УДК 004.8

Д. О. Галушко, К. В. Знова, О. І. Ролік

АВТОМАТИЗОВАНА СИСТЕМА НАЛАШТУВАННЯ ПРОДУКТІВ ДЛЯ ПРОВАЙДЕРІВ ІНФОРМАЦІЙНИХ ПОСЛУГ

Анотація: Розроблена автоматизована система налаштування продуктів для провайдерів інформаційних послуг. Запропоновано підхід до автоматизації аналізу змін у програмних продуктах, розроблених для провайдерів інформаційних послуг, на основі гетерогенних графових нейронних мереж та алгоритмів обробки природної мови. Розглянуто інтеграцію графу, побудованого на базі декомпозиції сценаріїв використання, документації та програмного коду для ідентифікації змін, ризикованих місць і можливостей перевикористання коду. Наведено приклади реалізації з використанням моделей BERT та графових алгоритмів для покращення стабільності та гнучкості системи.

Ключові слова: алгоритми обробки природної мови (NLP), гетерогенні графові нейронні мережі (HeteroGNN), декомпозиційний аналіз, машинне навчання, інформаційні послуги, інформаційні системи та технології, моделі BERT.

Вступ

У сучасному світі інформаційні послуги стають дедалі складнішими. Це вимагає від провайдерів інформаційних послуг (ПІП) швидкої реакції задля адаптування своїх продуктів до постійних змін умов на ринку надання послуг великій кількості користувачів. Кількість продуктів, що надають ПІП своїм користувачам, постійно зростає, їх складність підвищується, взаємозв’язок між послугами стає сильнішим. Це призводить до того, що процеси налаштування та оновлення продуктів стають трудомісткими. Традиційні ручні методи внесення змін до програмних продуктів (ПП) вже не можуть забезпечити таку швидкість і точність, яку вимагає ринок телекомунікаційних послуг. Традиційні ручні методи налаштування ПП часто призводять до затримок, збільшення операційних та фінансових витрат та ризику втрати якості обслуговування клієнтів. Це, в свою чергу, призводить до відтоку клієнтів та зменшення рентабельності інвестицій – ROI. Виходом з такої ситуації є автоматизація усіх етапів життєвого циклу ПП – від розробки, впровадження, модифікації до утилізації. Особливо важливими є автоматизація процесів розроблення ПП та внесення змін в існуючі ПП. Це дозволяє не тільки підвищити ефективність створення ПП, а й зменшити ризики під час внесення змін до вже існуючої кодової бази.

Серед процесів створення та впровадження ПП слід виокремити процес налаштування ПП. Без автоматизації цього процесу неможлива швидка адаптація до змін ринку. Сучасні підходи до автоматизації створення та налаштування ПП базуються на

 ISSN 1560-8956

використанні передових технологій та інструментів, які забезпечують гнучкість, масштабованість та надійність. Провідні компанії світу почали впроваджувати сучасні рішення для автоматизації аналізу та вдосконалення своїх програмних систем. Microsoft створила та активно використовує CodeBERT [1] – модель на основі трансформерів для представлення програмного коду. Модель основана на обробці природної мови та використовується в задачах розуміння коду, таких як прогнозування наступного токена, класифікація коду та інші. Це дозволяє моделі не тільки передбачати, яким має бути наступний блок коду, а й сприяє автоматизації аналізу коду та виявленню потенційних проблем при внесенні змін [2]. Facebook досліджує використання графових нейронних мереж для аналізу взаємодії між компонентами своїх систем [3]. Зокрема, вони використовують графові нейронні мережі для виявлення вразливостей та оптимізації продуктивності в масштабних програмних системах. Uber впровадила керування залежностями між своїми мікросервісами на основі графових нейронних мереж [4], що дозволяє автоматично аналізувати залежності та вплив змін на систему. Alibaba застосовує гетерогенні графові нейронні мережі для оптимізації своїх рекомендаційних систем та аналізу логістичних процесів [5], що включають складні взаємодії між різними компонентами. Ці приклади засвідчують зростаючу зацікавленість у використанні методів машинного навчання для автоматизації аналізу складних програмних систем.

Аналіз літературних джерел та обґрунтування проблематики роботи

Автоматизація процесу аналізу змін у програмних продуктах є однією з ключових задач сучасних інформаційних технологій (ІТ). Значний внесок у вирішення цієї проблематики зробили дослідники в галузі використання машинного навчання, графових нейронних мереж, а також алгоритмів обробки природної мови.

Методи графових нейронних мереж отримали широке визнання в задачах моделювання складних взаємозв’язків у програмних системах. Так, роботи [1–3] зосереджуються на застосуванні для аналізу залежностей між компонентами великих систем, таких як мікросервіси або розподілені архітектури. Дослідження [3, 5] демонструють ефективність використання гетерогенних графових нейронних мереж при вирішенні задач оптимізації процесів у складних системах, таких як логістика та управління залежностями в ПП.

Технології обробки природної мови активно використовуються для аналізу програмного коду та технічної документації. Роботи [1–2] продемонстрували успішне застосування моделей на основі трансформерів, таких як система обробки природної мови BERT, для задач класифікації коду, аналізу помилок та автоматичного оновлення документації. Зокрема, у [6] запропоновано використовувати NLP для ідентифікації ключових сутностей у документації, однак питання інтеграції цих результатів у загальний процес аналізу систем залишається відкритим.

У дослідженнях [7–9] акцент зроблено на створенні графів залежностей для управління складними системами. Роботи [10–11] показали, що побудова графів дій дозволяє моделювати процеси у програмних системах із високою точністю. Проте, більшість підходів обмежуються використанням внутрішніх даних, таких як логи або структура баз даних, не враховуючи зміни, що випливають із оновлень у документації.

Компанії, що спеціалізуються на створенні ПП, активно впроваджують інноваційні підходи у цій галузі. Наприклад, Microsoft використовує CodeBERT для аналізу програмного коду [1], тоді як Facebook застосовує GNN для виявлення вразливостей [3]. Uber розробила інструменти для аналізу залежностей у мікросервісах [4]. Alibaba інтегрує графові нейронні мережі в логістичні системи, що дозволяє автоматизувати складні процеси [5].

Попри практичні успіхи, більшість досліджень зосереджується на аналізі структурованих даних, таких як виклики API або модулі систем. Проблема застосування цих підходів для аналізу текстових документів, які містять функціональні вимоги або нотатки про зміни, залишається недостатньо дослідженою. Тому, незважаючи на суттєвий прогрес в розробленні та використанні перспективних методів та засобів автоматизації створення та аналізу ПП, залишається невирішеним така ключова проблема, як інтеграція аналізу текстової документації з кодовою базою. Вирішення цієї проблеми необхідно для побудови єдиного процесу та створення відповідного інструментарію, який об’єднає аналіз документації та коду з подальшою генерацією рекомендацій щодо змін у коді за результатами аналізу. Тому ця робота присвячена вирішенню зазначеної проблеми шляхом інтеграції моделей BERT, гетерогенних графових нейронних мереж та алгоритмів обробки графів у єдиний автоматизований процес.

Мета та задачі дослідження

Метою роботи є розроблення концепції автоматизації процесу додавання нового функціоналу або внесення змін до готового ПП, який використовують ПІП для надання послуг клієнтам. Для досягнення цієї мети пропонується підхід, що об’єднує графовий аналіз та обробку природної мови з ідентифікацією змін у функціоналі ПП, виявленням критичних частин системи та пошуком методів і засобів ефективного використання існуючої кодової бази.

Для реалізації такого підходу необхідно автоматизувати аналіз існуючої архітектури надання послуг, баз даних, інтерфейсів та інших критичних компонентів ПП задля вдосконалення ПП. За результатами аналізу мають коригуватися плани впровадження та модифікуватися відповідні технічні рішення. Це дозволить розробникам ПП автоматизувати частину процесів створення ПП та отримувати рекомендації щодо поліпшення та пришвидшення даного процесу.

Матеріали та методи досліджень

Об’єктом дослідження є процеси ПП, що використовують ПШП, для надання послуг великої кількості користувачів. Особливістю цих ПП є те, що вони потребують регулярного оновлення та налаштування для адаптації до умов ринку. Предметом дослідження є підходи до автоматизації та інтеграції змін у ПП за допомогою гетерогенних графових нейронних мереж, алгоритмів обробки природної мови та засобів аналізу коду.

Для виявлення ключових взаємозв’язків та оцінки впливу змін у документації на відповідні зміни у програмному коді використовувалися графові методи аналізу [8–9]. Алгоритми PageRank та Betweenness Centrality застосовано для визначення критичних вузлів у графах [7, 11], які були побудовані на основі декомпозиції [16] поставлених перед ПП задач. Для виділення сутностей та їх взаємозв’язків у нотатках зустрічей та документації ПП використано модель BERT [15]. Попередньо виконувались стандартні процедури токенізації, нормалізації тексту та витягування ключових слів [15]. Для обробки програмного коду застосовано AST-парсери, що забезпечують аналіз залежностей між методами, класами та модулями [2, 6]. Для моделювання графів із різнорідними вузлами та взаємозв’язками між ними використовувалися гетерогенні нейронні мережі [8–10]. Для ідентифікації у графі схожих компонентів, які можна перевикористати, застосовувався алгоритм K-Means [7].

В роботі використовуються такі стандарти: ISO/IEC 12207:2017 – процеси життєвого циклу програмного забезпечення; RFC 2616 – протокол HTTP для побудови RESTful API та Python PEP 8 – як керівництво з оформлення коду на мові програмування Python.

Опис задачі

Сучасні продуктові компанії, що розробляють ПП для ПШП, працюють в умовах інтенсивного розвитку інформаційних технологій. Жорстка конкуренція на ринку IT-послуг примушує такі компанії постійно розробляти нові продукти з новими властивостями, які зацікавлять клієнтів ПШП. Окрім того, потрібно постійно вдосконалювати вже існуючі ПП, для того, щоб зробити їх більш привабливими для клієнтів ПШП. Це викликає необхідність впроваджувати сучасні методи автоматизації процесів життєвого циклу програмних продуктів. Життєвий цикл ПП – це послідовність етапів, які проходить продукт від початкової ідеї до його експлуатації, підтримки та утилізації. На рис. 1 зображено схему взаємодії основних процесів життєвого циклу ПП.

На етапі обговорення із замовником команда аналітиків збирає та аналізує вимоги замовника, визначає бізнес-цілі, технічні потреби та бажані результати. Чітке формулювання функціональних вимог дозволяє уникнути суттєвих проблем на наступних етапах. На основі обговорень створюється технічна документація, яка включає опис

функціональних і нефункціональних вимог, архітектури, інтерфейсів і ресурсів, необхідних для реалізації продукту. Документація є основою для розробки ПП та управління проєктом.

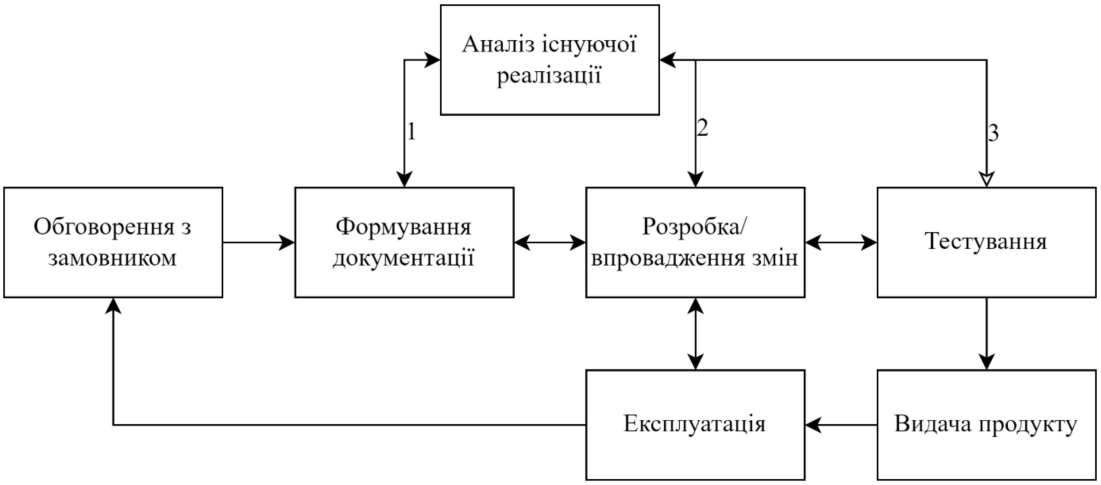

Рисунок 1. Взаємодія процесів життєвого циклу ПП у продуктовій ІТ компанії

Після того, як документація з вимогами сформована, починається етап розробки. Він охоплює проєктування архітектури, написання програмного коду, налаштування баз даних та інтеграцію системи. Розробники працюють відповідно до визначених специфікацій, забезпечуючи узгодженість з вимогами.

На наступному кроці відбувається тестування ПП – перевірка якості продукту, включаючи модульне, інтеграційне, регресійне тестування та тестування продуктивності. Тестування забезпечує відповідність продукту початковим вимогам та його стабільність у реальних умовах експлуатації.

Після того, як інженери-тестувальники визначають, що продукт відповідає необхідним вимогам до якості, продукт готується до релізу. Залежно від масштабів змін, реліз може проходити у кілька етапів, наприклад, у вигляді пілотного тестування для обмеженої аудиторії.

Нарешті, коли продукт проходить всі перевірки, він переходить на етап експлуатації і підтримки. Починається використання ПП кінцевими користувачами. При цьому здійснюється моніторинг ПП, підтримка та внесення необхідних змін. У разі появи додаткових вимог замовника або виявлення проблем розпочинається новий цикл розробки або вдосконалення ПП.

В роботі пропонуються методи вдосконалення процесу внесення змін та оновлень до продуктів, що вже перебувають на етапі експлуатації. Для цього пропонується автоматизувати процеси аналізу вивчення архітектури, баз даних, інтерфейсів та інших критичних елементів продукту, що дозволить краще підготувати впровадження змін у ПП.

 ISSN 1560-8956

Результати аналізу допоможуть скоригувати план внесення змін та підготувати відповідні технічні рішення для здійснення змін. Завдяки автоматизації частини процесів продуктові компанії можуть отримати рекомендації щодо поліпшення та пришвидшення процесу внесення змін та подальшої автоматизації процесів життєвого циклу ПП.

Під час розробки та тестування ПП, а також створення та внесення змін у документацію відбувається постійний аналіз поточного стану системи та продуктів задля виявлення потенційних проблем, вузьких місць та можливих ризиків.

Введемо поняття «граф дій» – модель, яка являє собою формалізоване представлення послідовності виконання завдань або процесів у системі. Таке представлення процесів у системі дозволяє зробити декомпозицію складних процесів на підпроцеси та операції (дії), що виконуються у визначеному порядку. Така декомпозиція забезпечує структуроване уявлення про функціонування системи та дозволяє виявляти ключові точки взаємодії між її компонентами. Вся інформація, що необхідна для побудови графу, отримується з документації та аналізу кодової бази.

Формування графу включає кілька етапів. На початку визначаються окремі дії, які мають бути виконані. Наприклад, це можуть бути виклики API, операції у базі даних, виконання методів з кодової бази або взаємодія між модулями системи. Кожна дія стає вузлом цього графу та отримує атрибути, які її описують (назва, тип, пріоритет, тощо). Далі встановлюються зв’язки між вузлами на основі залежностей, які існують у процесі, що розглядається.

Кожен вузол та ребра графа мають атрибути, які характеризують їх природу [7, 10], наприклад: назва дії, тип (код, запит у базу даних, виклик АПІ), важливість для системи або належність до певного продукту. Кожне ребро графу дій має напрямок (від вузла-джерела до вузла-цілі), що відповідає порядку виконання дій.

Такий граф може містити як прямі залежності (наприклад, модуль викликає метод), так і непрямі, що передбачають вплив змін у вузлі на інші вузли через проміжні зв’язки.

Граф буде зростати разом із системою при додаванні нового функціоналу чи навіть при зміні існуючого. Побудова такого графу дозволяє застосовувати алгоритми для визначення ключових вузлів, прогнозування впливу змін, оптимізації послідовності виконання або перевірки консистентності зв’язків.

Якщо процес має складну структуру, вузли об’єднуються в підграфи, які відповідають окремим модулям або компонентам. Це дозволяє спрощувати візуалізацію та аналіз.

Архітектура системи

Граф дій є головним елементом, що інтегрує результати роботи різних модулів. Нові вузли та зв’язки додаються на основі аналізу документації, а залежності між компонентами системи інтегруються через результати аналізу кодової бази. Це дозволяє

формувати повну картину взаємодії компонентів, прогнозувати вплив змін та генерувати рекомендації для бізнес-аналітиків і розробників. На рис. 2 зображено структурну схему системи, що розробляється.

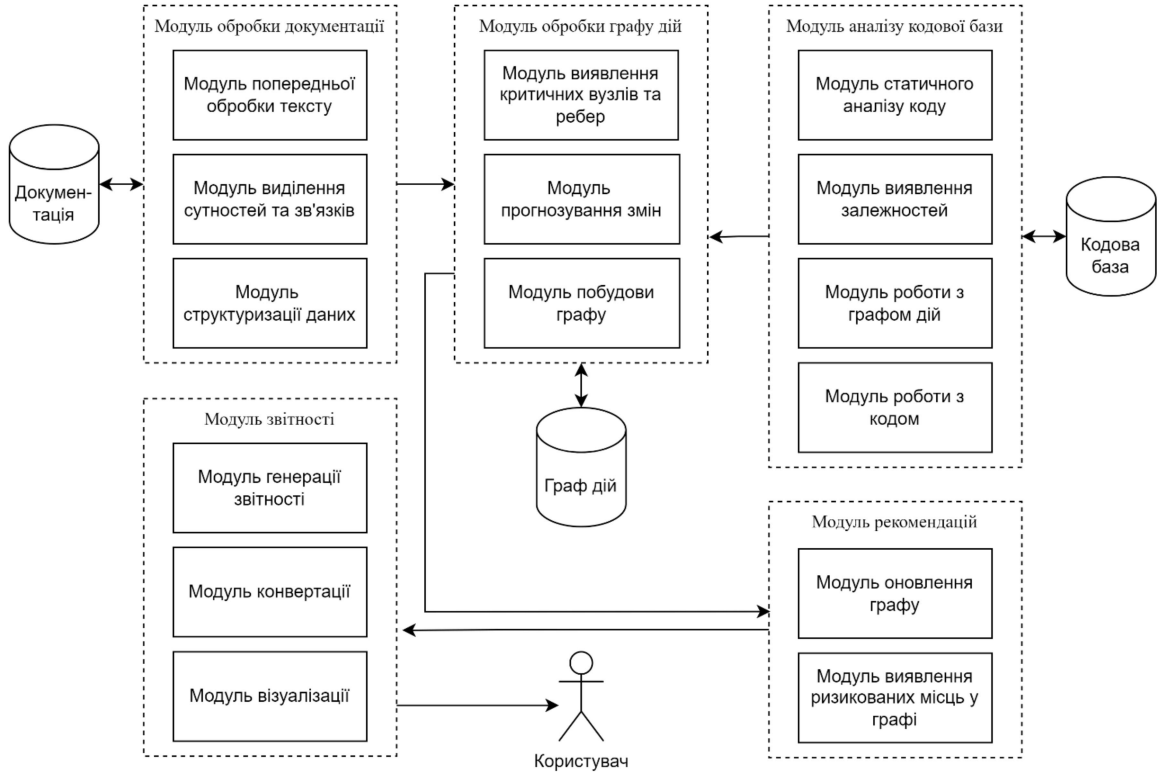

Рисунок 2. Структура системи

Модуль обробки документації отримує текстову документацію у вигляді файлів та виконує попередню обробку тексту – очищення, нормалізацію, токенізацію. Також виділяються сутності та зв’язки у тексті, формуються структуровані дані у вигляді JSON для інтеграції з іншими сервісами. Документація зберігається у вигляді структури папок: «Поточна документація», «Зміни». Це дає змогу аналізувати результати обговорення та внесені у документацію бізнес-аналітиками зміни, беручи за основу поточну, вже імплементовану документацію. Після аналізу текстових даних модуль генерує JSON-файли з сутностями та зв’язками, які передаються через API до модуля обробки графу дій, де ця інформація інтегрується в існуючий граф дій.

Модуль обробки графу дій виконує графові алгоритми PageRank для визначення вузлів із високим впливом на процеси системи та Betweenness Centrality для ідентифікації вузлів, через які проходять ключові зв’язки між компонентами. За допомогою гетерогенних графових нейронних мереж система моделює взаємодії між вузлами графу, враховуючи типи вузлів і зв’язків. На основі цих даних прогнозується необхідність регресійного

 ISSN 1560-8956

тестування окремих сценаріїв використання або необхідність додаткових змін до коду. Згенерований граф дій не є остаточним та може бути відкорегований вручну бізнес-аналітиками та програмістами при імплементації змін.

Модуль аналізу кодової бази здійснює статичний аналіз кодової бази, виявляє залежності між модулями, класами та методами за допомогою AST-парсерів. Результати аналізу інтегруються у граф дій. Кожен новий вузол і ребро супроводжується метаданими, що вказують на відповідний модуль у кодовій базі. Модуль аналізує граф дій і кодову базу для виявлення схожих функцій або компонентів, які можна перевикористати. Для цього застосовується кластеризація вузлів графу на основі їхніх атрибутів і зв’язків. Результати аналізу, включно з можливими змінами та рекомендаціями щодо перевикористання коду, передаються у форматі JSON для візуалізації в модулі інтеграції або як файл звіту для команди розробників.

Модуль рекомендацій використовує результати роботи модулів обробки документації, графу дій і кодової бази, об’єднуючи їх. На основі аналізу графу та даних із кодової бази модуль використовує критерії, такі як кількість залежностей вузла або його критичність, щоб виявити ризиковані місця. Рекомендації формуються у вигляді структурованого JSON-файлу, який включає список змін у графі, ризиковані вузли, пропозиції щодо перевикористання коду та пріоритизацію змін.

Модуль інтеграції збирає дані від усіх модулів, створюючи звіти для користувачів. Звіти містять графічну візуалізацію змін, перелік рекомендацій і аналіз ризиків. Інтерфейс модуля дозволяє бізнес-аналітикам переглядати звіти та взаємодіяти з графом у реальному часі через інтерактивну візуалізацію. Розробники та тестувальники можуть отримувати технічні рекомендації у вигляді структурованих даних або звітів.

Результати експериментів

Розроблена система вирішує такі задачі: автоматична обробка існуючої документації та нотаток після обговорень; автоматизована ідентифікація змін, які необхідно внести до графу дій при зміні процесу; виявлення ризикованих місць у продукті, де зміни можуть вплинути на стабільність або на інші продукти; пошук можливостей для перевикористання існуючого функціоналу при додаванні нових продуктів.

Для тестування розробленої системи використано запропонований первинний сценарій «Відключення користувача від послуги постачання інтернету». Попередньо згенеровано кодову базу, документацію та побудовано граф дій (рис. 3). Кожному вузлу відповідає конкретна дія, яку треба реалізувати в системі, щоб виконати запит споживача послуг, а ребра вказують на послідовність виконання дій.

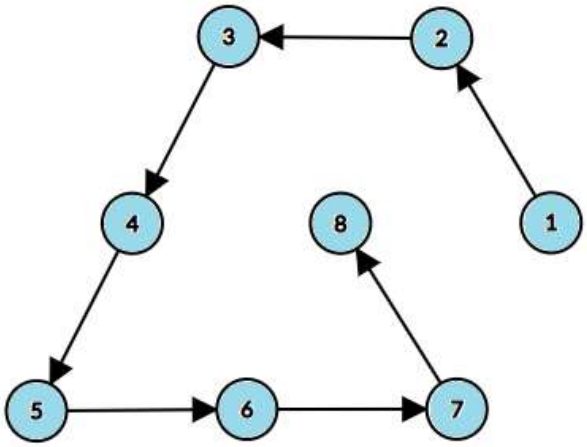

Рисунок 3. Граф дій до надходження змін

Дії в первинному сценарії:

1. Очікувати на заявку від споживача послуги на відключення від інтернету (Wait for event).

2. Надіслати електронного листа споживачу, що його заявка прийнята в обробку (Send email).

3. Надіслати запит на внутрішню систему провайдера, щоб визначити порт свіча, на якому споживач знаходиться (REST request).

4. Віддалено вимкнути отриманий порт (REST request).

5. Відключити споживача від послуги (REST request).

6. Згенерувати заявку для працівників провайдера, щоб вони фізично від’єднали кабель доступу до інтернету (Send email).

7. Очікувати на заявку про факт виконання робіт (Wait for event).

8. Надіслати лист споживачу про факт виконання його запиту (Send email).

Для аналізу обробки документації та нотаток обговорення змін використано алгоритм обробки природної мови сімейства BERT у поєднанні з федеративним навчанням. Це забезпечило конфіденційність даних та дозволило автоматично проаналізувати документацію та нотатки обговорень і виділити ключові сутності, відношення між ними та сформувати чернетки для оновлення документації.

Перед подачею тексту на обробку виконано попередню очистку тексту, яка включала видалення спеціальних символів, нормалізацію форматування, токенізацію та розбиття тексту на речення. Ці операції необхідні для стандартизації вхідних даних та забезпечення коректної роботи моделі. Модель обчислила векторні представлення для кожного слова з урахуванням його контексту в реченні. На основі цих представлень класифіковано токени. Додатково для кожної сутності класифіковано її зв’язки із іншими сутностями.

Виділені сутності та зв’язки структуровано та виділено текстові блоки з описом дій, їхньої взаємодії з існуючими компонентами, залежностей і можливих ризиків. Після закінчення цього процесу аналітики мають перевірити отримані документи, що дозволить адаптувати результати до специфіки продукту. Такий підхід забезпечує скорочення часу на створення документації, зменшення ризику упущення важливих деталей та відповідність сучасним вимогам до конфіденційності даних.

Вихідними даними на цьому етапі є набір структурованих елементів: сутності з атрибутами (тип, локація у тексті) та граф зв’язків між ними.

Далі у систему ввели зміни до продукту у вигляді текстових нотаток для нового сценарію «Відключення користувача від послуги з декількома точками доступу»: «Якщо користувач має декілька точок доступу, то відключати його лише від тої, на яку він подав заявку. Якщо клієнт відключає останню точку доступу, то відключити його від послуги повністю. Якщо ще є активні точки доступу, то лише змінити його тарифний план».

Система виділила ключові сутності та взаємозв'язки після обробки нової нотатки: «*Користувач*»: має кілька точок доступу; «*Точка доступу*»: ідентифікована для відключення; «*Тарифний план*»: змінюється відповідно до кількості активних точок доступу. Також вона виявила дії: «*Відключення точки доступу*», «*Перевірка кількості точок доступу користувача*», «*Відключення користувача від послуги*», «*Зміна тарифного плану*» та розпізнала логічні умови: «*Якщо це остання точка доступу, виконати повне відключення користувача від послуги*», «*Якщо є активні точки, змінити тарифний план*»

На основі даного аналізу система запропонувала нову документацію:

1. Очікувати заявку від споживача про відключення певної точки доступу.
2. Надіслати електронного листа споживачу, що його заявка прийнята в обробку.
3. Надіслати запит на внутрішню систему провайдера для ідентифікації порту точки доступу, на яку подано заявку.
4. Віддалено вимкнути отриманий порт.
5. Виконати перевірку кількості активних точок доступу користувача:

ЯКЩО це остання точка:

6. Відключити користувача від послуги.
7. Згенерувати заявку для працівників провайдера для фізичного вимкнення кабелю.
8. Очікувати підтвердження виконання робіт.
9. Надіслати лист про завершення запиту.

ІНАКШЕ залишаються активні точки:

10. Змінити тарифний план користувача.
11. Надіслати лист про зміну тарифного плану.

Отриману інформацію система порівняла з поточним графом дій для визначення місць, які потребують змін. На основі цього аналізу отримано вузли та ребра у графі дій, які необхідно додати, видалити або модифікувати.

Алгоритми обробки природної мови витягли з нотатки умови: «якщо», «то», «після цього», та класифікували їх за типами [14]: логічні, послідовні, каскадні. Також вони розпізнали тригери. Були сформовані сценарії:

1. Перевірка кількості точок доступу: результат (одна чи кілька точок) визначає, яка дія має виконуватися далі.

2. Зміна тарифу чи повне відключення: відповідає на результати перевірки.

В основі системи закладені правила, які визначають, як слід трактувати послідовність дій:

1. IF-THEN-ELSE конструкція: «Якщо точок доступу більше однієї, тоді змінити тарифний план, інакше повністю відключити користувача».

2. Правило залежності: деякі дії можуть виконуватися лише після завершення інших – неможливо змінити тарифний план до перевірки кількості точок доступу.

Ця інформація дозволила сформувати новий граф дій (рис. 4).

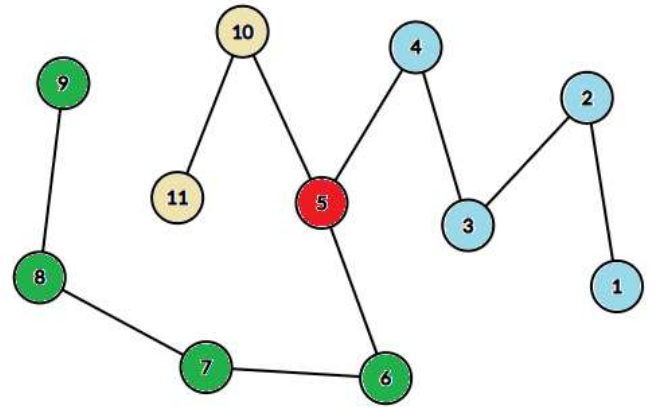

Рисунок 4. Граф дій після обробки змін до документації

Нові модулі та функції система додала до графу як нові вузли з відповідними зв’язками і зафарбувала у зелений колір.

За рахунок використання графових алгоритмів PageRank та Betweenness Centrality система ідентифікувала критичні вузли графу. Вузли з високим значенням центральності вона визначила як найбільш впливові та зафарбувала їх у червоний колір – зміни у них спричинили каскадний ефект на інші компоненти розробленого продукту.

Окрім аналізу графу система врахувала залежності у кодовій базі. Використання AST-парсерів дозволило виявити залежності між модулями, класами та методами, які доповнюють граф дій та порівняти зміни у кодовій базі з оновленнями графу, забезпечивши узгодженість між документацією, графом та реалізацією. Використання цих алгоритмів також дозволило системі виявити вузли у графі, що потребують регресійного тестування. Це дозволило уникнути негативного впливу від змін на стабільність продукту та уникнути

 ISSN 1560-8956

конфліктів або помилок у залежностях. Для легкості сприйняття система зафарбувала їх у жовтий колір.

За рахунок використання алгоритмів кластеризації вузлів графу система виявила схожі підграфи, які вже реалізують частину необхідної функціональності. На графі вони зафарбовані у синій колір. Це дозволило скоротити витрати на розробку та зменшило складність впровадження змін до продукту за рахунок перевикористання існуючих компонентів.

Висновки

Запропоновано та реалізовано підхід до автоматизації процесу внесення змін у програмні продукти для провайдерів інформаційних послуг. Розроблено структуру автоматизованої системи налаштування продуктів для провайдерів інформаційних послуг. Використання гетерогенних графових нейронних мереж та алгоритмів обробки природної мови дозволило створити систему, яка автоматично аналізує текстову документацію, кодову базу та граф дій для генерування рекомендацій щодо інтеграції нових функцій або змін у вже існуючі продукти.

Результати роботи системи показали її здатність автоматично генерувати рекомендації для таких компонентів продукту:

1) графи дій, що враховують нові сценарії та їх умови;

2) виявляти ризиковані місця у функціоналі продукту та пропонувати відповідні рекомендації щодо тестування;

3) генерувати рекомендації щодо змін у документації для бізнес-аналітиків у зручному форматі.

Запропонована система демонструє перспективність застосування інструментів машинного навчання для автоматизації задач, пов’язаних із впровадженням змін до існуючих програмних продуктів. Завдяки модульній архітектурі система може бути адаптована до різних сценаріїв використання, а також інтегрована в існуючі бізнес-процеси провайдерів.

Подальші напрямки розвитку системи включають покращення точності аналізу текстової документації при великих розмірах документації, розширення функціоналу для підтримки більш складних сценаріїв взаємодії між компонентами, а також оптимізацію часу виконання аналізу для великих програмних продуктів.

Таким чином, розроблена система є дієвим інструментом для забезпечення гнучкості та швидкості внесення змін у програмні продукти, що сприяє підвищенню ефективності роботи провайдерів інформаційних послуг.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Z. Feng et al., “CodeBERT: A pre-trained model for programming and natural languages,” in Proc. of the 2020 Empirical Methods in Natural Language Processing, 2020. URL: https://arxiv.org/abs/2002.08155

2. A. Kanade et al., “Learning and evaluating contextual embeddings for code,” in Proc. of ICML 2020, 2020. URL: https://arxiv.org/abs/2005.00057

3. Facebook AI, “Graph Neural Networks for vulnerability detection,” 2020. URL: https://ai.facebook.com/research

4. Uber Technologies, “Graph-based dependency management in microservices,” 2021. URL:https://eng.uber.com/microservices-dependency-graph

5. Alibaba Research, “Heterogeneous GNNs in logistics optimization,” 2021. URL: https://www.alibabacloud.com

6. M. Allamanis and C. Sutton, “Mining source code repositories for structured information,” in Proceedings of the 10th Working Conference on Mining Software Repositories, 2013. URL: https://doi.org/10.1109/MSR.2013.6624020

7. J. Zhou et al., “Graph neural networks: A review of methods and applications,” AI Open, vol. 1, pp. 57–81, 2020. URL: https://doi.org/10.1016/j.aiopen.2020.04.001

8. Microsoft Azure, “Dependency graph analysis in large-scale systems,” 2021. URL: https://azure.microsoft.com

9. R. A. Rossi and N. K. Ahmed, “The network data repository with interactive graph analytics and visualization,” in Proceedings of the 2020 Web Conference, 2020. URL: https://networkrepository.com

10. V. P. Dwivedi and X. Bresson, “A Generalization of graph neural networks,” arXiv preprint arXiv:2006.06972, 2020. URL: https://arxiv.org/abs/2006.06972

11. W. Hamilton et al., “Inductive representation learning on large graphs,” in Proceedings of NIPS 2017, 2017. URL: https://arxiv.org/abs/1706.02216

12. T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” arXiv preprint arXiv:1609.02907, 2017. URL: https://arxiv.org/abs/1609.02907

13. P. Velickovic et al., “Graph attention networks,” arXiv preprint arXiv:1710.10903, 2018. URL: https://arxiv.org/abs/1710.10903

14. J. Devlin et al., “BERT: Pre-training of deep bidirectional transformers for language understanding,” in Proc. of NAACL 2019, 2019. URL: https://arxiv.org/abs/1810.04805

15. A. Vaswani et al., “Attention is all you need,” in Proceedings of NIPS 2017, 2017. URL: https://arxiv.org/abs/1706.03762

16. O. Rolik, V. Kolesnik, and D. Halushko, “Decomposition-Compensation Method for IT Service Management.,” in: Kulczycki P., Kóczy L., Mesiar R., Kacprzyk J. (eds), Information Technology and Computational Physics. CITCEP 2016. Advances in Intelligent Systems and Computing, vol 462, Springer, Cham, 2017, pp. 89-107.