DOI: https://doi.org/10.25140/2411-5363-2025-4(42)-233-244
УДК 004.8:004.4

Олексій Валерійович Максимов[1], Дмитро Едуардович Лисенко[2]

[1]аспірант, Національний університет «Чернігівська політехніка» (Чернігів, Україна)
E-mail: maksimov98gmy@gmail.com. **ORCID:** https://orcid.org/0009-0008-4257-6549
[2]доктор технічних наук, професор кафедри інформаційних та комп'ютерних систем, Національний університет «Чернігівська політехніка» (Чернігів, Україна)
E-mail: lysenko.d@stu.cn.ua. **ORCID:** https://orcid.org/0000-0001-6870-6120

ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕТОДІВ ВИКОРИСТАННЯ ПРИРОДНОМОВНОЇ ОБРОБКИ ДЛЯ АВТОМАТИЗОВАНОГО ГЕНЕРУВАННЯ ТЕСТ-КЕЙСІВ

У статті здійснено порівняльний аналіз основних науково-методичних підходів до використання методів природномовної обробки (NLP) для автоматизованого генерування тест-кейсів. Розглянуто правило-орієнтовані методи, підходи на основі машинного навчання та трансформерні моделі. Подано узагальнену характеристику принципів роботи кожної групи методів, особливості їх застосування та вимоги до вхідних даних. Стаття також описує структуру процесу автоматизації, охоплює питання попередньої обробки вимог, формування тестових сценаріїв і критеріїв порівняння підходів.

Ключові слова: *природномовна обробка; автоматизоване тестування; генерація тест-кейсів; машинне навчання; оптимізація; якість програмного забезпечення.*
Рис.: 3. Табл.: 1. Бібл.: 30.

Актуальність теми дослідження. У сучасному світі швидкого розвитку програмного забезпечення якості продуктів є критично важливим завданням. Традиційні методи створення тест-кейсів, які базуються на ручній праці інженерів з тестування, є трудомісткими, схильними до помилок та не можуть задовольнити потреби agile-методології розробки. Як наслідок, збільшується ризик появи критичних помилок у продакшні та зростає вартість підтримки. Автоматизація процесу побудови тестів на основі природномовних вимог дозволяє зменшити навантаження на команди тестування та забезпечити швидший вихід продукту на ринок. Автоматизація процесу побудови тестів на основі природномовних вимог є актуальною для широкого спектра програмних систем – від веборієнтованих рішень і хмарних сервісів до вбудованих та промислових платформ. Зокрема, у розробці вебдодатків автоматичне формування сценаріїв перевірки користувацьких історій дає можливість скоротити час підготовки релізу без втрати якості.

Постановка проблеми. У сучасних програмних системах, включно з вебдодатками, вбудованими, промисловими та доменно-специфічними рішеннями, існує низка проблем, які роблять процес створення тест-кейсів на основі текстових вимог складним та ресурсомістким:

- великий обсяг неоднорідних вимог – специфікації програмних систем часто формулюються у вигляді user stories, функціональних вимог або технічної документації з природномовними описами; перетворення цих вимог у формалізовані тест-кейси вимагає значних зусиль тестувальників;

- динамічність інтерфейсів – часті зміни UI/UX у web- та мобільних застосунках, а також еволюція функціональності у вбудованих і промислових системах потребують постійного оновлення тестів, що призводить до суттєвих витрат часу;

- складність перевірки інтеграцій – сучасні системи активно взаємодіють з API, сторонніми сервісами, польовими пристроями та базами даних; ручне створення тестів для таких сценаріїв є довготривалим і схильним до помилок;

- вимоги до кросбраузерності та адаптивності – тест-кейси мають враховувати варіативність конфігурацій (операційні системи, браузери, апаратні платформи), що ще збільшує їх кількість.

Для вебсистем ці виклики детально аналізуються, зокрема, в сучасних дослідженнях тестування вебплатформ [1], а для вбудованих, супутникових та IoT-рішень – у роботах, присвячених автоматизованому генеруванню тестів на основі текстових специфікацій [3–7]. Усе

це створює потребу в технологіях, які можуть автоматично трансформувати природномовні вимоги у структуровані та придатні до виконання тест-кейси. Використання NLP та методів машинного навчання у цьому процесі дозволяє мінімізувати людський фактор, підвищити ефективність тестування та забезпечити швидке оновлення тестів у відповідь на зміни у програмній системі [2–7].

Аналіз останніх досліджень і публікацій. У процесі підготовки оглядової статті було проведено цілеспрямований пошук наукових публікацій, присвячених використанню методів природномовної обробки (NLP) для автоматизованого генерування тест-кейсів з текстових вимог. Пошук здійснювався у таких наукометричних базах та електронних бібліотеках: IEEE Xplore, ACM Digital Library, Scopus, SpringerLink та Google Scholar. До розгляду бралися публікації, опубліковані в період 2010–2025 рр., з акцентом на роботах останніх років, у яких відображено розвиток трансформерних моделей і великих мовних моделей (LLM).

Пошукові запити формувалися на основі комбінацій ключових слів англійською мовою, зокрема: “test case generation”, “automatic test case generation”, “test case derivation from requirements”, “natural language requirements”, “requirements-based testing”, “NLP for software testing”, “rule-based test generation”, “machine learning test generation”, “transformer-based test generation”, “large language models for testing” тощо. Для уточнення результатів використовувалися логічні оператори AND, OR та фільтри за роком публікації й типом документа (журнал / конференція).

До підсумкового набору були включені лише ті роботи, які відповідали таким критеріям включення:

- публікація у рецензованих наукових журналах або матеріалах наукових конференцій;
- наявність явного фокуса на генеруванні або виведенні тест-кейсів / тестових сценаріїв з природномовних вимог, специфікацій або інших текстових артефактів;
- використання методів NLP (правил, статистичних/машинно-навчальних моделей, трансформерів або LLM) як ключового елемента підходу;
- доступність повного тексту англійською мовою (у поодиноких випадках – іншими мовами за наявності англомовної анотації).

Критерії виключення охоплювали:

- нерецензовані джерела (блоги, технічні звіти без рецензування тощо);
- роботи, у яких NLP застосовується до текстових вимог, але не для підтримки тест-дизайну (наприклад, лише класифікація вимог чи аналіз якості без побудови тестів);
- дослідження, що стосуються автоматизованого генерування тестів, але базуються лише на моделях коду, моделях станів чи UML-діаграмах без використання текстових вимог;
- дублікати публікацій або розширені версії, де основні результати вже представлені в іншій роботі, врахованій в огляді.

На основі відібраних робіт було виконано подальшу якісну класифікацію підходів за трьома основними групами: правило-орієнтовані методи, методи на основі машинного навчання та NER та підходи, що використовують трансформерні моделі та LLM. Саме ці три групи підходів надалі порівнюються за типом вхідних артефактів, застосованими NLP-техніками, видами отриманих тестів і доступними метриками якості.

Останнім часом спостерігається активне дослідження застосування NLP-технологій у сфері програмного тестування. Систематичний огляд літератури показує, що в останні роки з’явилася обмежена, але зростаюча кількість робіт, які пропонують різні NLP-техніки, інструменти та фреймворки для автоматизованої генерації тест-кейсів [2; 14-18, 27].

Gröpler та співавтори представили підхід на основі правило-орієнтованої формалізації вимог з використанням NLP для автоматичної генерації тест-кейсів [3]. Їхній метод демонструє можливість перетворення текстових вимог у формальні артефакти тестування з використанням послідовності кроків: аналіз природномовних специфікацій, побудова проміжних моделей та автоматизований вивід тестових сценаріїв.

Tufano та співавтори розробили ATHENATEST – підхід, що використовує трансформерні моделі для генерації юніт-тестів шляхом навчання на реальних тестах, написаних розробниками [4]. Результати експериментів показують, що ATHENATEST здатна генерувати десятки тисяч коректних тестів і досягає покриття коду, порівнянного з EvoSuite, переважаючи як EvoSuite, так і GPT-3 за експертними оцінками читабельності, зрозумілості та ефективності тестування.

Medeshetty та співавтори порівняли правило-орієнтовані методи з методами розпізнавання іменованих сутностей (NER) для створення специфікацій тест-кейсів для високопродуктивних електронних блоків управління (ECU) [5]. Їхні результати показали, що правило-орієнтований підхід забезпечує високу точність і суттєве скорочення часу підготовки тестів, тоді як NER-базовані моделі мають потенціал, але потребують додаткового налаштування й розширення навчальних даних.

Окремі роботи присвячені використанню великих мовних моделей (LLM) для генерації тестів із текстових специфікацій, наприклад у доменах супутникових систем, інтелектуальних вимірювальних терміналів тощо [6–11]. Робота Albdeiwi демонструє застосування LLM до задачі генерації тест-кейсів з природномовних вимог та містить емпіричну оцінку як автоматичними метриками, так і з залученням експертів [12].

Разом із тим огляди підкреслюють, що комплексне порівняння різних підходів на спільному наборі критеріїв залишається недостатньо опрацьованим [2, 10, 14-18].

Виділення недосліджених частин загальної проблеми. Попри активні дослідження в області застосування NLP для генерації тест-кейсів, відсутній комплексний порівняльний аналіз різних підходів з погляду їхньої ефективності, точності та практичності застосування. Більшість робіт зосереджуються на окремих методах – правило-орієнтованих, машинного навчання або трансформерних – без систематичного порівняння їхніх сильних і слабких сторін в єдиному аналітичному полі. Недостатньо також вивчені питання оптимізації часу генерації тест-кейсів та їхньої якості в контексті різних типів проєктів і вимог [7–11].

Додаткові систематичні огляди та мапінг-дослідження доповнюють картину сучасних підходів до requirements-based і NLP-орієнтованої генерації тестів. Mustafa та співавтори виконали огляд методів автоматичного створення тест-кейсів із вимог, показавши домінування моделей на основі UML-діаграм і use case специфікацій [14]. Farooq і Tahreem узагальнили requirement-based підходи та запропонували таксономію технік автоматизації тестування [15]. Garousi та співавтори систематизували застосування NLP у програмному тестуванні загалом і виділили групи задач, для яких такі методи є найперспективнішими [16]. Окремі огляди зосереджуються саме на генерації тестів за допомогою NLP-методів [17, 18], а також на використанні NLP в інженерії вимог, що задає основу для подальшої автоматизації тест-дизайну [27].

Мета і завдання дослідження. Метою статті є проведення комплексного порівняльного аналізу трьох основних підходів до використання природномовної обробки для автоматизованого генерування тест-кейсів:

- правило-орієнтованих методів;
- підходів на основі машинного навчання з розпізнаванням іменованих сутностей;
- методів, що використовують трансформерні моделі та великі мовні моделі.

Основними завданнями дослідження є:

- описати методологічні основи кожного підходу, включаючи вимоги до вхідних даних і типові етапи обробки;
- узагальнити експериментальні результати, наведені в ключових роботах, із зазначенням використаних метрик та обмежень;
- сформулювати критерії вибору підходу залежно від характеристик проєкту (стабільність вимог, обсяг даних, доступні ресурси);

- надати узагальнену порівняльну таблицю, яка відображає концептуальні відмінності методів.

Виклад основного матеріалу.

Правило-орієнтовані методи.

Правило-орієнтовані методи базуються на попередньо визначених лінгвістичних правилах та техніках зіставлення шаблонів для ідентифікації й отримання релевантної інформації з написаних вимог [3; 5; 19–23, 26]. Ці методи включають операції з рядками для організації та трансформації текстових даних, зіставлення шаблонів через регулярні вирази для виявлення визначених патернів, та евристичні правила, що відображають специфічні структури в тексті на основі попередньо визначених правил.

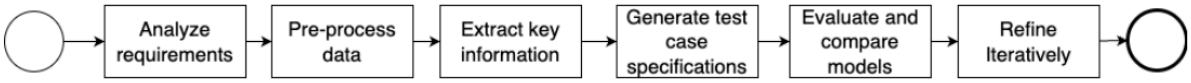

Рис. 1. Фази правило-орієнтованих методів

Джерело: [5].

На рисунку 1 зображено фази підходу розробленого Medeshetty та співавторів. У фазі аналізу вимог підхід зосереджується на вилученні умовно-дійових тверджень із фінальних розділів документів елементів функцій. На етапі попередньої обробки дані очищуються, нормалізуються та перетворюються на кортежі «умова–дія» з використанням GUI-автоматизації, регулярних виразів та POS-тегування. Далі, у фазі вилучення ключової інформації, застосовуються методи NER і правила на основі семантики для визначення назв сигналів, їхніх значень та очікуваних дій, після чого результати структуруються у вигляді словників і переносяться до датафрейму. На етапі формування сценаріїв тестування ці структуровані дані впорядковуються за допомогою шаблону, що охоплює всі релевантні комбінації «сигнал–значення». На кінцевій фазі специфікації тестів сформовані сценарії перетворюються на повноцінні тест-кейси з унікальними ідентифікаторами, початковими умовами, цільовим сигналом і очікуваним результатом, які потім експортуються для подальшої перевірки.

Дослідження Medeshetty та співавторів показало, що правило-орієнтовані методи досягають 95 % точності для простіших вимог з одиничними сигналами [5]. Основними перевагами цього підходу є:

- висока точність для структурованих та передбачуваних типів вимог;
- можливість детального контролю над перетворенням вимог у тест-кейси;
- повторюваність і прозорість результатів завдяки явним правилам;
- ефективність у випадках, коли формат документації стабільний.

Разом із тим правило-орієнтовані підходи мають обмеження у випадку складних або слабо структурованих вимог. Вони потребують постійного оновлення набору правил у разі змін формату документації або розширення предметної області, а також демонструють зниження точності на складних багатосигнальних вимогах [3].

Історично правило-орієнтовані підходи застосовувалися в низці промислових та наукових проєктів. Інструмент Litmus генерує тест-кейси з функціональних вимог природною мовою, спираючись на заздалегідь визначені патерни формулювань [19]. Santiago Júnior та Vijaykumar продемонстрували можливість побудови модельних тестів для програмного забезпечення космічних застосунків, автоматизуючи перехід від текстових вимог до формальних моделей [20]. Підхід NAT2TEST(SCR) показує, як структуровані природномовні специфікації можуть бути трансформовані в SCR-моделі з подальшою генерацією тестів [21]. Інші роботи пропонують перетворення вимог у сценарії на основі мереж Петрі [22] або кольорових мереж Петрі [23], а також виділення типових патернів

тестових сценаріїв із тексту [26]. У сукупності ці дослідження підкреслюють, що rule-based методи добре працюють у стабільних доменах із чітко регламентованими формулюваннями вимог.

Також, варта уваги робота Gröpler та співавторів, яка також належить до rule-based підходів, зокрема до підтипу синтаксично орієнтованих методів, які використовують результати залежнісного аналізу, але не застосовують моделей машинного навчання або трансформерів.

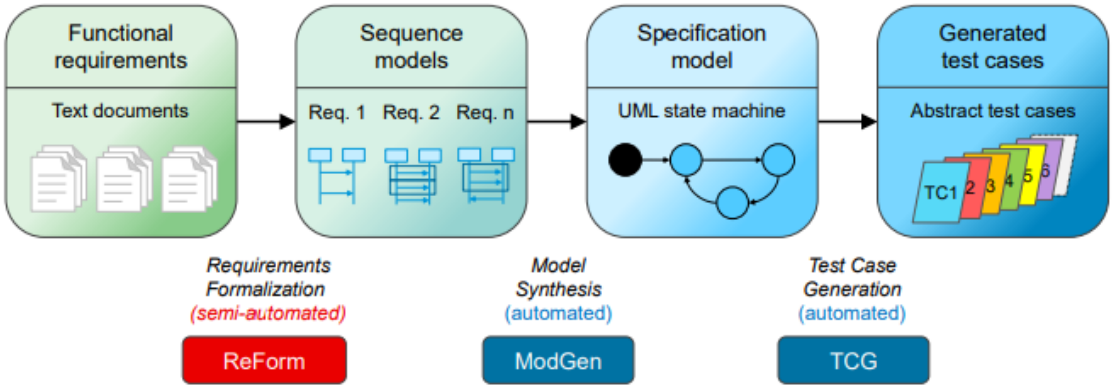

Рис. 2. Інструментарій для генерації тестових випадків на основі вимог
Джерело: [3].

У дослідженні застосовано rule-based підхід, який використовує інформацію, отриману зі стандартного конвеєра оброблення природної мови [3]. На рисунку 2 показано загальну послідовність інструментального ланцюжка, у якому природномовні вимоги проходять напівавтоматичну формалізацію, після чого на їх основі автоматично синтезується модель та генеруються тестові випадки. Схема демонструє три основні етапи: формалізацію вимог, синтез моделі та генерацію тестів. На першому етапі текстові вимоги проходять лінгвістичне передоброблення за допомогою бібліотеки spaCy, включно з токенізацією, лематизацією, частиномовним тегуванням та синтаксичним розбором залежностей. Подальші етапи алгоритму базуються на правилах і послідовно виконують розв'язання займенників, декомпозицію складних речень на елементарні кластери та виявлення ключових синтаксичних сутностей (дій, суб'єктів, об'єктів, порівнянь). На основі цих синтаксичних одиниць визначаються семантичні сутності, необхідні для побудови моделей взаємодії – актори, компоненти, сигнали, атрибути та стани. Після цього алгоритм автоматично формує текстове представлення вимог у вигляді елементів IRDL, яке може бути візуалізоване як UML sequence diagram. Згенеровані моделі потім використовуються для синтезу UML state machine та автоматичного генерування тестових сценаріїв. Загалом метод комбінує лінгвістичний аналіз зі спеціально визначеними правилами перетворення, забезпечуючи напівавтоматичну формалізацію природномовних вимог без використання методів машинного навчання.

Методи машинного навчання з розпізнаванням іменованих сутностей.

Підходи на основі машинного навчання використовують алгоритми розпізнавання іменованих сутностей (NER) для автоматичного виявлення та класифікації ключових елементів у природномовних вимогах [8–10]. NER-моделі навчаються на анотованих корпусах, де кожен фрагмент тексту позначений відповідною роллю (сигнал, значення, дія, обмеження тощо).

У дослідженні Medeshetty NER використовується разом з класичними алгоритмами машинного навчання – SVM, Random Forest, Decision Tree, Gradient Boosting – для віднесення фрагментів тексту до відповідних типів сутностей [5]. За результатами експериментів найкращі показники демонструє SVM, досягаючи точності на рівні 77,3 % на тестовому наборі, тоді як інші алгоритми показують нижчі значення точності та F1-міри [5].

Переваги методів машинного навчання з NER:

- адаптивність до різних типів і форматів вимог за наявності відповідних анотованих корпусів;
- можливість донавчання моделей при надходженні нових даних;
- здатність обробляти складніші формати, де жорсткі правила важко сформулювати вручну;
- потенційна масштабованість до великих обсягів документації.

Недоліки:

- значна залежність від якості та обсягу навчальних даних;
- складність налаштування моделей для вузьких доменів;

У проаналізованому дослідженні загальна практична ефективність NER-підходу виявилася нижчою, ніж у правило-орієнтованого методу для простіших типів вимог [5].

Таким чином, NER-базовані методи доцільно розглядати як інструмент для сценаріїв, де формат вимог більш варіативний, але доступні достатньо великі й репрезентативні набори навчальних даних [8–10].

Окремий напрям пов'язаний з інтеграцією NER та інших технік обробки вимог із моделями приймального тестування. Wang та співавтори запропонували NLP-орієнтований підхід до автоматичної генерації acceptance-тестів із use case специфікацій, у якому вимоги спочатку структуруються, а потім перетворюються на формальні сценарії з умовами та очікуваними результатами [24]. Allala та ін. комбінують моделювання, кероване моделлю (MDSE), з NLP для побудови абстрактних тест-кейсів на основі користувацьких вимог, забезпечуючи трасованість між вимогами, моделями й тестами [25]. Такі роботи демонструють, що ML- та NLP-методи можуть ефективно поєднуватися з інженерією вимог, забезпечуючи більш тісний зв'язок між текстовими артефактами та тестовими сценаріями.

Трансформерні моделі.

Сучасні трансформерні моделі є основою багатьох підходів до автоматизованої генерації коду та тестів. У контексті генерації тест-кейсів вони використовуються як у форматі спеціалізованих моделей (наприклад, ATHENATEST), так і у вигляді великих мовних моделей загального призначення [4, 6, 11, 12]. Ці моделі використовують архітектуру sequence-to-sequence та здатні генерувати тест-кейси навчаючись на великих корпусах реальних тестів написаних розробниками.

У роботі Tufano та співавторів запропоновано ATHENATEST – підхід, який навчається на реальних юніт-тестах, написаних розробниками, і генерує кандидатні тести для заданих «фокальних» методів [4]. Основні результати:

- згенеровано 25 000 коректних тестів, які автори публічно виклали у відкритий доступ;
- при генерації до 30 кандидатів на один метод ATHENATEST спроможна правильно протестувати 43 % фокальних методів, при цьому близько 16 % кандидатних тестів виявляються коректними;
- покриття коду, досягнуте ATHENATEST, є порівняним із EvoSuite і в окремих випадках перевищує його;
- за результатами опитування професійних розробників, тести, згенеровані ATHENATEST, сприймаються як більш читабельні, зрозумілі та ефективні порівняно з тестами EvoSuite [4].

Робота Albdeiwi присвячена застосуванню великих мовних моделей до задачі генерації тест-кейсів на основі текстових вимог [12]. Автор оцінює якість за допомогою:

- автоматичної метрики BLEU (до 32,93 для найкращої конфігурації),
- людської експертної оцінки (максимальна середня оцінка 3,71/4.68)[12].

Переваги трансформерних і LLM-підходів:

- здатність враховувати широкий контекст і семантику вимог;
- потенційно висока якість та різноманітність згенерованих сценаріїв;

- можливість адаптації до різних предметних областей за рахунок донавчання або підказкового програмування (prompting).

Обмеження:

- значні обчислювальні витрати на навчання й «інференс» великих моделей;
- потреба у якісних наборах даних для конкретних доменів;
- необхідність додаткових механізмів контролю коректності та повноти згенерованих тестів [11, 13].

Останні дослідження демонструють інтеграцію великих мовних моделей безпосередньо в процес requirements-driven тестування. Arora та співавтори запропонували RAGTAG — підхід до генерації тестових сценаріїв із двомовних (англійсько-німецьких) вимог із використанням Retrieval-Augmented Generation; результати промислового кейсу показали, що згенеровані сценарії є релевантними, зрозумілими для експертів і придатними до виконання на практиці [28]. Karmouda та співавтори досліджують застосування Claude 3.5 Sonnet для автоматичного виділення умовних тверджень із функціональних вимог і формування на їх основі тест-кейсів [29]. Узагальнений огляд застосування LLM у програмному тестуванні, включно з генерацією тестів, пріоритизацією та аналізом результатів, подано в роботі Wang та співавторів. [30], що підтверджує зростаючу роль трансформерних моделей як універсального інструменту підтримки QA-процесів.

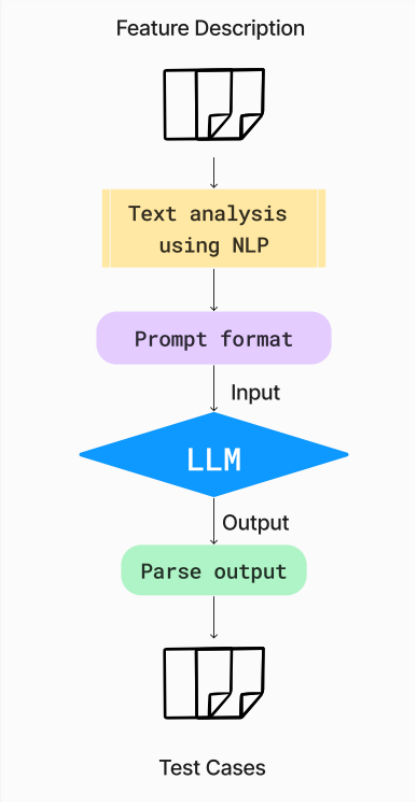

Рис. 3. Схема загального пайплайну генерації тест-кейсів за допомогою LLM
Джерело: [12].

На рисунку 3 подано узагальнений конвеєр автоматичної генерації тест-кейсів на основі великих мовних моделей. Спочатку текстове описання функціональності проходить попередню обробку за допомогою методів NLP, після чого формуються інструкції та шаблон промпта. Отриманий промпт передається LLM, яка генерує первинний варіант тест-кейсів. Згенерований вихід додатково розбирається та структурується у формат тестової документації.

Порівняльний аналіз підходів.

Таблиця 1 – Порівняння методів NLP

Підхід / Джерело	**Технічна реалізація підходу**	**Типові вхідні дані**	**Якісна оцінка результатів**	**Переваги**	**Обмеження**	**Тип генерованих тестів**
Rule-based (Gröpler та ін. 2021)	Лінгвістичний парсинг, POS-теги, dependency parsing, ручні правила	Текстові вимоги	Повнота: висока; коректність: висока; консистентність: дуже висока.	Інтерпретованість, низькі вимоги до даних	Залежність від правил, чутливість до формулювань	Абстрактні тест-кейси
Rule-based vs NER (Medeshetty та ін., 2023)	Rule-based патерн-матчинг; NER на SVM	400 документів Polarion	Rule-based: **висока** точність для простих випадків, **середня** для складніших; NER (SVM):– **висока** точність	Висока точність, скорочення ручної роботи	NER потребує навчання; rule-based слабкі для складних фраз	Специфікації тест-кейсів для ECU
Transformer (ATHENATEST, Tufano та ін., 2021)	Seq2Seq BART; попереднє + supervised навчання	Java-код (фокальні методи) + тести	Покриття фокальних методів: **середній рівень;** якість і корисність тестів – високій рівень.	Висока якість, наближеність до людських тестів	Потребує великих обчислювальних ресурсів	Юніт-тести
LLM (Albdeiwi та El-Khalil, 2024)	LLM (Mistral-7B), prompt engineering, fine-tuning	Текстові специфікації	BLEU: висока відповідність еталонним тестам.	Працює на малих GPU, стабільність	Залежність від промптів та обсягів даних	Текстові структуровані тести

Джерело: складено авторами на основі [3; 4; 5; 12].

Порівняльний аналіз в оглядовій Таблиці 1 показує, що сучасні підходи до автоматизованої генерації тест-кейсів на основі NLP суттєво відрізняються за рівнем формалізації, вимогами до ресурсів і характером результатів. Правило-орієнтовані методи (Gröpler; Medeshetty) демонструють високу точність у вузьких доменах та забезпечують передбачуваність і інтерпретованість, але потребують ручного налаштування правил і погано масштабуються на складні або неоднорідні вимоги. Підходи на основі NER і класичних ML-моделей показують помірну точність і вимагають анотованих даних, проте здатні частково зменшувати залежність від жорстких шаблонів.

Методи, що використовують трансформерні моделі (Tufano; Albdeiwi), досягають значно вищої гнучкості у роботі з текстом і демонструють кращу відповідність людським прикладам – як за покриттям, так і за експертними оцінками якості. ATHENATEST підтверджує здатність моделей Seq2Seq генерувати тест-кейси, близькі до таких, які пишуть розробники, а LLM-підходи дозволяють отримувати якісні текстові тест-кейси навіть на відносно обмежених обчислювальних ресурсах.

Висновки. Проведений у статті аналіз дозволив систематизувати три основні підходи до автоматизованого генерування тест-кейсів на основі природномовних вимог та узагальнити їх методологічні особливості. На відміну від існуючих робіт, у дослідженні структуровано послідовність перетворення вимог у тестові сценарії та окреслено ключові фактори, що впливають на ефективність обраного підходу. Практична цінність роботи полягає в можливості застосування отриманих узагальнень для обґрунтованого вибору інструментів і методів у процесі автоматизації тест-дизайну. Порівняння показало, що правило-орієнтовані методи доцільні для проєктів зі стабільними вимогами, алгоритми машинного навчання – для середовищ зі змінною структурою вимог, а трансформерні моделі – для комплексних систем із великим обсягом текстових даних. Подальші дослідження можуть бути спрямовані на створення гібридних підходів, удосконалення методів попередньої обробки вимог та адаптацію сучасних моделей до специфічних предметних областей. Отримані узагальнення можуть слугувати практичними рекомендаціями для вибору підходу залежно від типу проєкту.

Список використаних джерел

1. Макуховська Д.А. (2025) Моделі тестування при розробці web-платформи для обміну 3D-моделями. *Вісник ХНТУ*, (3), 45–52.

2. Boukhlif, M., Hanine, M., Kharmoum, N., Ruigómez Noriega, A., García Obeso, D., & Ashraf, I. (2024). Natural Language Processing-Based Software Testing: A Systematic Literature Review. *IEEE Access*, 12, 43132–43154. https://doi.org/10.1109/ACCESS.2024.3407753

3. Gröpler, R., Sudhi, V., García, E. J. C., & Bergmann, A. (2021). NLP-Based Requirements Formalization for Automatic Test Case Generation. *CEUR Workshop Proceedings*, (Vol. 2951). https://ceur-ws.org/Vol-2951/paper15.pdf.

4. Tufano, M., Drain, D., Svyatkovskiy, A., Deng, S. K., & Sundaresan, N. (2020). Unit Test Case Generation with Transformers and Focal Context. *arXiv preprint arXiv:2009.05617*. https://doi.org/10.48550/arXiv.2009.05617.

5. Medeshetty, N., et al. (2025). Fropm Requirements to Test Cases: An NLP-Based Approach for High-Performance ECU Test Case Automation. *arXiv preprint arXiv:2505.00547*. https://doi.org/10.48550/arXiv.2505.00547.

6. Shakthi, S., Srivastava, P., Kumar, R. L., & Prasad, S. G. (2024). Automated Test Case Generation for Satellite FRD Using NLP and Large Language Model. *4th International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME*). (pp. 1–6). IEEE. https://doi.org/10.1109/ICECCME62383.2024.10796866.

7. Kumar, S., Napte, K., Rani, R., Pippal, S. K. (2025). A method for IoT devices test case generation using language models. *MethodsX*, 14, 103340. https://doi.org/10.1016/j.mex.2025.103340.

8. Mahalakshmi, G. S., Vani, V., & Antony, B. (2018). Named Entity Recognition for Automated Test Case Generation. *The International Arab Journal of Information Technology*, 15(1), 112–120. https://www.iajit.org/paper/3691.

9. Malik, G., Cevik, M., Khedr, Y., Parikh, D., Başar, A. (2021). Named Entity Recognition on Software Requirements Specification Documents. *34th Canadian Conference on Artificial Intelligence (Canadian AI 2021)*. https://doi.org/10.21428/594757db.507e7951.

10. Celik, A., & Mahmoud, Q. H. (2025). A Review of Large Language Models for Automated Test Case Generation. *Machine Learning and Knowledge Extraction*, 7(3), 97. https://doi.org/10.3390/make7030097.

11. Korraprolu, B. R., Pinninti, P., Reddy, Y. R. (2025). Test Case Generation for Requirements in Natural Language – An LLM Comparison Study. *18th Innovations in Software Engineering Conference (ISEC 2025)*. https://doi.org/10.1145/3717383.3717389.

12. Albdeiwi, Y. (2024). Generating Test Cases Using Natural Language Processing. Master's thesis, Lund University. https://lup.lub.lu.se/student-papers/search/publication/9168588.

13. Iznaga, Y. S., Rato, L., Salgueiro, P., & León, J. L. (2025). Integrating Large Language Models into Automated Software Testing. *Future Internet*, 17(10), 476. https://doi.org/10.3390/fi17100476.

14. Mustafa, A., Wan-Kadir, W. M. N., Ibrahim, N., Shah, M. A., Younas, M., et al. (2021). Automated Test Case Generation from Requirements: A Systematic Literature Review. *Computers, Materials & Continua*, 67(2), 1819–1833. https://doi.org/10.32604/cmc.2021.014391.

15. Farooq, M. S., & Tahreem, T. (2022). Requirement-Based Automated Test Case Generation: Systematic Literature Review. *VFAST Transactions on Software Engineering*, 10(2), 133–142. https://doi.org/10.21015/vtse.v10i2.940.

16. Garousi, V., Bauer, S., & Felderer, M. (2020). NLP-assisted software testing: A systematic mapping of the literature. *Information and Software Technology*, 126, 106321. https://doi.org/10.1016/j.infsof.2020.106321.

17. Ayenew, H., & Wagaw, M. (2024). Software Test Case Generation Using Natural Language Processing (NLP): A Systematic Literature Review. *Artificial Intelligence Evolution*, 5(1), 35-56. https://doi.org/10.37256/aie.5120243220.

18. Navarro Calizaya, J., & Ibarra, R. (2025). Automatic test case generation using natural language processing: A systematic mapping study. *Information and Software Technology*, 189, 107929. https://doi.org/10.1016/j.infsof.2025.107929.

19. Dwarakanath, A., & Sengupta, B. (2012). Litmus: Generation of Test Cases from Functional Requirements in Natural Language. *Natural Language to Information Systems (NLDB 2012),* (Lecture Notes in Computer Science, Vol. 7337). Springer. https://doi.org/10.1007/978-3-642-31178-9_6.

20. Santiago Júnior, V., & Vijaykumar, N. (2012). Generating Model-Based Test Cases from Natural Language Requirements for Space Application Software. *Software Quality Journal*, 20(1), 77–143. https://doi.org/10.1007/s11219-011-9155-6.

21. Carvalho, G., Falcão, D., Barros, F. A., Blackburn, M. R. (2014). NAT2TEST(SCR): Test Case Generation from Natural Language Requirements Based on SCR Specifications. *Science of Computer Programming*, 95, 275–297. https://doi.org/10.1145/2480362.2480591.

22. Sarmiento, E., Leite, J. C. S. P., Almentero, E., & Alzamora, G. S. (2016). Test Scenario Generation from Natural Language Requirements Descriptions based on Petri-Nets. *Electronic Notes in Theoretical Computer Science*, 329, 123–148. https://doi.org/10.1016/j.entcs.2016.12.008.

23. Silva, B. C. F., Carvalho, G., & Sampaio, A. (2019). CPN Simulation-Based Test Case Generation from Controlled Natural-Language Requirements. *Science of Computer Programming*, 181, 111–139. https://doi.org/10.1016/j.scico.2019.04.001.

24. Wang, C., Pastore, F., Goknil, A., & Briand, L. C. (2020). Automatic Generation of Acceptance Test Cases from Use Case Specifications: An NLP-Based Approach. *IEEE Transactions on Software Engineering*, 48(2), 585–616. https://doi.org/10.48550/arXiv.1907.08490.

25. Allala, S. Ch., Sotomayor, J. P., Santiago, D., King, T. M., & Clarke, P. J. (2022). Generating Abstract Test Cases from User Requirements Using MDSE and NLP. *2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)* (c. 138-147). IEEE. https://doi.org/10.1109/QRS57517.2022.00080.

26. Vani, V., & Mahalakshmi, G. S. (2015). Generation of Patterns for Test Cases. *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)* (c. 526-531). IEEE. https://doi.org/10.17485/ijst/2015/v8i24/80176.

27. Chioasca, E.-V., Zhao, L., Alhoshan, W., & Ferrari, A. (2020). Natural Language Processing (NLP) for Requirements Engineering: A Systematic Mapping Study. *Proceedings of the 17th International Conference on Mining Software Repositories* (c. 391–405). https://doi.org/10.48550/arXiv.2004.01099.

28. Arora, C., Herda, T., & Homm, V. (2024). Generating Test Scenarios from NL Requirements Using Retrieval-Augmented LLMs: An Industrial Study. *32nd IEEE International Requirements Engineering Conference (RE 2024)* (c. 240–251). IEEE. https://doi.org/10.48550/arXiv.2404.12772.

29. Karmouda, O., Ghaouat, M., & Feuilloley, G. (2025). Automated Test Case Generation from Natural Language Requirements Using Large Language Models. *Future Technologies Conference (FTC 2025)* (Lecture Notes in Networks and Systems, Vol. 1677, c. 260–281). Springer. https://doi.org/10.1007/978-3-032-07995-4_18.

30. Wang, J., Huang, Y., Chen, C., Liu, Z., Wang, S., & Wang, Q. (2024). Software Testing with Large Language Models: Survey, Landscape, and Vision. *IEEE Transactions on Software Engineering*, 50(4), 911–936. https://doi.org/10.48550/arXiv.2307.07221.

References

1. Makukhovska, D. A. (2025). Modeli testuvannia pry rozrobtsi web-platformy dlia obminu 3D-modeliamy [Testing models in the development of a web platform for sharing 3D models]. *Visnyk KhNTU – Bulletin of KhNTU,* (3), 45–52.

2. Boukhlif, M., Hanine, M., Kharmoum, N., Ruigómez Noriega, A., García Obeso, D., & Ashraf, I. (2024). Natural Language Processing-Based Software Testing: A Systematic Literature Review. *IEEE Access*, 12, 43132–43154. https://doi.org/10.1109/ACCESS.2024.3407753

3. Gröpler, R., Sudhi, V., García, E. J. C., & Bergmann, A. (2021). NLP-Based Requirements Formalization for Automatic Test Case Generation. *CEUR Workshop Proceedings*, (Vol. 2951). https://ceur-ws.org/Vol-2951/paper15.pdf.

4. Tufano, M., Drain, D., Svyatkovskiy, A., Deng, S. K., & Sundaresan, N. (2020). Unit Test Case Generation with Transformers and Focal Context. *arXiv preprint arXiv:2009.05617*. https://doi.org/10.48550/arXiv.2009.05617.

5. Medeshetty, N., et al. (2025). From Requirements to Test Cases: An NLP-Based Approach for High-Performance ECU Test Case Automation. *arXiv preprint arXiv:2505.00547*. https://doi.org/10.48550/arXiv.2505.00547.

6. Shakthi, S., Srivastava, P., Kumar, R. L., & Prasad, S. G. (2024). Automated Test Case Generation for Satellite FRD Using NLP and Large Language Model. *4th International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME*). (pp. 1–6). IEEE. https://doi.org/10.1109/ICECCME62383.2024.10796866.

7. Kumar, S., Napte, K., Rani, R., Pippal, S. K. (2025). A method for IoT devices test case generation using language models. *MethodsX*, 14, 103340. https://doi.org/10.1016/j.mex.2025.103340.

8. Mahalakshmi, G. S., Vani, V., & Antony, B. (2018). Named Entity Recognition for Automated Test Case Generation. *The International Arab Journal of Information Technology*, 15(1), 112–120. https://www.iajit.org/paper/3691.

9. Malik, G., Cevik, M., Khedr, Y., Parikh, D., Başar, A. (2021). Named Entity Recognition on Software Requirements Specification Documents. *34th Canadian Conference on Artificial Intelligence (Canadian AI 2021)*. https://doi.org/10.21428/594757db.507e7951.

10. Celik, A., & Mahmoud, Q. H. (2025). A Review of Large Language Models for Automated Test Case Generation. *Machine Learning and Knowledge Extraction*, 7(3), 97. https://doi.org/10.3390/make7030097.

11. Korraprolu, B. R., Pinninti, P., Reddy, Y. R. (2025). Test Case Generation for Requirements in Natural Language – An LLM Comparison Study. *18th Innovations in Software Engineering Conference (ISEC 2025)*. https://doi.org/10.1145/3717383.3717389.

12. Albdeiwi, Y. (2024). Generating Test Cases Using Natural Language Processing. Master's thesis, Lund University. https://lup.lub.lu.se/student-papers/search/publication/9168588.

13. Iznaga, Y. S., Rato, L., Salgueiro, P., & León, J. L. (2025). Integrating Large Language Models into Automated Software Testing. *Future Internet*, 17(10), 476. https://doi.org/10.3390/fi17100476.

14. Mustafa, A., Wan-Kadir, W. M. N., Ibrahim, N., Shah, M. A., Younas, M., et al. (2021). Automated Test Case Generation from Requirements: A Systematic Literature Review. *Computers, Materials & Continua*, 67(2), 1819–1833. https://doi.org/10.32604/cmc.2021.014391.

15. Farooq, M. S., & Tahreem, T. (2022). Requirement-Based Automated Test Case Generation: Systematic Literature Review. *VFAST Transactions on Software Engineering*, 10(2), 133–142. https://doi.org/10.21015/vtse.v10i2.940.

16. Garousi, V., Bauer, S., & Felderer, M. (2020). NLP-assisted software testing: A systematic mapping of the literature. *Information and Software Technology*, 126, 106321. https://doi.org/10.1016/j.infsof.2020.106321.

17. Ayenew, H., & Wagaw, M. (2024). Software Test Case Generation Using Natural Language Processing (NLP): A Systematic Literature Review. *Artificial Intelligence Evolution*, 5(1), 35-56. https://doi.org/10.37256/aie.5120243220.

18. Navarro Calizaya, J., & Ibarra, R. (2025). Automatic test case generation using natural language processing: A systematic mapping study. *Information and Software Technology*, 189, 107929. https://doi.org/10.1016/j.infsof.2025.107929.

19. Dwarakanath, A., & Sengupta, B. (2012). Litmus: Generation of Test Cases from Functional Requirements in Natural Language. *Natural Language to Information Systems (NLDB 2012),* (Lecture Notes in Computer Science, Vol. 7337). Springer. https://doi.org/10.1007/978-3-642-31178-9_6.

20. Santiago Júnior, V., & Vijaykumar, N. (2012). Generating Model-Based Test Cases from Natural Language Requirements for Space Application Software. *Software Quality Journal*, 20(1), 77–143. https://doi.org/10.1007/s11219-011-9155-6.

21. Carvalho, G., Falcão, D., Barros, F. A., Blackburn, M. R. (2014). NAT2TEST(SCR): Test Case Generation from Natural Language Requirements Based on SCR Specifications. *Science of Computer Programming*, 95, 275–297. https://doi.org/10.1145/2480362.2480591.

22. Sarmiento, E., Leite, J. C. S. P., Almentero, E., & Alzamora, G. S. (2016). Test Scenario Generation from Natural Language Requirements Descriptions based on Petri-Nets. *Electronic Notes in Theoretical Computer Science*, 329, 123–148. https://doi.org/10.1016/j.entcs.2016.12.008.

23. Silva, B. C. F., Carvalho, G., & Sampaio, A. (2019). CPN Simulation-Based Test Case Generation from Controlled Natural-Language Requirements. *Science of Computer Programming*, 181, 111–139. https://doi.org/10.1016/j.scico.2019.04.001.

24. Wang, C., Pastore, F., Goknil, A., & Briand, L. C. (2020). Automatic Generation of Acceptance Test Cases from Use Case Specifications: An NLP-Based Approach. *IEEE Transactions on Software Engineering*, 48(2), 585–616. https://doi.org/10.48550/arXiv.1907.08490.

25. Allala, S. Ch., Sotomayor, J. P., Santiago, D., King, T. M., & Clarke, P. J. (2022). Generating Abstract Test Cases from User Requirements Using MDSE and NLP. *2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)* (c. 138-147). IEEE. https://doi.org/10.1109/QRS57517.2022.00080.

26. Vani, V., & Mahalakshmi, G. S. (2015). Generation of Patterns for Test Cases. *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)* (c. 526-531). IEEE. https://doi.org/10.17485/ijst/2015/v8i24/80176.

27. Chioasca, E.-V., Zhao, L., Alhoshan, W., & Ferrari, A. (2020). Natural Language Processing (NLP) for Requirements Engineering: A Systematic Mapping Study. *Proceedings of the 17th International Conference on Mining Software Repositories* (c. 391–405). https://doi.org/10.48550/arXiv.2004.01099.

28. Arora, C., Herda, T., & Homm, V. (2024). Generating Test Scenarios from NL Requirements Using Retrieval-Augmented LLMs: An Industrial Study. *32nd IEEE International Requirements Engineering Conference (RE 2024)* (c. 240–251). IEEE. https://doi.org/10.48550/arXiv.2404.12772.

29. Karmouda, O., Ghaouat, M., & Feuilloley, G. (2025). Automated Test Case Generation from Natural Language Requirements Using Large Language Models. *Future Technologies Conference (FTC 2025)* (Lecture Notes in Networks and Systems, Vol. 1677, c. 260–281). Springer. https://doi.org/10.1007/978-3-032-07995-4_18.

30. Wang, J., Huang, Y., Chen, C., Liu, Z., Wang, S., & Wang, Q. (2024). Software Testing with Large Language Models: Survey, Landscape, and Vision. *IEEE Transactions on Software Engineering*, 50(4), 911–936. https://doi.org/10.48550/arXiv.2307.07221.

Дата першого надходження статті до видання: 08.10.2025
Дата прийняття статті до друку після рецензування: 21.11.2025

UDC 004.8:004.4

Oleksii Maksymov[1], Dmytro Lysenko[2]

[1]PhD student of the Department of Information and Computer Systems
Chernihiv Polytechnic National University (Chernihiv, Ukraine)
E-mail: maksimov98gmy@gmail.com. **ORCID:** https://orcid.org/0009-0008-4257-6549

[2]Doctor of Technical Sciences, Professor of the Department of Information and Computer Systems,
Chernihiv Polytechnic National University (Chernihiv, Ukraine)
E-mail: lysenko.d@stu.cn.ua. **ORCID:** https://orcid.org/0000-0001-6870-6120

COMPARATIVE ANALYSIS OF NATURAL LANGUAGE PROCESSING METHODS FOR AUTOMATED TEST CASE GENERATION

The article addresses the problem of systematizing existing NLP-based approaches used for automated test case generation. Despite the availability of different methods, there is still a lack of comprehensive comparison that would highlight their conceptual differences, and limitations. The need to clarify the structure of these approaches and evaluate their suitability for various software domains forms the core motivation of the presented research.

The objective of the article is to provide an integrated comparative overview of three main groups of methods: rule-based approaches, machine-learning techniques relying on named entity recognition, and approaches based on transformer models. The article aims to describe the methodological foundations of each group, the stages of processing natural-language requirements, and the typical workflow of transforming textual input into test scenarios.

The main part of the article outlines the structure of automated test case generation using NLP, including the extraction of key entities, identification of system actions and conditions, and formation of steps for test scenarios. Each methodological group is described in terms of its internal principles, linguistic processing techniques, adaptability to different formats of requirements, and dependence on training data. Special attention is paid to the role of preprocessing, domain specificity, rule formulation, annotation needs in machine-learning methods, and the modelling capabilities of transformer-based architectures. A comparative description summarizes their conceptual differences, strengths, and potential limitations.

The conclusions generalize the findings of the analysis, emphasizing the importance of selecting an appropriate NLP method depending on the requirements. The material presented in the article can be used as a methodological basis for choosing or developing NLP-based tools for automated test design in software projects.

Keywords*: natural language processing; automated testing; test case generation; machine learning; optimization; software quality.*

Fig.: 3. Table: 1. References: 30.

Максимов О. В., Лисенко Д. Е. Порівняльний аналіз методів використання природномовної обробки для автоматизованого генерування тест-кейсів. *Технічні науки та технології,* 2025. № 4(42). С. 233-244. DOI: https://doi.org/10.25140/2411-5363-2025-4(42)-233-244.