https://doi.org/10.31891/2219-9365-2026-85-53
УДК 004.8

ВОНСОВИЧ Богдан
Хмельницький національний університет
https://orcid.org/0009-0002-9746-5287
bodya.the.best.of.all@gmail.com
БАГРІЙ Руслан
Хмельницький національний університет
https://orcid.org/0000-0001-5219-1185
e-mail: bahriiro@khmnu.edu.ua
СКРИПНИК Тетяна
Хмельницький національний університет
https://orcid.org/0000-0002-8531-5348
e-mail: tkskripnik1970@gmail.com
ПАСІЧНИК Олександр
Хмельницький національний університет
https://orcid.org/0000-0002-8760-4688
e-mail: manziuk.e@khmnu.edu.ua

АВТОМАТИЗОВАНЕ ВИЯВЛЕННЯ ТА КЛАСИФІКАЦІЯ НЕОДНОЗНАЧНОСТЕЙ У ВИМОГАХ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІТ-ПРОЄКТІВ З ВИКОРИСТАННЯМ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ ТА RAG-ТЕХНОЛОГІЙ

Проблема забезпечення якості вимог на ранніх етапах життєвого циклу розробки набуває критичного значення для успіху програмних проєктів, адже наявність лінгвістичних та семантичних неоднозначностей у специфікаціях є однією з головних причин виникнення дефектів, перевитрат бюджету та зривів термінів реалізації. Ручне рецензування документації є надмірно трудомістким та суб'єктивним процесом, а традиційні автоматизовані інструменти на основі правил часто генерують високий рівень хибних спрацювань через нездатність розуміти контекст. Використання великих мовних моделей відкриває нові можливості для семантичного аналізу тексту, проте їх пряме застосування ускладнюється схильністю до галюцинацій та ігноруванням специфічної термінології проєкту.

У статті запропоновано гібридний метод виявлення неоднозначностей у вимогах до програмного забезпечення, що базується на інтеграції великих мовних моделей із технологією пошукового доповнення генерації (RAG). Запропонований підхід ґрунтується на двоагентній архітектурі, яка передбачає послідовну взаємодію агента-ідентифікатора для первинної фільтрації вимог та агента-класифікатора для їх глибокої семантичної типізації згідно зі стандартом IEEE 830. Метод охоплює такі етапи, як сегментація тексту, динамічне збагачення запитів контекстом із бази знань проєкту, застосування стратегії ланцюжка міркувань для генерації пояснень та структурний парсинг результатів.

Експерименти проведено на спеціалізованому датасеті з домену телекомунікацій, що містить 1983 вимоги. Отримані результати засвідчили перевагу розробленого гібридного методу на базі моделі Flan-T5-Large над базовими підходами (Zero-Shot та Few-Shot): метод забезпечує Precision 87%, повноту Recall 91% та F1-score 89%. Додатково підтверджено ефективність використання RAG для зниження кількості хибних спрацювань на вузькоспеціалізованій технічній лексиці та продемонстровано здатність системи надавати інтерпретовані пояснення виявлених дефектів. Результати доводять, що інтеграція контекстно-орієнтованих LLM-агентів істотно підвищує рівень автоматизації та надійності процесу аудиту вимог у сучасних середовищах розробки.

Ключові слова: вимоги до програмного забезпечення, неоднозначність, великі мовні моделі, RAG, Flan-T5, Chain-of-Thought, автоматизація перевірки.

VONSOVYCH Bohdan, BAHRII Ruslan, SKRYPNYK Tetiana, PASICHNYK Oleksandr
Khmelnytskyi National University

AUTOMATED DETECTION AND CLASSIFICATION OF AMBIGUITIES IN SOFTWARE REQUIREMENTS OF IT PROJECTS USING LARGE LANGUAGE MODELS AND RAG TECHNOLOGIES

The problem of ensuring the quality of requirements in the early stages of the development life cycle is of critical importance for the success of software projects, since the presence of linguistic and semantic ambiguities in specifications is one of the main causes of defects, budget overruns and disruptions in implementation deadlines. Manual review of documentation is an excessively laborious and subjective process, and traditional automated rule-based tools often generate a high level of false positives due to the inability to understand the context. The use of large language models opens up new opportunities for semantic text analysis, but their direct application is complicated by the tendency to hallucinations and ignoring specific project terminology.

The article proposes a hybrid method for detecting ambiguities in software requirements, which is based on the integration of large language models with the technology of search-addition generation (RAG). The proposed approach is based on a two-agent architecture, which involves the sequential interaction of an identifier agent for initial filtering of requirements and a classifier agent for their deep semantic typing according to the IEEE 830 standard. The method includes such stages as text segmentation, dynamic enrichment of queries with context from the project knowledge base, application of the Chain-of-Thought strategy for generating explanations, and structural parsing of results.

The experiments were conducted on a specialized dataset from the telecommunications domain containing 1983 requirements. The results obtained demonstrated the superiority of the developed hybrid method based on the Flan-T5-Large model

over the basic approaches (Zero-Shot and Few-Shot): the method provides Precision 87%, Recall completeness 91%, and F1-score 89%. Additionally, the effectiveness of using RAG to reduce the number of false positives on highly specialized technical vocabulary is confirmed and the system's ability to provide interpreted explanations of detected defects is demonstrated. The results prove that the integration of context-oriented LLM agents significantly increases the level of automation and reliability of the requirements audit process in modern development environments.

Keywords: software requirements, ambiguity, large language models, RAG, Flan-T5, Chain-of-Thought, verification automation

ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

У сучасних дослідженнях у сфері розробки програмного забезпечення проблема забезпечення якості вимог на ранніх етапах життєвого циклу набуває особливої актуальності. Вимоги виступають основним засобом комунікації між сторонами процесу розробки, тому їхня чіткість та однозначність є вирішальними для успіху проєкту [1, 2]. Наявність лінгвістичних або семантичних неоднозначностей у специфікаціях призводить до різного тлумачення завдань учасниками розробки, що спричиняє виникнення дефектів, перевитрати бюджету та необхідність суттєвих доопрацювань на пізніх стадіях [3, 4]. Аналітичні дані свідчать, що значна частина програмних збоїв зароджується саме через комунікаційні розриви під час фіксації вимог [5, 6].

Ручна перевірка документації залишається трудомістким, суб'єктивним та схильним до помилок процесом, який складно масштабувати. Існуючі автоматизовані інструменти, що базуються на лексико-синтаксичних правилах або класичних методах NLP (наприклад, QuARS), часто забезпечують лише поверхневий аналіз тексту, вони генерують високий рівень хибних спрацювань, оскільки не враховують глибокі семантичні залежності та специфічний контекст предметної області [7].

Останні досягнення у сфері штучного інтелекту, зокрема поява великих мовних моделей (LLM) та трансформерних архітектур, відкрили нові можливості для підвищення якості аналізу вимог [8, 9]. Великі мовні моделі здатні виконувати глибокий семантичний розбір, виявляти приховані неточності та навіть генерувати пояснення причин неоднозначності, використовуючи механізми міркування [10].

Залишається невирішеним питання адаптації універсальних мовних моделей до специфіки конкретних проєктів. Самі по собі LLM схильні до "галюцинацій" та ігнорування вузькоспеціалізованої термінології чи архітектурних обмежень, що знижує довіру до результатів їх роботи [11]. Відсутність доступу моделі до актуального глосарію та бази знань проєкту призводить до помилкової класифікації коректних технічних вимог як неоднозначних.

Таким чином, актуальною науково-практичною задачею є розроблення гібридного методу автоматизованого виявлення неоднозначностей, який поєднує аналітичну потужність великих мовних моделей із технологією пошукового доповнення генерації (RAG). Використання RAG дозволить забезпечити динамічне врахування контексту проєкту, підвищити точність класифікації та мінімізувати кількість хибних спрацювань порівняно з базовими методами.

АНАЛІЗ ДОСЛІДЖЕНЬ ТА ПУБЛІКАЦІЙ

Проблематика неоднозначності в технічній документації була вперше детально описана в фундаментальних дослідженнях, де це явище характеризується як стан, коли зафіксовані вимоги отримують відмінні, а іноді й суперечливі трактування різними учасниками життєвого циклу [12, 13]. Подальший розвиток цієї тематики продемонстрував, що основна причина криється у специфіці природної мови, яка проявляється через полісемію, синтаксичну варіативність та недостатню визначеність контексту, що створює так званий семантичний розрив між наміром автора та сприйняттям розробника [14, 15].

Сучасні огляди підкреслюють, що ручні методи рецензування є трудомісткими та суб'єктивними, тоді як традиційні автоматизовані інструменти (наприклад, QuARS) мають суттєві обмеження. Окремі дослідження акцентують увагу на тому, що синтаксичні підходи, які базуються на жорстких правилах та шаблонах, часто генерують високий рівень хибних спрацювань, оскільки оперують поверхневими структурами тексту і не здатні охопити глибинний семантичний зміст вимог [16].

Поява методів машинного навчання та використання векторних представлень стали важливим етапом еволюції аналізу вимог, оскільки дозволили перейти від пошуку ключових слів до оцінки семантичної близькості [17]. Подальші підходи, що використовують ембединги домен-специфічних мов, забезпечили певний рівень адаптивності, проте виявилися залежними від наявності великих розмічених корпусів даних, яких у сфері інженерії вимог критично бракує [18]. Додаткові дослідження підтверджують ефективність режимів Zero-shot та Few-shot в умовах дефіциту даних, демонструючи здатність моделей розрізняти семантичні відтінки без повноцінного донавчання [19].

Суттєвий прогрес у сфері виявлення неоднозначностей став можливим завдяки розвитку великих мовних моделей (LLM) та трансформерних архітектур. У низці сучасних робіт наголошується, що використання моделей сімейства GPT або T5 дозволяє змінити підхід від бінарної класифікації до глибокого аналітичного розбору з генерацією пояснень. Систематичні огляди останніх років демонструють формування

нових підходів, де LLM використовуються як інтелектуальні агенти, здатні до «розумного міркування» та врахування широкого лінгвістичного контексту [20].

Моделі на основі трансформерів відкрили можливість застосування передових технік, таких як стратегія ланцюжка міркувань (CoT - Chain-of-Thought), що дало змогу не лише виявляти дефекти, а й забезпечувати інтерпретованість результатів для інженерів. Подальші дослідження показали, що інтеграція зовнішніх баз знань через механізм RAG дозволяє суттєво знизити рівень галюцинацій та забезпечити фактологічну точність при роботі зі специфічною проєктною термінологією, змушуючи модель спиратися на надані докази [21].

Незважаючи на досягнутий прогрес, низка проблем залишається відкритою. Зокрема, актуальними залишаються питання адаптації універсальних LLM до вузькоспеціалізованих доменів без високовартісного донавчання, подолання проблеми «катастрофічного забування» контексту та забезпечення стабільності вихідних даних. Відсутність гібридних архітектур, які б поєднували гнучкість генеративних моделей із точністю детермінованих правил, обмежує промислове впровадження таких систем [22].

Отже, сучасні дослідження демонструють перехід від класичних NLP-методів [23,24].до контекстно-орієнтованих агентних систем на базі LLM. Це створює передумови для розроблення нових методів, що поєднують семантичну потужність нейромереж із надійністю перевірки фактів через RAG.

ФОРМУЛЮВАННЯ ЦІЛЕЙ СТАТТІ

Метою роботи є розроблення та експериментальна перевірка гібридного методу автоматизованого виявлення та класифікації неоднозначностей у вимогах до програмного забезпечення з використанням великих мовних моделей. Запропонований підхід спрямований на підвищення точності та пояснюваності аналізу завдяки двохетапній обробці та динамічному врахуванню контексту проєкту.

ВИКЛАД ОСНОВНОГО МАТЕРІАЛУ

Запропонований метод базується на використанні гібридної архітектури, що поєднує генеративні можливості великих мовних моделей із технологією пошукового доповнення (RAG). Його концепція ґрунтується на декомпозиції задачі аналізу вимог на два послідовні етапи: ідентифікацію потенційних проблем та їхню поглиблену семантичну класифікацію. Такий підхід дозволяє нівелювати основні недоліки прямих запитів до LLM – схильність до галюцинацій та ігнорування вузькоспеціалізованого контексту проєкту.

Схема методу включає такі основні етапи, як попередня обробка й сегментація тексту, ідентифікація неоднозначностей за допомогою RAG-фільтрації, а також фінальна класифікація з автоматичним формуванням пояснень. Кожен етап реалізується окремим програмним агентом, що забезпечує гнучкість налаштування та можливість незалежної оптимізації метрик повноти (Recall) та точності (Precision) . Загальну схему архітектури подано на рисунку 1.

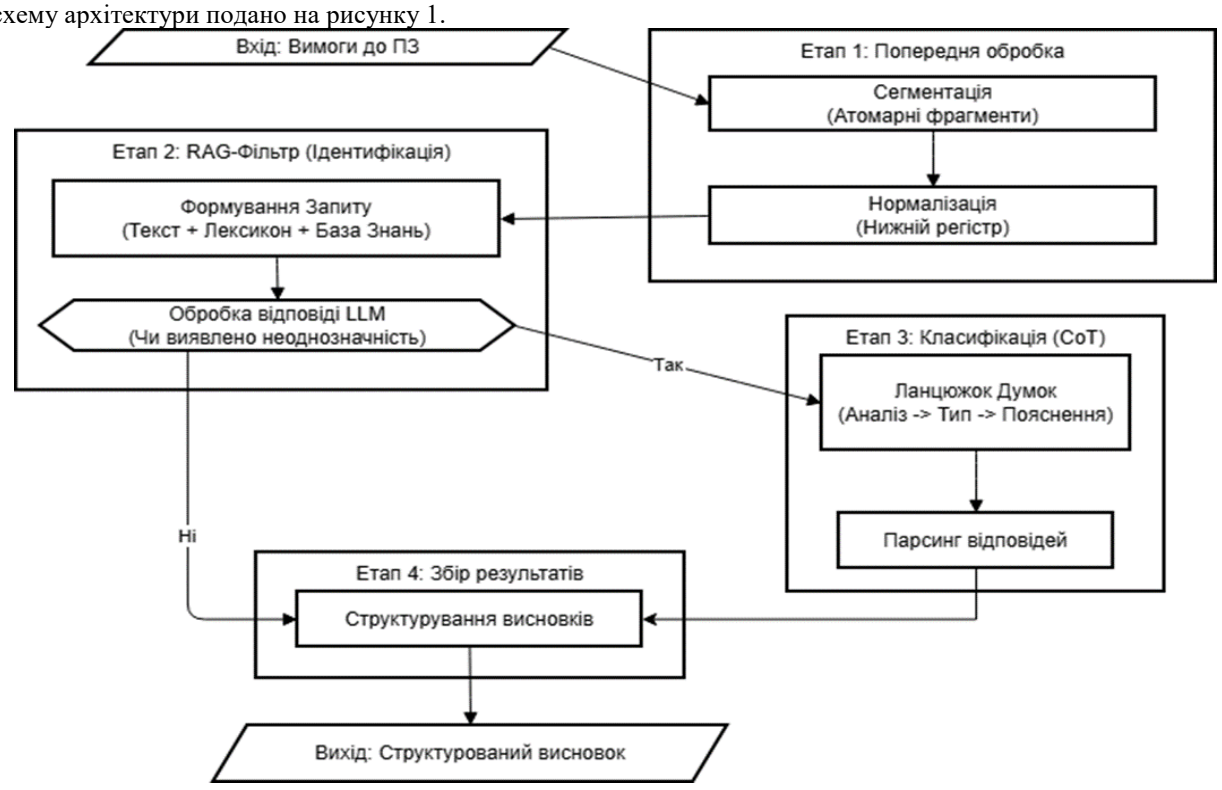

Рис. 1. Схема методу виявлення неоднозначностей у вимогах до програмного забезпечення

На етапі ідентифікації функціонує агент-ідентифікатор, завданням якого є первинна фільтрація вимог. Для цього формується композитний запит, що об'єднує текст вимоги, лексичні маркери (словник тригерів) та факти з бази знань проєкту (опис архітектури, глосарій) . Використання RAG дозволяє динамічно верифікувати технічні терміни, відсіюючи хибні спрацювання на абревіатурах чи специфічних назвах компонентів. Результатом етапу є оцінка впевненості моделі: якщо вона перевищує порогове значення, вимога передається на наступний рівень обробки.

Глибокий аналіз виконується Агентом-Класифікатором із застосуванням стратегії ланцюжка міркувань. Замість простої класифікації, модель послідовно виконує лінгвістичний розбір, зіставлення ознак із таксономією стандарту IEEE 830 та генерацію текстового пояснення причини дефекту, це дозволяє не лише виявити лексичні, синтаксичні чи референційні неоднозначності, а й надати інженеру аргументований висновок для їх виправлення.

АНАЛІЗ ЕФЕКТИВНОСТІ ЗАПРОПОНОВАНОГО МЕТОДУ

Метою експериментальної частини є кількісна й якісна оцінка ефективності запропонованого гібридного методу виявлення та класифікації неоднозначностей у вимогах до програмного забезпечення з використанням великих мовних моделей. Основна увага приділяється оцінюванню здатності методу точно ідентифікувати семантичні дефекти з урахуванням специфічного контексту проєкту, а також визначенню впливу технології RAG та стратегії ланцюжка міркувань на показники точності та повноти аналізу.

Дослідження проводилося на спеціалізованій підмножині датасету «Cornelius_2025_user_story_ambiguity_dataset» , що охоплює предметну область телекомунікацій. Набір даних містить 1983 вимоги, серед яких 890 (44,9%) мають підтверджені експертами неоднозначності, а 1093 (55,1%) є коректними. Вибір цього домену зумовлений високою насиченістю специфікацій складною технічною термінологією та абревіатурами, що дозволяє об'єктивно перевірити стійкість моделей до «галюцинацій» та хибних спрацювань.

Оцінювання виконувалося за метриками Precision (Точність), Recall (Повнота), F1-score (середнє гармонічне) та Classification Accuracy (точність мультикласової класифікації).

Одним із ключових чинників, що впливають на якість автоматизованого аналізу вимог, є наявність контекстної обізнаності – тобто врахування не лише лінгвістичних ознак тексту, а й описів архітектури системи, глосарію та бізнес-логіки. Класичні підходи (Zero-Shot) часто ігнорують ці залежності, сприймаючи вузькоспеціалізовані терміни як помилки. Щоб оцінити вплив запропонованих архітектурних рішень (RAG та агентна декомпозиція), було проведено порівняння трьох сценаріїв:

1. Zero-Shot – пряме використання LLM без додаткового контексту;
2. Few-Shot – використання LLM з наданням демонстраційних прикладів;
3. запропонований метод із використанням RAG, агентів та CoT.

Порівняльні результати ефективності методів наведено у таблиці 1.

Таблиця 1

Порівняльний аналіз ефективності методів виявлення неоднозначностей

Метод	Precision (Точність)	Recall (Повнота)	F1-score
Zero-Shot	0.62	0.54	0.58
Few-Shot	0.74	0.65	0.69
Запропонований метод	0.87	0.91	0.89

Як видно з таблиці, запропонований гібридний метод демонструє суттєву перевагу за всіма ключовими метриками. Значення Precision зросло з 0.62 (у Zero-Shot) до 0.87, що свідчить про ефективність механізму RAG у фільтрації хибних спрацювань на технічній лексиці. Показник Recall досяг рівня 0.91 (проти 0.54 у базовому сценарії), що підтверджує правильність налаштування агента-ідентифікатора як «широкого фільтра», здатного фіксувати найменші ознаки потенційних проблем.

Низькі значення метрик для підходу Zero-Shot (F1-score = 0.58) є очікуваними, оскільки ізольована від контексту модель схильна до гіперкорекції – помилкової класифікації коректних абревіатур (наприклад, назв серверів "S1", "GW1") як незрозумілих термінів. Впровадження Few-Shot підходу покращило результати (F1 = 0.69) завдяки адаптації моделі до формату задачі, проте лише повна архітектура з ін'єкцією знань дозволила досягти показника F1-score на рівні 0.89.

На наступному етапі дослідження було проаналізовано здатність методу розрізняти специфічні типи неоднозначностей згідно зі стандартом IEEE 830, це важливо для практичного застосування, оскільки різні типи дефектів вимагають різних стратегій виправлення. Точність класифікації для основних категорій неоднозначностей наведено у таблиці 2.

Таблиця 2

Точність класифікації за типами неоднозначностей

Тип неоднозначності (Клас)	Точність класифікації (Accuracy)	Частка у вибірці
Семантична (Semantic)	"84.8%"	Висока
Сфери дії (Scope)	"82.3%"	Середня
Виконавця (Actor)	"83.8%"	Середня
Приймання (Acceptance)	"77.0%"	Низька
Пріоритетів (Priority)	"68.8%"	Дуже низька

Результати показали, що метод забезпечує високу точність для найбільш поширених та складних категорій: семантичних (84,8%) та референційних (Actor – 83,8%) неоднозначностей, це підтверджує ефективність стратегії ланцюжка міркувань, яка дозволяє моделі виконувати глибокий синтаксичний розбір та встановлювати логічні зв'язки між елементами вимоги (наприклад, резолюцію анафори). Дещо нижчий результат для категорії пріоритетів (68,8%) пояснюється їхньою малою представленістю у навчальному контексті, що є типовим для незбалансованих реальних датасетів.

Отже, результати експериментів підтверджують, що інтеграція великих мовних моделей з контекстною базою знань та агентною архітектурою дозволяє вирішити проблему «семантичного розриву». Запропонований метод не лише виявляє дефекти з високою точністю (0.87), а й забезпечує їх коректну типізацію та інтерпретацію, що створює підґрунтя для його ефективного використання в промислових процесах аудиту вимог.

ВИСНОВКИ З ДАНОГО ДОСЛІДЖЕННЯ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ РОЗВІДОК У ДАНОМУ НАПРЯМІ

У ході проведеного дослідження було розроблено, реалізовано та експериментально перевірено гібридний метод автоматизованого виявлення та класифікації неоднозначностей у вимогах до програмного забезпечення з використанням великих мовних моделей. Основою підходу стало поєднання генеративних можливостей LLM із технологією RAG, що дозволило забезпечити контекстну орієнтованість аналізу. Модель Flan-T5-Large у поєднанні з агентною архітектурою продемонструвала здатність не лише фіксувати лінгвістичні дефекти, а й верифікувати технічну термінологію через базу знань проєкту, ефективно долаючи проблему «галюцинацій» та ігнорування контексту.

Експериментальні результати підтвердили суттєву перевагу запропонованого гібридного методу над базовими підходами. Порівняння зі сценаріями Zero-Shot та Few-Shot показало, що розроблений метод забезпечує найкращі показники: Precision досягає 0.87, Recall – 0.91, а F1-score – 0.89, значно перевищуючи результати базового використання LLM (F1-score 0.58 для Zero-Shot та 0.69 для Few-Shot). Значення мультикласової точності (0.83) свідчить про коректну типізацію виявлених дефектів згідно зі стандартом IEEE 830 та здатність системи надавати релевантні пояснення.

Особливе значення має впровадження механізму RAG та двохетапної обробки даних. Додавання динамічного контексту проєкту дозволило мінімізувати кількість хибних спрацювань на специфічній технічній лексиці, що стало ключовим фактором зростання точності з 0.62 до 0.87 порівняно з базовим сценарієм. Це підтверджує, що ефективний аналіз вимог потребує врахування не лише універсальних лінгвістичних правил, а й архітектурних обмежень та глосарію конкретної предметної області.

Проведений аналіз ефективності стратегії ланцюжка міркувань засвідчив її критичну роль у вирішенні складних семантичних задач. Використання CoT дозволило досягти високої точності розпізнавання для семантичних (84,8%) та референційних (83,8%) неоднозначностей, забезпечивши глибоку інтерпретацію зв'язків між елементами вимог. Незважаючи на збільшення середнього часу обробки до 2,6 с, такий підхід забезпечує необхідну аргументацію рішень, перетворюючи модель на «віртуального експерта».

Загалом результати роботи підтверджують доцільність застосування гібридних архітектур на базі LLM у задачах аудиту вимог. Запропонований метод забезпечує високу точність та об'єктивність аналізу завдяки інтеграції контекстних знань, зменшує навантаження на аналітиків шляхом автоматизації рутинних перевірок і створює підґрунтя для впровадження інтелектуальних асистентів у процеси розробки програмного забезпечення.

Література

1. Luminous Men. The Importance of Clear Software Requirements. Luminousmen.com. 2024. URL: https://luminousmen.com/post/the-importance-of-clear-software-requirements/
2. Requiment. Why Requirements Management Is So Important. Requiment. 2024. URL: https://www.requiment.com/the-importance-of-requirements-management/
3. Burkin V. Mitigating Risks in Software Development through Effective Requirements Engineering. arXiv. 2023. URL: https://arxiv.org/abs/2305.05800
4. Dale-Jones R. The Importance of Clear Requirements. SpireSpark International. 2017. URL: https://spirespark.com/spirespark-blog/2017/4/26/the-importance-of-clear-requirements

5. Beta Breakers. Software Survival in 2024: Understanding 2023 Project Failure Statistics and the Role of Quality Assurance. Betabreakers.com. 2024. URL: https://www.betabreakers.com/blog/software-survival-in-2024-understanding-2023-project-failure-statistics-and-the-role-of-quality-assurancet/

6. Jama Software. Why Investing in Requirements Management Software Makes Business Sense During an Economic Downturn. Jama Software. URL: https://www.jamasoftware.com/requirements-management-guide/requirements-management-tools-and-software/why-investing-in-rm-software-makes-good-business-sense/

7. Bakar N. A. A., et al. "Ambiguity Detection and Improvement for Malay Requirements Specification: A Systematic Review." International Journal on Advanced Science, Engineering and Information Technology, Vol. 13, No. 6, 2023. URL: http://ijaseit.insightsociety.org/index.php/ijaseit/article/view/18535

8. Zhao L., Alhoshan W., Ferrari A., Letsholo K. J., Ajagbe M. A., Chioasca E.-V., Batista-Navarro R. T. Natural Language Processing for Requirements Engineering: A Systematic Mapping Study. ACM Computing Surveys. 2021. Vol. 54. No. 3. P. 1–41. URL: https://dl.acm.org/doi/10.1145/3444689

9. Hou X., Zhao Y., Liu Y., Yang Z., Wang K., Li L., Luo X., Lo D., Grundy J., Wang H. Large Language Models for Software Engineering: A Systematic Literature Review. ACM Transactions on Software Engineering and Methodology. 2024. Vol. 33. No. 7. Article 206. P. 1–58. URL: https://dl.acm.org/doi/10.1145/3695988

10. Bashir S., Ferrari A., Khan M. A., Strandberg P. E., Haider Z., Saadatmand M., Bohlin M. Requirements Ambiguity Detection and Explanation with LLMs: An Industrial Study. Proceedings of the 41st International Conference on Software Maintenance and Evolution (ICSME). 2025. P. 620–631. URL: https://www.ipr.mdu.se/publications/7221-Requirements_Ambiguity_Detection_and_Explanation_with_LLMs__An_Industrial_Study

11. Ghosh B. Advanced Prompt Engineering for Reducing Hallucination. Medium. 2024. URL: https://medium.com/@bijit211987/advanced-prompt-engineering-for-reducing-hallucination-bb2c8ce62fc6

12. Berry D.M., Kamsties E. Ambiguity in Requirements Specification. Perspectives on Software Requirements. Springer. 2003. URL: https://link.springer.com/chapter/10.1007/978-1-4615-0465-8_2

13. Kamsties E., Berry D.M., Paech B. Detecting Ambiguities in Requirements Documents Using Inspections. Proceedings of the First Workshop on Inspection in Software Engineering (WISE'01). 2001. URL: https://cs.uwaterloo.ca/~dberry/FTP_SITE/reprints.journals.conferences/KamstiesBerryPaech2001DetectingAmbiguity.pdf

14. Gervasi V., Ferrari A., Zowghi D., Spoletini P. Ambiguity in Requirements Engineering: Towards a Unifying Framework // Lecture Notes in Computer Science. – 2019. – P. 191–210. https://doi.org/10.1007/978-3-030-30985-5_12

15. Kaur J., Kaur A. An Analysis of Ambiguity Detection Techniques for Software Requirements Specification (SRS). International Journal of Computer Sciences and Engineering. 2018. Vol. 6. No. 12. P. 103–108. URL: https://www.researchgate.net/publication/326730868_An_Analysis_of_Ambiguity_Detection_Techniques_for_Software_Requirements_Specification_SRS

16. Fantechi A., Gnesi S., Semini L. Rule-based NLP vs ChatGPT in Ambiguity Detection, a Preliminary Study. CEUR Workshop Proceedings. 2023. Vol. 3378. URL: https://ceur-ws.org/Vol-3378/NLP4RE-paper1.pdf

17. Ezzini S., Abualhaija S., Arora C., Sabetzadeh M. TAPHSIR: Towards AnaPHoric Ambiguity Detection and ReSolution In Requirements. arXiv. 2022. URL: https://arxiv.org/abs/2206.10227

18. Ferrari A., Bano M., Zowghi D., Gervasi V. An NLP approach for cross-domain ambiguity detection in requirements engineering. Software and Systems Modeling. 2019. Vol. 18. P. 1799–1812. URL: https://iris.cnr.it/handle/20.500.14243/386974

19. Fazelnia M., Koscinski V., Herzog S., Mirakhorli M. Lessons from the Use of Natural Language Inference (NLI) in Requirements Engineering Tasks. arXiv. 2024. URL: https://arxiv.org/abs/2405.05135

20. Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J.-Y., Wen, J.-R. A Survey of Large Language Models. arXiv. 2023. URL: https://arxiv.org/abs/2303.18223

21. Murugan T. Design Principles of Chain of Thoughts (CoT) Prompt Engineering in the Context of GenAI. LinkedIn. 2025. URL: https://www.linkedin.com/pulse/design-principles-chain-thoughts-cot-prompt-context-genai-murugan-qijic

22. Patronus AI. Advanced Prompt Engineering Techniques: Examples & Best Practices. Patronus.ai. URL: https://www.patronus.ai/llm-testing/advanced-prompt-engineering-techniques.

23. An adaptive approach to detecting fake news based on generalized text features(Conference Paper) Shupta, A., Barmak, O., Wierzbicki, A., Skrypnyk, T. // 7th International Conference on Computational Linguistics and Intelligent Systems. Volume I: Machine Learning Workshop, CoLInS 2023; Kharkiv; Ukraine; 20 April 2023 до 21 April 2023; Код 188444 // CEUR Workshop Proceedings Volume 3387, 2023, Pages 300-310

24. Manziuk E., Barmak O., Krak I., Mazurets O., Skrypnyk T. Formal Model of Trustworthy Artificial Intelligence Based on Standardization. / CEUR Workshop Proceedings, Khmelnytskyi, Ukraine, CEUR, March 24, 2021. Pp. 190–197.

References

1. Luminous Men. The Importance of Clear Software Requirements. Luminousmen.com. 2024. URL: https://luminousmen.com/post/the-importance-of-clear-software-requirements/
2. Requiment. Why Requirements Management Is So Important. Requiment. 2024. URL: https://www.requiment.com/the-importance-of-requirements-management/
3. Burkin V. Mitigating Risks in Software Development through Effective Requirements Engineering. arXiv. 2023. URL: https://arxiv.org/abs/2305.05800
4. Dale-Jones R. The Importance of Clear Requirements. SpireSpark International. 2017. URL: https://spirespark.com/spirespark-blog/2017/4/26/the-importance-of-clear-requirements
5. Beta Breakers. Software Survival in 2024: Understanding 2023 Project Failure Statistics and the Role of Quality Assurance. Betabreakers.com. 2024. URL: https://www.betabreakers.com/blog/software-survival-in-2024-understanding-2023-project-failure-statistics-and-the-role-of-quality-assurancet/
6. Jama Software. Why Investing in Requirements Management Software Makes Business Sense During an Economic Downturn. Jama Software. URL: https://www.jamasoftware.com/requirements-management-guide/requirements-management-tools-and-software/why-investing-in-rm-software-makes-good-business-sense/
7. Bakar N. A. A., et al. "Ambiguity Detection and Improvement for Malay Requirements Specification: A Systematic Review." International Journal on Advanced Science, Engineering and Information Technology, Vol. 13, No. 6, 2023. URL: http://ijaseit.insightsociety.org/index.php/ijaseit/article/view/18535
8. Zhao L., Alhoshan W., Ferrari A., Letsholo K. J., Ajagbe M. A., Chioasca E.-V., Batista-Navarro R. T. Natural Language Processing for Requirements Engineering: A Systematic Mapping Study. ACM Computing Surveys. 2021. Vol. 54. No. 3. P. 1–41. URL: https://dl.acm.org/doi/10.1145/3444689
9. Hou X., Zhao Y., Liu Y., Yang Z., Wang K., Li L., Luo X., Lo D., Grundy J., Wang H. Large Language Models for Software Engineering: A Systematic Literature Review. ACM Transactions on Software Engineering and Methodology. 2024. Vol. 33. No. 7. Article 206. P. 1–58. URL: https://dl.acm.org/doi/10.1145/3695988
10. Bashir S., Ferrari A., Khan M. A., Strandberg P. E., Haider Z., Saadatmand M., Bohlin M. Requirements Ambiguity Detection and Explanation with LLMs: An Industrial Study. Proceedings of the 41st International Conference on Software Maintenance and Evolution (ICSME). 2025. P. 620–631. URL: https://www.ipr.mdu.se/publications/7221-Requirements_Ambiguity_Detection_and_Explanation_with_LLMs__An_Industrial_Study
11. Ghosh B. Advanced Prompt Engineering for Reducing Hallucination. Medium. 2024. URL: https://medium.com/@bijit211987/advanced-prompt-engineering-for-reducing-hallucination-bb2c8ce62fc6
12. Berry D.M., Kamsties E. Ambiguity in Requirements Specification. Perspectives on Software Requirements. Springer. 2003. URL: https://link.springer.com/chapter/10.1007/978-1-4615-0465-8_2
13. Kamsties E., Berry D.M., Paech B. Detecting Ambiguities in Requirements Documents Using Inspections. Proceedings of the First Workshop on Inspection in Software Engineering (WISE'01). 2001. URL: https://cs.uwaterloo.ca/~dberry/FTP_SITE/reprints.journals.conferences/KamstiesBerryPaech2001DetectingAmbiguity.pdf
14. Gervasi V., Ferrari A., Zowghi D., Spoletini P. Ambiguity in Requirements Engineering: Towards a Unifying Framework // Lecture Notes in Computer Science. – 2019. – P. 191–210. https://doi.org/10.1007/978-3-030-30985-5_12
15. Kaur J., Kaur A. An Analysis of Ambiguity Detection Techniques for Software Requirements Specification (SRS). International Journal of Computer Sciences and Engineering. 2018. Vol. 6. No. 12. P. 103–108. URL: https://www.researchgate.net/publication/326730868_An_Analysis_of_Ambiguity_Detection_Techniques_for_Software_Requirements_Specification_SRS
16. Fantechi A., Gnesi S., Semini L. Rule-based NLP vs ChatGPT in Ambiguity Detection, a Preliminary Study. CEUR Workshop Proceedings. 2023. Vol. 3378. URL: https://ceur-ws.org/Vol-3378/NLP4RE-paper1.pdf
17. Ezzini S., Abualhaija S., Arora C., Sabetzadeh M. TAPHSIR: Towards AnaPHoric Ambiguity Detection and ReSolution In Requirements. arXiv. 2022. URL: https://arxiv.org/abs/2206.10227
18. Ferrari A., Bano M., Zowghi D., Gervasi V. An NLP approach for cross-domain ambiguity detection in requirements engineering. Software and Systems Modeling. 2019. Vol. 18. P. 1799–1812. URL: https://iris.cnr.it/handle/20.500.14243/386974
19. Fazelnia M., Koscinski V., Herzog S., Mirakhorli M. Lessons from the Use of Natural Language Inference (NLI) in Requirements Engineering Tasks. arXiv. 2024. URL: https://arxiv.org/abs/2405.05135
20. Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J.-Y., Wen, J.-R. A Survey of Large Language Models. arXiv. 2023. URL: https://arxiv.org/abs/2303.18223
21. Murugan T. Design Principles of Chain of Thoughts (CoT) Prompt Engineering in the Context of GenAI. LinkedIn. 2025. URL: https://www.linkedin.com/pulse/design-principles-chain-thoughts-cot-prompt-context-genai-murugan-qijic
22. Patronus AI. Advanced Prompt Engineering Techniques: Examples & Best Practices. Patronus.ai. URL: https://www.patronus.ai/llm-testing/advanced-prompt-engineering-techniques.
23. An adaptive approach to detecting fake news based on generalized text features(Conference Paper) Shupta, A., Barmak, O., Wierzbicki, A., Skrypnyk, T. // 7th International Conference on Computational Linguistics and Intelligent Systems. Volume I: Machine Learning Workshop, CoLInS 2023; Kharkiv; Ukraine; 20 April 2023 до 21 April 2023; Код 188444 // CEUR Workshop Proceedings Volume 3387, 2023, Pages 300-310
24. Manziuk E., Barmak O., Krak I., Mazurets O., Skrypnyk T. Formal Model of Trustworthy Artificial Intelligence Based on Standardization. / CEUR Workshop Proceedings, Khmelnytskyi, Ukraine, CEUR, March 24, 2021. Pp. 190–197.