреб. Загальний огляд показує макроетапи, а аналіз несправності або вимоги розкриває локальний фрагмент разом із посиланнями на джерельні документи. Такий режим відповідає результатам досліджень модульного та ієрархічного подання процесів [10], [11], [12], [13].

Зв'язки з джерелами дають змогу визначити наслідки локальної зміни. Оновлення D2 або D3 потребує повторної перевірки відповідної гілки та агрегатів, що її містять. Зміна ролі або системи в D4 впливає на S_{ctx} і Q кандидатів із відповідним вузлом. Новий маршрут у D5 потребує повторної перевірки керувального потоку. Натомість частини моделі, не пов'язані зі зміненим фрагментом, можуть залишатися без перегляду. Цей режим особливо корисний для продуктів, документація яких уточнюється протягом супроводу.

У контрольному прикладі результати автоматичної сегментації було перевірено аналітиком: сформовані гіпотези підтверджено або відхилено, усунуто дублікати й помилково виділені фрагменти, а також уточнено неоднозначні умови та переходи. Після такої верифікації граф G^0 містив 18 вузлів. Для додаткової перевірки технічної відтворюваності метод застосовано до двох інших комплектів документації [16] без попереднього підтвердження гіпотез аналітиком. Їхні первинні графи на рівні L_0 містили значно більше вузлів через надмірну деталізацію та недоліки автоматичної сегментації, однак на рівнях L_1 і L_2 було сформовано компактніші подання зі збереженням зв'язку з вузлами L_0 . Окремий третій набір – 12-вузловий сценарій активації постачальника – використано для контрольованого порівняння комбінованого, семантичного й топологічного варіантів критерію. Отже, два додаткові комплекти перевіряють роботу реалізації на інших документах, а третій ізолює вплив складових інтегрального показника.

Для контрольованого порівняння використано граф активації постачальника з 12 вузлами та 11 переходами [16]. Еталонне розбиття аналітика містило три чотиривузлові етапи: перевірку відповідності, налаштування інтеграції та активацію розрахунків. У всіх запусках залишено незмінними множину кандидатів, $Q_{min}=0,65$, правила цілісності розгалужень і детермінований порядок відбору. Щоб ізолювати вплив ваг, усі кандидати, зокрема виділені за структурою джерельних документів, приймалися лише за умови досягнення порогу; автоматичний пріоритет для них не застосовувався. Змінювалися лише ваги інтегрального показника. Семантичний базовий варіант задано як $Q_{sem}=S_{txt}$, тобто через наявну в методі TF-IDF-подібність назв операцій; топологічний варіант – як $Q_{top}=S_{flow}$.

Відповідність еталонному розбиттю на рівні L_1 оцінено за pairwise F1. Для кожної пари вузлів перевірялося, чи належать вони до однієї групи одночасно в результаті методу й у розбитті аналітика. Значення 1 означає повний збіг, а 0 – відсутність правильно відтворених пар. Кількість міжетапних груп показує, скільки прийнятих агрегатів перетнули межі еталонних етапів.

Таблиця 5. Кількісне порівняння вагових конфігурацій

Варіант	Ваги	L_1	L_2	Міжетапні групи	Pairwise F1
Комбінований	0,2/0,4/0,4	3/2	2/1	0	1,000
Лише семантичний	1/0/0	12/11	не сформовано	0	0,000
Лише топологічний	0/0/1	4/3	не сформовано	2	0,632

Джерело: розроблено автором за даними з [16]

Комбінований критерій точно відтворив три еталонні етапи на рівні L_1 , скоротивши граф до трьох вузлів і двох переходів, а потім сформував рівень L_2 із двох вузлів та одного переходу. Для нього pairwise F1 дорівнював 1,000, а міжетапні групи були відсутні. Семантичний варіант за незмінного порогу не прийняв жодного кандидата: значення S_{ext} для трьох еталонних груп становили 0,303, 0,499 і 0,431, тобто не досягли 0,65. Тому L_1 залишився тотожним L_0 , а pairwise F1 прийнято рівним 0,000, оскільки жодну з 18 еталонних пар не було об'єднано.

Топологічний варіант скоротив граф до чотирьох вузлів і трьох переходів, однак прийняв дві щільні п'ятивузлові групи, кожна з яких перетнула межу між сусідніми етапами. Через це pairwise F1 знизився до 0,632. Отже, в цьому сценарії семантичний критерій виявився надто суворим за заданого порогу, а топологічний – схильним до об'єднання функціонально різних дій. Комбінований показник уникнув обох помилок і єдиний сформував коректні L_1 та L_2 . Цей результат стосується наведеного сценарію і не доводить універсальної переваги ваг 0,2/0,4/0,4 без перевірки на ширшій вибірці.

Підходи з огляду літератури порівняно окремо за вхідними даними, об'єктом узагальнення та охопленням етапів методу (Таблиця 6). Їх не включено до числового порівняння в Таблиці 5, оскільки вони працюють із готовими процесними моделями, журналами подій або довільними графами. Застосування цих алгоритмів безпосередньо до документації потребувало б окремого перетворення вхідних даних і зміни їхньої початкової задачі.

Таблиця 6. Порівняння з підходами з огляду літератури

Підхід	Вхідні дані	Об'єкт узагальнення	Відмінність від запропонованого методу
Абстракція процесної моделі [1]	Готова процесна модель	Фрагменти моделі	Не охоплює формування G^0 з документації та простежуваність до її фрагментів
Абстракція подій [2], [3], [4]	Журнали подій	Події та траси виконання	Використовує фактичні журнали виконання, а не проєктну документацію
Узагальнення графів [5]	Довільний граф	Структурно пов'язані вузли	Не враховує цілісність процесних розгалужень і джерела документації
Семантичний аналіз [6], [7]	Назви операцій або елементи моделі	Подібність і повнота назв	Оцінює окрему ознаку, але самостійно не формує наскрізну ієрархію
Запропонований метод	Документація та перевірений граф G^0	Рівні L_0 – L_n процесної моделі	Поєднує текст, контекст, потік, цілісність гілок і простежуваність

Джерело: розроблено автором

Порівняння показує, що праці [1], [2], [3], [4], [5], [6], [7] є релевантними аналогами окремих складових, але не повними аналогами запропонованого наскрізного методу. Його відмінність полягає в послідовному переході від документаційних фрагментів до перевіреного G^0 , а від нього – до ієрархії L_1 – L_n зі збереженням джерел. Тому кількісний внесок комбінованого критерію оцінено через внутрішні базові конфігурації на одному графі, тоді як зіставлення з літературою характеризує відмінність постановок задачі.

Висновки. Запропоновано метод ієрархічного узагальнення процесної моделі програмного продукту, сформованої на основі документаційних артефактів і підтвердженої бізнес-аналітиком. Метод зберігає відповідність між агрегатами, атомарними вузлами та

фрагментами-джерелами, а кандидатні групи оцінює за текстовою, контекстною й структурною узгодженістю. Перевірка цілісності розгалужень виконується до числового ранжування.

У контрольному графі з 18 вузлів і 21 переходу рівень L_1 скоротив кількість видимих вузлів на 72,2 % і переходів на 76,2 %, а L_2 – на 83,3 % і 90,5 % відповідно. Кандидат із $Q=0,574$ було відхилено через порушення цілісності альтернативної гілки. Це підтвердило, що структурне обмеження запобігає спотворенню процесної логіки навіть за достатнього значення інтегрального показника. У додатковому 12-вузловому сценарії комбінований критерій відтворив еталонне розбиття (pairwise $F1=1,000$) і сформував L_2 . Семантичний варіант за тим самим порогом не сформував груп, а топологічний утворив дві міжетапні групи (pairwise $F1=0,632$).

Пряме кількісне порівняння запропонованого методу з наявними підходами до узагальнення процесів є обмеженим через відмінність розв'язуваних задач і використовуваних вхідних даних. Підхід [1] передбачає спрощення вже побудованої процесної моделі, тоді як методи [2], [3], [4] формують ієрархічні представлення на основі подій, зафіксованих у журналах виконання процесів. Метод [5] розв'язує загальнішу задачу узагальнення графа шляхом групування його вузлів, але не враховує специфіку документаційних джерел і процесного потоку. Підходи [6], [7] дають змогу оцінювати семантичну близькість або повноту назв операцій, однак не формують наскрізної ієрархії процесної моделі. На відміну від них, запропонований метод працює з проектною документацією, охоплює її сегментацію, формування та перевірку первинного графа G^0 , а потім будує рівні $L_1 - L_n$ з урахуванням текстових, контекстних і структурних ознак та зі збереженням простежуваності до джерел. Отже, зазначені підходи є методологічним підґрунтям і частковими аналогами окремих етапів, але не повними функціональними аналогами запропонованого наскрізного перетворення.

Отже, результатом є кероване багаторівневе подання, у якому функціональні блоки й макроетапи слугують точками навігації до вихідної моделі. Стабільні ідентифікатори забезпечують однозначність обчислень і посилань, а змістові назви - зрозумілість рівнів L_1 та L_2 . Усі подання формуються з одного підтвердженого графа G^0 .

Практично така організація дає змогу використовувати одну модель для різних завдань. На верхньому рівні вона показує послідовність макроетапів, а під час аналізу вимоги, помилки або зміни документації чи програмного коду користувач розкриває лише потрібний агрегат і отримує його атомарні операції, ролі, системи та джерельні фрагменти. Отже, керування видимістю відокремлюється від керування змістом: подання можна змінювати без дублювання чи редагування G^0 , зберігаючи доказовий ланцюг від макроетапу до документації. Оскільки кожен показаний блок має однозначний склад, користувач може зіставляти подання різної деталізації та перевіряти, якими елементами обґрунтовано його межі. Це важливо під час погодження процесу між аналітиками, розробниками й відповідальними за супровід. Це також створює основу для локального переобчислення залежних агрегатів після оновлення окремого документа.

Відсутність втрати інформації стосується підтвердженого графа G^0 : його вузлів, ребер, умов, атрибутів і зв'язків із джерелами. Метод не компенсує пропуски початкової документації; їх потрібно усунути під час формування G^0 . Після цього узагальнення змінює лише спосіб відображення моделі, а кожен агрегат можна розкрити до атомарних операцій і джерельних фрагментів.

Подальша робота має охопити апробацію на реальних моделях, калібрування параметрів за експертними еталонами та оцінювання впливу рівнів узагальнення на швидкість і коректність інтерпретації. Окремо доцільно дослідити локальне оновлення ієрархії після змін у документації.

Конфлікт інтересів: Автор декларує, що не має конфлікту інтересів, зокрема фінансового, особистого, авторського чи будь-якого іншого характеру, який міг би вплинути на дослідження, а також на результати, опубліковані в цій статті

Фінансування: Дослідження проводилося без фінансової підтримки

Доступність даних: Документаційні набори, контрольні графи, результати експериментів і програмну реалізацію розміщено у відкритому репозиторії [16]; ключові параметри й підсумкові результати наведено також у тексті рукопису.

Використання засобів штучного інтелекту: Під час підготовки рукопису використовувався інструмент ChatGPT (OpenAI GPT-5.6 Sol) штучного інтелекту (ШІ) для граматичної та стилістичної перевірки тексту. Усі фрагменти, опрацьовані за його допомогою, були перевірені й відредаговані автором. ШІ не використовувався для генерації тексту, формул і таблиць, формування наукових положень, отримання результатів або формулювання висновків. Наукові положення, результати та висновки є власним інтелектуальним внеском автора

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Tsagkani, C. & Tsalgaidou, A. “Process model abstraction for rapid comprehension of complex business processes”. *Information Systems*. 2022; 103: 101818. DOI: <https://doi.org/10.1016/j.is.2021.101818>.
2. van Zelst, S. J., Mannhardt, F., de Leoni, M. & Koschmider, A. “Event abstraction in process mining: literature review and taxonomy”. *Granular Computing*. 2021; 6 (3): 719–736, <https://www.scopus.com/pages/publications/85107857424>. DOI: <https://doi.org/10.1007/s41066-020-00226-2>.
3. Li, C.-Y., van Zelst, S. J. & van der Aalst, W. M. P. “An activity instance based hierarchical framework for event abstraction”. In *2021 3rd International Conference on Process Mining (ICPM)*. IEEE. 2021. p. 160–167. DOI: <https://doi.org/10.1109/ICPM53251.2021.9576868>.
4. Lu, X., Gal, A. & Reijers, H. A. “Discovering hierarchical processes using flexible activity trees for event abstraction”. In *2020 2nd International Conference on Process Mining (ICPM)*. IEEE. 2020. p. 145–152, <https://www.scopus.com/pages/publications/85096594196>. DOI: <https://doi.org/10.1109/ICPM49681.2020.00030>.
5. Merchant, A., Mathioudakis, M. & Wang, Y. “Graph summarization via node grouping: A spectral algorithm”. In: *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. Association for Computing Machinery. 2023. p. 742–750. DOI: <https://doi.org/10.1145/3539597.3570441>.
6. Kopp, A. M. & Orlovskiy, D. L. “An algorithm for NLP-based similarity measurement of activity labels in a database of business process models”. *Вісник Національного технічного університету «ХПІ». Серія: Системний аналіз, управління та інформаційні технології*. 2023; 1 (9): 54–59. DOI: <https://doi.org/10.20998/2079-0023.2023.01.08>.
7. Rudskiy, O., Kopp, A., Goncharenko, T. & Gamayun, I. “Intelligent technology for semantic completeness assessment of business process models”. *Вісник Національного технічного університету «ХПІ». Серія: Системний аналіз, управління та інформаційні технології*. 2024; 2 (12): 56–65. DOI: <https://doi.org/10.20998/2079-0023.2024.02.09>.
8. Winter, M., Pryss, R., Fink, M. & Reichert, M. “Towards measuring and quantifying the comprehensibility of process models: the process model comprehension framework”. *Information Systems and e-Business Management*. 2023; 21 (3): 723–751. DOI: <https://doi.org/10.1007/s10257-023-00642-2>.
9. Abbad-Andaloussi, A., Burattin, A., Slaats, T., Kindler, E. & Weber, B. “Complexity in declarative process models: Metrics and multi-modal assessment of cognitive load”. *Expert Systems with Applications*. 2023; 233: 120924. DOI: <https://doi.org/10.1016/j.eswa.2023.120924>.
10. Schreiber, C., Abbad-Andaloussi, A. & Weber, B. “On the cognitive and behavioral effects of abstraction and fragmentation in modularized process models”. *Information Systems*. 2024; 125: 102424. DOI: <https://doi.org/10.1016/j.is.2024.102424>.
11. Baß, J., Winter, M., Pryss, R. & Reichert, M. “Exploring comprehension strategies of modular process models: A combined eye-tracking and concurrent think-aloud study”. *Brain Sciences*. 2024; 14 (4): 303. DOI: <https://doi.org/10.3390/brainsci14040303>.
12. Winter, M. & Pryss, R. “The effects of modular process models on gaze patterns – A follow-up investigation about modularization in process model literacy”. *Expert Systems with Applications*. 2024; 237: 121176. DOI: <https://doi.org/10.1016/j.eswa.2023.121176>.

13. Jurik, P., Schmidt, P., Misut, M., Brezina, I. & Reiff, M. “The composition diagram of a complex process: Enhancing understanding of hierarchical business processes”. *Information Systems*. 2025; 128: 102489. DOI: <https://doi.org/10.1016/j.is.2024.102489>.

14. Möller, M., Winter, M. & Reichert, M. “Cognitive factors in process model comprehension – A systematic literature review”. *Brain Sciences*. 2025; 15 (5): 505. DOI: <https://doi.org/10.3390/brainsci15050505>.

15. Kopp, A., Orlovskyi, D., Gamayun, I. & Sapozhnykov, I. “Software development and research for machine learning-based structural errors detection in BPMN models”. *Вісник Національного технічного університету «ХПІ»*. Серія: Системний аналіз, управління та інформаційні технології. 2024; 2 (12): 46–55. DOI: <https://doi.org/10.20998/2079-0023.2024.02.08>.

16. “Hierarchical generalization of a software product process model: research data and implementation”. *GitHub*. 2026. – Available from: <https://github.com/halushko/articles/tree/2026-06-sermentation>. – [Accessed: 25, August 2026].

Отримано 28.08.2026

Отримано після перегляду 24.09.2026

Прийнято 08.10.2026

DOI: <https://doi.org/10.15276/ict.03.2026.45>

UDC 004.942:004.415

A method for hierarchical generalization of a software product process model

Dmytro O. Halushko

ORCID: <https://orcid.org/0000-0002-1980-7387>; halushko.dmytro@edu.kpi.ua
National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”,
37, Beresteyskyi Ave. Kyiv, 03056, Ukraine

ABSTRACT

This paper proposes a method for hierarchical generalization of a software product process model formed from documentation artifacts. The initial model is derived from an end-to-end process specification, two operating procedures, a role-and-system matrix, and acceptance-test scenarios. The documents are segmented into logical fragments, candidates for actions, conditions, roles, systems, and transitions are extracted, and only elements validated or edited by a business analyst are included in the graph. The model is formalized as a typed directed graph whose attributes retain links to documentation sources. Candidate groups are generated within a local graph neighborhood and checked for the integrity of alternative branches. Structurally valid fragments are evaluated by an integral metric combining textual, contextual, and control-flow components. The method was checked on test graph. After execution of the method the numbers of visible nodes and transitions decreased. During generalization, the method preserves the control flow, branching integrity, and traceability of aggregated elements to the source documentation. The proposed approach combines the semantic characteristics of process elements, their execution context, and the control-flow structure, which makes it possible to form consistent levels of representation without violating the logic of alternative branches. The reproducibility of the construction is ensured by fixed rules for forming candidate groups, preserving links to the source documents, and successively expanding generalized elements to the atomic level. The method was evaluated on a test graph. After generalization, the numbers of visible nodes and transitions decreased. A candidate with a sufficient numerical score was rejected because it covered only part of an alternative branch. The method provides a compact multilevel view while preserving traceability to atomic process elements and the documentation from which they were obtained. The proposed approach can be applied to the multilevel representation of software product process models.

Keywords: Process model; software product; documentation; hierarchical generalization; traceability; directed graph; process abstraction; control flow

ІНФОРМАЦІЯ ПРО АВТОРА



Галушко Дмитро Олександрович - кандидат технічних наук, доцент кафедри Інформаційних систем та технологій. Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Берестейський просп., 37. Київ, 03056, Україна

Галузь досліджень: Інформатика та інтелектуальні інформаційні технології

Dmytro O. Halushko - Candidate of Technical Sciences, Associate Professor, Department of Information Systems and Technologies. National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”,
37, Beresteyskyi Ave. Kyiv, 03056, Ukraine

ORCID: <https://orcid.org/0000-0002-1980-7387>; halushko.dmytro@edu.kpi.ua

Research field: Informatics and Intelligent Information Technologies