УДК 004.415.5:004.8 **doi: 10.32620/aktt.2024.2.10**

А. О. СТРЯПУНІН, В. С. ХАРЧЕНКО

Національний аерокосмічний університет імені М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна

ВИКОРИСТАННЯ ЗАСОБІВ ШТУЧНОГО ІНТЕЛЕКТУ В ІНЖЕНЕРІЇ ВИМОГ: АНАЛІЗ МОЖЛИВОСТЕЙ ТА ЧАТ-БОТ ДЛЯ ВАЛІДАЦІЇ

Об'єктом дослідження *є процеси і засоби інженерії вимог (ІВ) програмного забезпечення і складних систем (ПЗС).* ***Предметом дослідження*** *є інструментальні засоби ІВ ПЗС, які базуються на методах штучного (обчислювального) інтелекту (ШІ).* ***Метою*** *дослідження є покращення точності і ефективності процесів розроблення вимог до ПЗС з використанням, засобів ШІ завдяки забезпеченню кращих комунікацій між бізнес-командами та технічними командами та автоматизації складних процесів збору та документації вимог.* ***Завдання****: проаналізувати принципи і засоби інтеграції інструментів ШІ в процеси інженерії вимог, розроблення і перевірка яких є критичним етапом у розробленні ПЗС; визначити проблеми використання традиційних методів ІВ ПЗС і виконати їх порівняльний аналіз із засобами на основі ШІ; розробити архітектуру чат-боту для валідації вимог та визначити характеристики процесів ІВ, які поліпшуються завдяки його використанню.* ***Висновки та результати****. Результати дослідження демонструють, що інструменти ШІ мають потенціал для суттєвого покращення точності,, часових характеристик і ефективності процесів розроблення вимог та комунікації проєктних команд. Запропоновано архітектуру і програмне рішення для інтелектуального чат-боту для валідації вимог. Сформульовано напрями подальших розробок і досліджень стосовно забезпечення наскрізного впровадження засобі ШІ в процеси ІВ.*

Ключові слова: *штучний інтелект; інженерія вимог; комунікації в проєктах; валідація вимог; чат-бот.*

1. Вступ

1.1. Актуальність

Роль інженерії вимог (ІВ) набуває все більшої ваги у забезпеченні успіху програмних проєктів. Визначення та управління вимогами є критично важливими для розроблення якісного програмного забезпечення і складних інформаційних систем (ПЗ), що відповідає потребам та очікуванням замовників.

Можливо найбільш яскравим прикладом неузгодженості вимог та процесів роботи між командами технічних спеціалістів була катастрофа космічного апарату Mars Climate Orbiter [1], який був запущений NASA 11 грудня 1998 року з метою вивчення клімату та атмосфери Марса. Одна з команд під час передачі даних про курс та розрахунку місії, використовувала англійську систему мір замість метричної. Це призвело до того, що через неправильні отримані результати обчислень марсохід наблизився занадто близько до поверхні Марса при його входженні в атмосферу і зіткнувся з нею.

Це дослідження фокусується на використанні технологій штучного інтелекту (ШІ) для оптимізації процесів інженерії вимог. Застосування ШІ має потенціал значно підвищити ефективність цього процесу, забезпечуючи більш точний аналіз, виявлення та управління вимогами [2].

Інженерія вимог (Requirements Engineering - RE) — це важлива практика розробки програмного забезпечення, яка передбачає визначення, документування та підтримку вимог і потреб зацікавлених сторін щодо системи програмного забезпечення. Це основа для успішного розроблення ПЗ, яка гарантує, що кінцевий продукт відповідає поставленій меті та задовольняє потреби користувачів [3].

The Software Engineering Institute (SEI) визначає розроблення вимог як процес виявлення, документування та вдосконалення бажаних можливостей та якостей програмної системи, які відповідають потребам зацікавлених сторін. Цей процес починається з визначення зацікавлених сторін та їхніх інтересів, а потім виявлення їхніх вимог, аналізуючи та визначаючи пріоритети, чітко та недвозначно вказуючи їх, а також перевіряти та підтверджувати.

Завдяки якісному (повному, детальному та корректному) документуванню вимог усі зацікавлені сторони працюють злагоджено, зменшуючи непорозуміння та витрати на переробку. Своєчасне

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

виявлення та вирішення потенційних проблем може запобігти вартісним і критичним помилкам у подальшому процесі розроблення. Вимоги, які чітко визначені та зрозумілі, з більшою ймовірністю будуть реалізовані правильно, що забезпечить вищу якість кінцевого продукту ПЗ.

1.2. Мотивація

Процес оцінювання вимог до програмного забезпечення є одним із найважливіших і водночас найскладніших у життєвому циклі розроблення програмного забезпечення. Ефективне управління та оцінка вимог має вирішальне значення для успіху будь-якого програмного проєкту. Проте, існує ряд проблем та викликів, які можуть призвести до серйозних наслідків, якщо вони не вирішуються належним чином.

Традиційні процеси розроблення вимог стикаються з декількома проблемами, які можуть вплинути на якість і успіх проєктів ПЗ:

– якість вимог часто страждає від двозначності та неправильного тлумачення, що призводить до неузгодженості очікувань між зацікавленими сторонами та розробниками. Зміни до вимог ускладнюють процес керування та відстеження цих коригувань;

– ефективне розуміння та документування потреб усіх зацікавлених сторін є складним завданням, особливо в складних проєктах із великою кількістю стейкхолдерів та складною бізнес-логікою;

– належне документування та управління вимогами у спосіб, який є доступним і зрозумілим для всіх сторін, потребує значних зусиль та відповідальності;

– забезпечення того, щоб кінцевий програмний продукт найбільш повно відповідає окресленим вимогам, є складним завданням, яке часто ускладнюється початковими проблемами у визначенні цих вимог і управлінні ними.

При розробленні архітектури системи крім функціональних та нефункціональних вимог також виділяють архітектурно значущі вимоги (ASR) [4] – це тип вимог до програмного забезпечення, які мають значний вплив на архітектуру системи. Ці вимоги зазвичай описують фундаментальні властивості або поведінку системи. Кожна зміна або помилка визначення таких вимог може призвести до значних наслідків у процесі створення та експлуатації системи, великих втрат часу і коштів.

Але навіть якщо архітектурні вимоги буди вірно визначені та дизайн системи було обрано правильно, кожна наступна зміна або реалізація нового функціоналу має проходити узгодження з основними вимогами та обмеженнями системи. Зазвичай ця робота лягає на плечі архітектора проєкту і проводиться у “ручному” режимі, де кожен спеціаліст проводить перевірку відповідно до свого досвіду, доступних інструментів та наявної документації. Цей процес може покращити завдяки використанню більш формалізованого підходу до збору та документування вимог, а також його автоматизації з використанням ШІ.

Впроваджуючи формалізовані стандартні специфікації, можемо спростити подальше написання тестової документації також використовуючи ШІ для аналізу та генерації документації [2]. В подальшому це надасть змогу створити систему, яка на підставі неформально описаних вимог синтезує формалізовані специфікації до програм та формалізовану тестову документацію, і навіть зможе додавати приклади готових компонентів програм, використовуючи відповідні генератори коду. Оскільки функціонування систем ШІ не є стовідсотково коректним і ефективним, а також залежить від якості даних для навчання, то процес документування ще довго не буде повністю автономним, хоча зможе значно прискоритися та покращити ефективність роботи технічних спеціалістів.

Інтеграція інструментів штучного інтелекту у процеси розроблення вимог створює значний потенціал для вдосконалення:

– ШІ може автоматизувати аналіз складних вимог, зменшуючи ручні зусилля та підвищуючи точність оцінки;

– інструменти ШІ сприяють кращій комунікації між зацікавленими сторонами шляхом тлумачення та роз’яснення вимог, зменшення двозначності;

– системи ШІ швидко адаптуються до змін у вимогах, забезпечуючи відповідність процесу розробки програмного забезпечення потребам, що розвиваються;

– ШІ може передбачати потенційні проблеми або конфлікти у вимогах, дозволяючи вносити активні коригування;

– технології ШІ дозволяють ефективніше перевіряти вимоги до розробленого програмного забезпечення, забезпечуючи відповідність очікуванням зацікавлених сторін.

Отже, засоби штучного інтелекту можуть оптимізувати процес розроблення вимог, надаючи йому більшої точності, узгодженості та ефективності.

1.3. Термінологія

Термінологія, яка використовується в статті, базується на положеннях, викладених в [3, 5, 6].

Штучний інтелект – це галузь інформатики, яка об’єднує методи створення інтелектуальних машин, здатних виконувати завдання, що зазвичай потребують людського інтелекту. Системи ШІ навчаються на досвіді, розпізнають закономірності та приймають рішення на основі вхідних даних.

Машинне навчання (МН) – це галузь штучного інтелекту, яка дає змогу комп'ютерним системам навчатися без явного програмування. Алгоритми машинного навчання навчаються на даних, виявляючи закономірності та роблячи прогнози або приймаючи рішення на основі нових даних.

Оброблення природної мови (NLP) – це галузь штучного інтелекту (ШІ), яка прагне дати комп'ютерам можливість розуміти, обробляти та генерувати людську мову. ОНМ використовує різні методи, такі як машинне навчання, статистику та лінгвістику, для вирішення широкого кола завдань, пов'язаних з мовою.

Велика мовна модель (LLM) – це тип штучного інтелекту (ШІ), який використовує машинне навчання для обробки та генерування великих обсягів текстових даних. LLM навчаються на великих наборах даних тексту й коду, що дає їм можливість виконувати широкий спектр завдань, пов'язаних з роботою з мовою.

Generative Pre-training Transformer (GPT) - це сімейство великих мовних моделей, розроблених компанією OpenAI та інших дослідницьких лабораторіях. Їх навчають на величезних обсягах текстових даних, щоб генерувати текст, подібний до людського, перекладати мови, писати різні види творчого контенту та інформативно відповідати на ваші запитання.

1.4. Мета і завдання

У статті досліджуються проблематика використання засобів штучного інтелекту для удосконалення процесів інженерії вимог.

Метою є покращення точності і ефективності процесів розроблення вимог до ПЗ з використанням,засобів ШІ завдяки забезпеченню кращих комунікацій між бізнес-командами та технічними командами та автоматизації складних процесів збору та документації вимог.

Завдання є наступними:

– проаналізувати принципи і засоби інтеграції інструментів ШІ в процеси інженерії вимог, розроблення і перевірка яких є критичним етапом у розробленні ПЗС (розділ 2);

– визначити проблеми використання традиційних методів ІВ ПЗ і виконати їх порівняльний аналіз із засобами на основі ШІ (розділ 3);

– розробити архітектуру чат-боту для валідації вимог та визначити характеристики процесів ІВ, які поліпшуються завдяки його використанню (розділ 4);

– сформулювати загальні висновки щодо використання засобів ШІ та напрями подальших досліджень і розробок задля покращення якості вимог до ПЗ (розділ 5).

2. Аналіз публікацій

У [3] проводиться огляд сучасних варіантів застосувань ШІ у сфері розроблення вимог до програмного забезпечення. Вона спрямована на поєднанні останніх досліджень і визначення того, як різні методи ШІ, такі як машинне навчання та обробка природної мови (NLP) були інтегровані в процеси інженерії вимог для підвищення якості та точності вимог до програмного забезпечення.

Огляд охоплює літературу з січня 2015 року по грудень 2021 року, з останніми поглядами на інтеграцію штучного інтелекту в інженерії вимог. У документі описано, як NLP і машинне навчання, були застосовані для вдосконалення таких завдань, як документування вимог та управління.

Підкреслюється значні досягнення в галузі штучного інтелекту, такі як підвищена якість і деталізація вимог. У дослідженні обговорюються потенціал майбутніх розробок ШІ для інженерії вимог, наголошується на необхідності більш складних програм ШІ для подальшого вдосконалення процесів виявлення вимог і документування.

Автори як приклад наводять метод із використанням чат-бота ШІ з НЛП для взаємодії з користувачами для визначення вимог. Базу знань з різних галузей, таких як банківська справа, охорона здоров’я, розваги тощо пропонується інтегрувати з чат-ботом. Це гарантує, що чат-бот задасть необхідні запитання для формування адекватних, повних та чітких вимог. Взаємодіючи з людьми, чат-бот використовує методи вилучення сутностей, класифікації намірів і НЛП, щоб зрозуміти сутності та наміри у відповідях користувача. Виявлені вимоги розподіляються за допомогою класифікації на функціональні та нефункціональні використовуючи Метод Опорних Векторів (SVM) і Мульти-номіальний Наївний Байєсівський алгоритм (MNB). Ці дві техніки досягли 88% і 91% точності відповідно.

У висновку для авторів стало очевидно, що застосування ШІ в інженерії вимог має високий потенціал. Системи ШІ можуть допомагати інженерам фокусуватись на більш творчих процесах, після того, як багато дій з аналізу та класифікації вимог будуть автоматизовані такими системами, а також вся парадигма роботи зі створення

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

програмного забезпечення буде направлена на залучення інструментів ШІ на всіх етапах.

У роботі [7] представлено методологію для автономного виявлення та удосконалення вимог до програмного забезпечення за допомогою інтелектуальної математики. Він представляє Інтелектуальний інструмент для специфікацій автономного програмного забезпечення (ITASS), який фіксує вимоги та створює формальні специфікації. На підставі введених користувачем неформальних вимог, ITASS автоматично генерує формальні специфікації. Це досягається за допомогою передових технологій в області інтелектуальної математики та науки про програмне забезпечення. Процес включає аналіз системної архітектури, специфікації моделей і процесів. ITASS використовує такі математичні засоби, як алгебра концептів (concept algebra), системна алгебра (system algebra) та алгебра процесів реального часу (RTPA), щоб формалізувати структуру та поведінку програмного забезпечення. Після формулювання формальних специфікацій ITASS використовує їх для автоматичної генерації коду, що дозволяє створювати програмне забезпечення для мов програмування, таких як C++, Java, Python чи MATLAB.

У статті автори демонструють методологію ITASS та експериментують з проблемами розробки програмного забезпечення в реальному світі, підкреслюючи потенціал штучного інтелекту для автономного створення вимог до програмного забезпечення. У висновку також говориться про потенціал методології для подальшого розширення і застосування до більш складних систем і вимог у майбутньому, підкреслюючи її масштабованість та адаптивність.

Основна мета роботи [5] полягає у картографуванні стану досліджень та практики використання NLP в інженерії вимог шляхом огляду існуючих систем, виявлення дослідницьких прогалин та пропозиції майбутніх напрямків досліджень. Робота надає огляд того, як NLP може допомагати у завданнях, таких як виявлення, аналіз, специфікація та управління вимогами.

Для дослідження був використаний метод систематичного картографічного огляду, що включав комплексний огляд 404 досліджень між 1983 та 2019 роками. Ці дослідження були категоризовані залежно від їх дослідницького фокусу, використаних технологій NLP та їхнього внеску в галузь інженерії вимог. Методологія включала ідентифікацію відповідних цифрових бібліотек, формулювання стратегії пошуку та використання критеріїв включення та виключення для вибору відповідних досліджень.

Відзначається певний розрив між можливостями інструментів NLP та їх впровадженням у реальні умови RE, що вказує на бар'єри до практичного впровадження. Складність технологій NLP та спеціалізовані знання, необхідні для ефективного впровадження їх у контекстах інженерії вимог, є основними перешкодами.

Підкреслено, що, хоча існує значна активність і потенціал у галузі використання NLP для роботи з вимогами, потрібні більш цілеспрямовані зусилля для подолання розриву між теоретичними досягненнями та практичними застосуваннями. Запропоновано, що майбутні дослідження повинні прагнути розробляти більш дружні користувачам інструменти та проводити емпіричні дослідження для валідації ефективності технік NLP у різних галузевих контекстах.

3. Аналіз засобів штучного інтелекту для інженерії вимог

3.1. Можливості засобів ШІ

Сучасні інструменти ШІ, особливо ті, що базуються на системах обробки природної мови (NLP) та генерації тексту і даних (GPT), відіграють значну роль у вдосконаленні процесів інтерпретації та аналізу вимог до програмного забезпечення. Ці технології відкривають нові можливості для автоматизації та підвищення точності управління вимогами. NLP системи дозволяють комп'ютерам "розуміти" текстову інформацію, що вводиться людьми, аналізуючи при цьому смислове навантаження та контекст висловлювань. Це дуже важливо у контексті інженерії вимог, де вимоги часто формулюються невизначено або неоднозначно. GPT та LLM системи в свою чергу дозволяють генерувати необхідні дані у визначеному контексті.

За допомогою NLP систем потенційно можна ефективно виявляти потенційні суперечності та неоднозначності у вимогах, що є критично важливим для запобігання помилок на ранніх етапах розробки. Сучасні NLP системи мають високу точність у розпізнаванні та інтерпретації людської мови, що дозволяє забезпечити детальний та точний аналіз вимог [7].

Однією з ключових проблем у процесі інженерії вимог є розбіжності у розумінні та сприйнятті технічних термінів між технічними спеціалістами та представниками бізнесу. Часто бізнес-стейкхолдери можуть не повністю розуміти технічні аспекти проєкту, тоді як розробники можуть не враховувати всі бізнес-вимоги. Це може призвести до неправильної інтерпретації вимог, невідповідності очікувань та затримок у проєкті.

Системи, що базуються на технологіях NLP, можуть значно покращити комунікацію між технічними командами та бізнес-стейкхолдерами:

– переклад технічних термінів на "Бізнес-Мову": такі системи можуть автоматизувати процес перекладу технічних термінів та концепцій на мову, зрозумілу для бізнес-стейкхолдерів, сприяючи кращому розумінню та забезпеченню точності спілкування;

– автоматизація виявлення невідповідностей та протиріч: з використанням NLP можливо автоматизувати процес виявлення невідповідностей або неоднозначностей між технічними та бізнес-вимогами, що сприяє своєчасному усуненню розбіжностей;

– поліпшення документації та звітності: застосування NLP може покращити якість документації, роблячи її більш доступною та зрозумілою для всіх зацікавлених сторін, незалежно від їх технічного досвіду.

Використання NLP систем в процесі оцінювання та валідації вимог може значно підвищити ефективність комунікації між технічними командами та бізнес-стейкхолдерами, забезпечуючи більшу прозорість та сприяючи кінцевому успіху проєкту [8, 9].

Сучасні інструменти ШІ, особливо ті, що базуються на системах обробки природної мови (NLP) та генерації тексту і даних (GPT), відіграють значну роль у вдосконаленні процесів інтерпретації та аналізу вимог до програмного забезпечення. Ці технології відкривають нові можливості для автоматизації та підвищення точності управління вимогами. NLP системи дозволяють комп'ютерам "розуміти" текстову інформацію, що вводиться людьми, аналізуючи при цьому смислове навантаження та контекст висловлювань.

Це дуже важливо у контексті інженерії вимог, коли вони часто формулюються неоднозначно або недостатньо визначені. GPT та LLM системи, в свою чергу, дозволяють генерувати необхідні дані у визначеному контексті.

Сучасні NLP системи мають високу точність у розпізнаванні та інтерпретації людської мови, що дозволяє забезпечити детальний та точний аналіз вимог.

Однією з ключових проблем інженерії вимог є розбіжності у розумінні та сприйнятті технічних термінів між технічними спеціалістами та представниками бізнесу.

Часто бізнес-стейкхолдери можуть не повністю розуміти технічні аспекти проєкту, тоді як розробники можуть не враховувати всі бізнес-вимоги. Це може призвести до неправильної інтерпретації вимог, невідповідності очікувань та затримок у проєкті.

3.2. Аналіз сучасних засобів ШІ

Розробки, які спрямовано на використання інструментів ШІ у процесі створення програмного забезпечення, вже виконуються досить давно як в науковій, так і в прикладній сферах.

Проаналізуємо низку існуючих систем для управління проєктами, які також надають інструменти для документації вимог та іншої інформації.

Jira Software (Atlassian). Широко використовуваний інструмент для управління проєктами та вимогами, що забезпечує гнучкість і адаптивність:

– *переваги:* висока налаштованість, інтеграція з різними інструментами, підтримка Agile-методологій;

– *недоліки:* може бути складним у налаштуванні та використанні для новачків;

– *використання AI інструментів:* Jira сама по собі може не мати вбудованих AI-функцій, але її можна інтегрувати з різними AI-плагінами та зовнішніми інструментами. Наприклад, інтеграція з інструментами, які використовують машинне навчання для автоматизації завдань, аналізу продуктивності команди, або навіть для прогнозування термінів виконання проєктів.

IBM Rational DOORS. Інструмент управління вимогами, який забезпечує високий рівень управління складними проєктами:

– *переваги:* потужні функції управління вимогами, підтримка великих та складних проєктів;

– *недоліки:* висока вартість, складність у використанні;

– *використання ШІ-інструментів:* IBM Rational DOORS, особливо його новіші версії, може включати певні функції, що базуються на ШІ, хоча ці функції можуть бути більше зосереджені на інтеграції з іншими IBM-продуктами, які використовують ШІ, такими як Watson. Це може включати покращений аналіз даних та більш ефективне управління вимогами.

Trello. Простий і інтуїтивно зрозумілий інструмент для управління завданнями та вимогами:

– *переваги:* легкість у використанні, візуальна організація інформації;

– *недоліки:* обмежена функціональність для складного управління вимогами;

– *використання ШІ-інструментів:* хоча Trello не включає безпосередньо функцій штучного інтелекту, його можна інтегрувати з

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

ШІ-інструментами через API. Це дозволяє автоматизувати рутинні процеси, аналізувати ефективність роботи команди та навіть допомагати у прийнятті рішень на основі зібраних даних.

ReqSuite® RM (OSSENO Software). Сучасний інструмент для управління вимогами з підтримкою ШІ:

– *переваги:* інтеграція ШІ для покращення якості вимог, інтуїтивний інтерфейс;

– *недоліки:* може вимагати певного часу для освоєння;

– *використання ШІ-інструментів:* може включати функції на основі штучного інтелекту, які покращують управління вимогами. Це може включати автоматичний аналіз вимог, визначення пріоритетів та залежностей між вимогами, а також підтримку в прийнятті рішень.

Microsoft Azure DevOps. Комплексний інструмент, який включає управління вимогами, версійний контроль, засоби автоматизації та багато іншого:

– *переваги:* інтеграція з широким спектром інструментів Microsoft, гнучкість управління проєктами;

– *недоліки:* вимагає інтеграції з іншими інструментами Microsoft для повної функціональності;

– *використання ШІ-інструментів:* Microsoft Azure DevOps може використовувати AI для оптимізації різних аспектів управління проєктами та розроблення ПЗ, включаючи управління вимогами. Це може включати автоматизацію завдань, аналіз ефективності та навіть прогнозування ризиків.

Окремо варто відзначити **GPT-подібні системи з генерації коду** – на синтетичних тестах та використовуючи відкалібровані моделі вдається досягати точності понад 70%.

Такі інструменти як **Github Copilot** або **IntelliJ AI Assistant** вже досить давно і успішно допомагають розробникам у з написанні коду та тестів.

Інструменти і системи ШІ варіюються за своїми можливостями: від простих додатків для управління завданнями до складних систем, що підтримують великі проєкти.

Вибір конкретного інструменту залежить від специфіки проєкту, розміру команди, бюджету та інших факторів.

4. Розроблення і застосування інтелектуального чат-боту

4.1. Функції інтелектуальних чат-ботів

Одним з прикладів використання інструментів ШІ в інженерії вимог є чат-бот з використанням NLP інструментів та LLM моделей [3]. Чат-боти можуть бути використані для автоматизованого обміну інформацією та надання відповідей на запитання щодо вимог до проєкту наступним чином:

– **збір інформації про вимоги.** Чат-бот може почати спілкування з користувачем, запитуючи про деталі щодо вимог до проєкту. Він може задавати питання про функціональні та нефункціональні вимоги, пріоритети, обмеження та інші важливі аспекти;

– **пояснення вимог.** Чат-бот може надавати користувачеві пояснення щодо певних вимог або термінів, які є недостатньо зрозумілими. Він може надавати додаткові ресурси або посилання для докладного вивчення конкретних аспектів вимог;

– **автоматичне уточнення деталей.** Чат-бот може автоматично уточнювати деякі деталі вимог, якщо їхнє формулювання неоднозначне або недостатньо конкретне. Він може запитувати додаткові деталі або запропонувати альтернативні варіанти для уточнення вимог;

– **аналіз та класифікація вимог**. Чат-бот може використовувати штучний інтелект для аналізу та класифікації вимог зібраних під час спілкування з користувачем. Він може автоматично визначати типи вимог, їхню важливість та інші параметри для подальшої обробки;

– **генерація звітів та документації**. На основі інформації, зібраної в ході спілкування з користувачем, чат-бот може генерувати звіти та документацію щодо вимог до проєкту. Це може включати створення специфікацій вимог, таблиць пріоритетів та іншої необхідної документації.

Загалом, чат-боти можуть значно полегшити процес збирання, уточнення та документування вимог до проєкту, зменшуючи час та зусилля, які необхідно витратити на цей процес вручну.

Такий підхід також може покращити комунікацію між різними учасниками проєкту та забезпечити більшу чіткість та узгодженість вимог [3].

Оскільки створення нових моделей машинного навчання та систем чат ботів з нуля є дуже витратним та доволі складним процесом, пропонується розглянути побудову системи з існуючих компонентів.

Ринок інтелектуальних систем ботів та асистентів досить потужно представлений продуктами майже від кожної великої технологічної компанії.

4.2. Архітектура і реалізація інтелектуального чат-боту

Для реалізації концепту чат-боту вз використанням інструментів ШІ пропонується

вибрати один технологічний стек, оскільки це значно спрощує взаємодію між компонентами системи, розробку, підтримку та подальше вдосконалення продукту.

Оскільки найбільш повною лінійка продуктів, яку можна використати для вирішення завдання, є саме в Amazon AWS, його використання є цілком виправданим.

Однією з найбільших переваг є наявність двох систем для роботи з моделями машинного навчання. Отже, спроєктуючи відповідним чином систему, можемо на першому етапі використати більш простий сервіс Bedrock, а після збору та аналізу результатів – використовувати зібрані дані для детального навчання та використання своєї моделі за допомогою сервісу SageMaker [10].

Крім того, AWS пропонує доволі потужні інструментиі і надає можливості для моніторингу, логування і зберігання даних та подальшої обробки аналітичними сервісами.

Основними компонентами запропонованої системи є наступні:

– графічний інтерфейс чат боту для взаємодії з користувачами;

– Lex v2 чат-бот;

– Lambda сервіс для обробки зібраних даних;

– Bedrock сервіс для використання LLM моделей з предналаштованим запитом;

– база даних DynamoDB для зберігання результату;

– CloudWatch сервіс для логування та моніторингу.

Загальна діаграма взаємодії основних компонентів показана на рисунку 1. Процес роботи системи є наступним:

1. Користувач ініціює діалог з чат ботом через веб інтерфейс або мобільний додаток.
2. Чат-бот Lex, слідкуючи за налаштуванням діалогу, просить надати бажані характеристики системи: спочатку забезпечити функціональні характеристики, а потім визначити загальні вимоги та існуючі обмеження.
3. Lex сервіс отримує текстові дані від користувача, формуючи іх у власних слотах. Після кожного етапу сервіс передає дані слотів в Lambda функцію.
4. Lambda функція отримує дані і використовуючи API з бібліотеки внутрішніх сервісів формує запит до моделі Bedrock сервісу з вимогою групувати надану інформацію та валідувати її.
5. Структурований список від моделі передається у відповідь кожній функції.
6. Lex сервіс отримує відповідь від функції і передає її користувачу для перевірки. При необхідності вносяться правки и повторюються кроки 3-5 для доповнення даних.
7. Коли користувач підтверджує достатність згрупованих вимог, вони передаються на виклик іншої Lambda функції, яка зберігає дані у базу DynamoDB.
8. Діалог завершується.

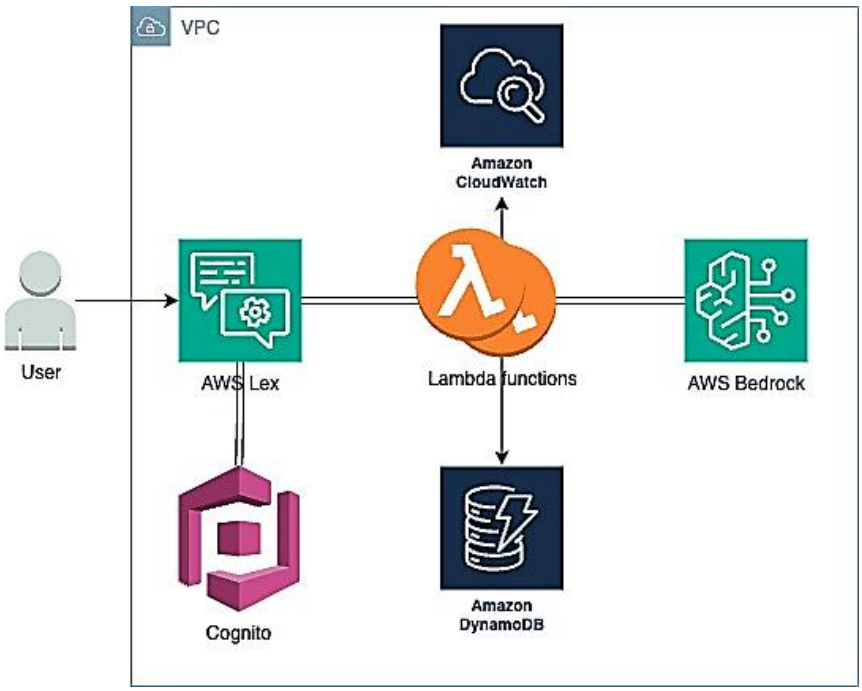

Рис. 1. Діаграма компонентів системи

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Для спрощення збір вимог можна обмежити виключно діалогом сервісу Lex з підключенням внутрішнього функціоналу генеративного ШІ для створення більш інтерактивного діалогу. Після цього на завершальному етапі отримані дані в Bedrock передаються для подальшої обробки і аналізу.

Для більш точного калібрування системи є можливість:

– експериментувати з налаштуваннями запиту до моделі;

– тестувати запити до різних наявних моделей;

– додавати свої дані для доповнення та тренування моделей.

Як вже зазначалось раніше, отримані дані можна використати для тренування своєї моделі через сервіс SageMaker, що буде вимагати більш значних витрат часу, але потенційно надавати більш точний результат.

Використання Lambda сервісу як проміжного компоненту дозволяє навіть використовувати інші сторонні сервіси ШІ як Open AI GPT або подібні.

Таким чином, можемо отримати дуже гнучку систему, яка буде адаптуватися під проєкти різного розміру та сфери застосування, що є дуже важливою перевагою.

Використання безсерверних компонентів надає широкі можливості до розширення та підтримки високих навантажень, а також мінімізує витрати на розробку власних сервісів.

4.3. Переваги і недоліки

Суттєвими перевагами запропонованої системи є наступні:

– система використовує безсерверні компоненти, такі як Lex, Lambda та DynamoDB, що робить її високомасштабованою. Зі збільшенням трафіку ці служби можуть автоматично обробляти навантаження без дадоткового втручання;

– використання компонентів AWS дозволяє гнучко інтегрувати інші сервіси. Можна розширити систему, додавши інші сервіси AWS, такі як API Gateway, SNS/SQS і сторонні інструменти за потреби;

– завдяки дотримання правил використання та IAM політик систему можна ефективно захистити. Lex пропонує автентифікацію через Amazon Cognito, а інші служби мають вбудоване шифрування;

– безсерверна архітектура несе витрати лише під час активної обробки даних;

– задіяння повністю керованих служб AWS прискорює розроблення, спрощує налаш-тування і обслуговування інфраструктури, дозволяє зосередитися на бізнес-логіці;

– використання потужних сервісів ШІ дозволяє вибирати з великої кількості вже готових до використання та налаштовувати моделі під свої умови застосування.

Серед недоліків розробленого чат-боту варто зазначити наступні:

– Lambda функції мають специфіку «прогріву» для першого запиту, і ліміти на паралельне виконання;

– дані зібрані від користувачів передаються в систему AWS, що має бути окремо пропрацьовано з точки зору захисту персональних даних;

– безсерверні системи дуже гнучкі, але потребують додаткової уваги у налаштуваннях та моніторингу.

Використання сервісів лише одного вендора значно ускладнює перехід до інших систем якщо виникне така потреба.

5. Висновки

5.1. Обговорення результатів

Дане дослідження демонструє значний потенціал використання інструментів штучного інтелекту в інженерії вимог. ШІ здатний трансформувати традиційні підходи, підвищуючи точність і ефективність процесів аналізу та документації вимог. Однак, необхідно подальше дослідження для оцінки довгострокової ефективності цих технологій у реальних умовах. Це вимагає тісної співпраці між практиками та науковцями для розробки інструментів, які будуть не тільки потужними, а й доступними та зрозумілими для всіх зацікавлених сторін.

Такий підхід не лише сприятиме успіху програмних проєктів, але й торуватиме шлях для новаторських вдосконалень у технологіях інженерії вимог. Успішне застосування методології використання інструментів ШІ дозволить також використовувати такі техніки і для складних інформаційних та інформаційно-керуючих систем, оскільки створення будь-якого продукту починається саме зі збору і аналізу вимог до нього.

Запропонований інтелектуальний чат-бот для збору та оцінювання вимог надає можливість в процесі діалогу між користувачем та ботом:

– збирати необхідні дані;

– групувати ці дані;

– проводити перевірку, використовуючи наявні моделі машинного навчання.

Отриманий результат згрупованих вимог зберігається для подальшої обробки. Архітектура проєкту забезпечує масштабованість, гнучкість у

налаштуванні та використанні найбільш доцільних інструментів.

5.2. Подальші дослідження і розробки

Основні напрями подальшої роботи можна розділити на дві групи:

1. Продовжити дослідження щодо використання систем з ШІ для автоматизації роботи з вимогами, а саме:

– визначення етапів життєвого циклу вимог та оцінка можливості застосування ШІ на кожному з них;

– визначення характеристик повноти вимог та оцінка ефективності за допомогою автоматизованих систем;

– доопрацювання представленого чат-боту для оцінки ефективності використання різних моделей та конфігурації запитів до компоненту ШІ;

– розроблення інструментів інтеграції комплексних систем з системами управління проєктами у вигляді програмних інтерфейсів та плагінів.

2. Використання інструментів ШІ для всіх етапів життєвого циклу програмного забезпечення:

– оцінювання можливості використання ШІ на кожному етапі життєвого циклу як окремого процесу, так і в якості комплексної системи;

– визначення можливостей для застосування ШІ як єдиної системи взаємодії всіх учасників процесу та як інструменту для збору вимог, аналізу дизайну системи, відслідковування процесу розробки, результатів тестування, контролю розгортання та підтримки кінцевого продукту.

Важливим є також доповнення процесів вбудовування засобів ШІ в процеси інженерії вимог оцінюванням якості цих засобів відповідно до моделей, запропонованих в [11], з використанням метричних методів [12].

Внесок авторів: проаналізував засоби штучного інтелекту, які використовуються в інженерії вимог, та їх можливості щодо підвищення ефективності процесів; розробив чат-бот для валідації вимог; підготував рукопис статті – **А.О. Стряпунін**; сформулював мету, задачі та методику досліджень, обґрунтував вимоги до чат-боту та напрями подальших розробок і досліджень; відредагував рукопис – **В. С. Харченко**.

Конфлікт інтересів

Автори заявляють, що немає конфлікту інтересів щодо цього дослідження, фінансового, особистого, авторського чи іншого, який міг би вплинути на дослідження та його результати, представлені в статті.

Фінансування

Дослідження проводилося без фінансової підтримки.

Доступність даних

Рукопис не має пов'язаних даних.

Використання засобів штучного інтелекту

Автори підтверджують, що не використовували технології штучного інтелекту при створенні представленої роботи.

Усі автори прочитали та погодилися з опублікованою версією рукопису.

Література

1. Mars Climate Orbiter, Mishap Investigation Board, Phase I Report, November 10, 1999 [Electronic resource]. – 48 p. – Available at: https://llis.nasa.gov/llis_lib/pdf/1009464main1_0641-mr.pdf. – 1.03.2024.

2. Arulmohan, S. Extracting Domain Models from Textual Requirements in the Era of Large Language Models [Text] / S. Arulmohan, M. - J. Meurs, & S. Mosser // 2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C). – Västerås, Sweden, 2023. – P. 580-587. DOI: 10.1109/MODELS-C59198.2023.00096.

3. Liu, K. Artificial Intelligence in Software Requirements Engineering: State-of-the-Art [Text] / K. Liu, S. Reddivari, & K. Reddivari // 2022 IEEE 23rd International Conference on Information Reuse and Integration for Data Science (IRI). – San Diego, CA, USA, 2022. – P. 106-111. DOI: 10.1109/IRI54793.2022.00034.

4. Chen, L. Characterizing Architecturally Significant Requirements [Text] / L. Chen, M. Ali Babar, & B. Nuseibeh // IEEE Software. – 2013. – Vol. 30, Iss. 2. – P. 38-45. DOI: 10.1109/MS.2012.174.

5. Natural language processing for requirements engineering: A systematic mapping study [Text] / L. Zhao, W. Alhoshan, A. Ferrari, K. J. Letsholo, M. A. Ajagbe, E.-V. Chioasca, & R. T. Batista-Navarro // ACM Computing Surveys (CSUR) – 2021. – Vol. 54, iss. 3. – Atricle no. 55. – P. 1-14. DOI: 10.1145/3444689.

6. Brackett, J. W. Software Requirements. SEI Curriculum Module SEI-CM-19-1.2 [Text] / J. W. Brackett. – Carnegie Mellon University, Software Engineering Institute, 1990. 27 p. Available at:

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

https://insights.sei.cmu.edu/documents/1547/1990_007_001_15809.pdf. – 1.03.2024.

*7. Wang, Y. Autonomous Software Requirement Specifications towards AI Programming [Text] / Y. Wang, & J. Y. Xu // 2021 IEEE 20th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC). – Banff, AB, Canada, 2021. – P. 123-130. DOI: 10.1109/ICCICC53683.2021.9811311.*

8. Arellano, A. Natural Language Processing of Textual Requirements [Text] / A. Arellano, E. Carney, & M. A. Austin // ICONS 2015 : The Tenth International Conference on Systems. – 2015. – P. 93-97. Available at: https://user.eng.umd.edu/~austin/reports.d/ICONS2015-AA-EC-MA.pdf. – 1.03.2024.

9. Natural Language Processing for Requirements Engineering: The Best Is Yet to Come [Text] / F. Dalpiaz, A. Ferrari, X. Franch, & C. Palomares // IEEE Software. – 2018. – Vol. 35, no. 5. – P. 115-119. DOI: 10.1109/MS.2018.3571242.

10. Vahidi, P. Information extraction with LLMs using Amazon SageMaker JumpStart [Electronic resource] / P. Vahidi, & R. Sharifpour // AWS Machine Learning Blog. – 2024. Available at: https://aws.amazon.com/blogs/machine-learning/information-extraction-with-llms-using-amazon-sagemaker-jumpstart. – 1.03.2024.

11. Kharchenko, V. Quality Models for Artificial Intelligence Systems: Characteristic-Based Approach, Development and Application [Text] /, V. Kharchenko, H. Fesenko, & O. Illiashenko // Sensors. – 2022. – Vol. 22, iss. 13. – Article no. 4865. – P. 1-36, DOI: 10.3390/s22134865.

12. Васильєв, І. Фреймворк для метричного оцінювання систем штучного інтелекту на основі моделі якості [Текст] / І. Васильєв, & В. Харченко // Системи управління, навігації та зв'язку. Збірник наукових праць. – Полтава : ПНТУ, 2022. – Т. 2(68). – С. 41-45. DOI: 10.26906/SUNZ.2022.2.041.

References

1. *Mars Climate Orbiter, Mishap Investigation Board, Phase I Report, November 10, 1999*. 48 p. Available at: https://llis.nasa.gov/llis_lib/pdf/1009464main1_0641-mr.pdf. (accessed 1.03.2024).

2. Arulmohan, S., Meurs, M. - J., & Mosser, S. Extracting Domain Models from Textual Requirements in the Era of Large Language Models. *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, Västerås, Sweden, 2023, pp. 580-587. DOI: 10.1109/MODELS-C59198.2023.00096.

3. Liu, K., Reddivari, S., & Reddivari, K. Artificial Intelligence in Software Requirements Engineering: State-of-the-Art. *2022 IEEE 23rd International Conference on Information Reuse and Integration for Data Science (IRI)*, San Diego, CA, USA, 2022, pp. 106-111. DOI: 10.1109/IRI54793.2022.00034.

4. Chen, L., Ali Babar, M., & Nuseibeh, B. Characterizing Architecturally Significant Requirements. *IEEE Software*, 2013, vol. 30, iss. 2, pp. 38-45. DOI: 10.1109/MS.2012.174.

5. 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 (CSUR)*, 2021, vol. 54, iss. 3, atricle no. 55, pp. 1-14. DOI: 10.1145/3444689.

6. Brackett, J. W. *Software Requirements. SEI Curriculum Module SEI-CM-19-1.2*, Carnegie Mellon University, Software Engineering Institute, 1990. 27 p. Available at: https://insights.sei.cmu.edu/documents/1547/1990_007_001_15809.pdf. (accessed 1.03.2024).

7. Wang, Y., & Xu, J. Y. Autonomous Software Requirement Specifications towards AI Programming. *2021 IEEE 20th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC)*, Banff, AB, Canada, 2021, pp. 123-130. DOI: 10.1109/ICCICC53683.2021.9811311.

8. Arellano, A., Carney, E., & Austin, M. A. Natural Language Processing of Textual Requirements. *ICONS 2015 : The Tenth International Conference on Systems*, 2015, pp. 93-97. Available at: https://user.eng.umd.edu/~austin/reports.d/ICONS2015-AA-EC-MA.pdf. (accessed 1.03.2024).

9. Dalpiaz, F., Ferrari, A., Franch, X., & Palomares, C. Natural Language Processing for Requirements Engineering: The Best Is Yet to Come. *IEEE Software*, 2018, vol. 35, no. 5, pp. 115-119. DOI: 10.1109/MS.2018.3571242.

10. Vahidi, P., & Sharifpour, R. I*nformation extraction with LLMs using Amazon SageMaker JumpStart*. AWS Machine Learning Blog, 2024. Available at: https://aws.amazon.com/blogs/machine-learning/information-extraction-with-llms-using-amazon-sagemaker-jumpstart. (accessed 1.03.2024).

11. Kharchenko, V., Fesenko, H., & Illiashenko, O. Quality Models for Artificial Intelligence Systems: Characteristic-Based Approach, Development and Application. *Sensors*, 2022, vol. 22, iss. 13, article no. 4865, pp. 1-36, DOI: 10.3390/s22134865.

12. Vasyl′yev, I., & Kharchenko, V. Freymvork dlya metrychnoho otsinyuvannya system shtuchnoho intelektu na osnovi modeli yakosti [A framework for metric evaluation of artificial intelligence systems based on quality model]. *Systemy upravlinnya, navihatsiyi ta zv"yazku. Zbirnyk naukovykh prats' – Control, navigation and communication systems. academic journal*, Poltava, PNTU Publ., 2022, vol. 2(68), pp. 41-45. DOI: 10.26906/SUNZ.2022.2.041.

Надійшла до редакції 20.02.2024, прийнята до опублікування 15.04.2024

USING AI TOOLS IN REQUIREMENTS ENGINEERING: ANALYSIS OF CAPABILITIES AND CHATBOT FOR VALIDATION

Anton Striapunin, Vyacheslav Kharchenko

This study investigates the processes and means of requirements engineering (RE) of software and complex information systems (SWS). The subject of this study is the instrumental means of SWS RE based on the methods of artificial (computational) intelligence (AI). **The goal** is to improve the accuracy and efficiency of CCD requirements development processes using AI tools by providing better communications between business teams and technical teams and automating complex requirements collection and documentation processes. **The tasks** are as follows: to analyze the principles and means of integrating AI tools into requirements engineering processes, the development and verification of which is a critical stage in the development of SWS; to determine the problems of using traditional SWS RE methods and perform their comparative analysis with AI-based methods; and to develop a chatbot architecture for requirements validation and identify characteristics of RE processes that are improved through its use. **The results.** The results of this study demonstrate that AI tools can significantly improve the accuracy, timeliness, and efficiency of requirements development processes and project team communication. An architecture and software solution for an intelligent chatbot for requirements validation is proposed, and the benefits and limitations of its application are discussed. Directions for further development and research regarding the end-to-end implementation of AI tools in IP processes have been developed.

Keywords: artificial intelligence; requirements engineering; team communication; chat-bot.

Стряпунін Антон Олександрович – магістрант комп'ютерних систем, мереж і кібербезпеки, Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна.

Харченко Вячеслав Сергійович – д-р техн. наук, проф., зав. каф. комп'ютерних систем, мереж і кібербезпеки, Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна.

Anton Striapunin – MSc-student, Department of Computer Systems, Networks and Cybersecurity, National Aerospace University "Kharkiv Aviation Institute", Kharkiv, Ukraine,
e-mail: a.o.striapunin@student.csn.khai.edu

Vyacheslav Kharchenko – Doctor of Technical Science, Professor, Head of the Department of Computer Systems, Networks and Cybersecurity, National Aerospace University "Kharkiv Aviation Institute", Kharkiv, Ukraine,
e-mail: v.kharchenko@csn.khai.edu, ORCID: 0000-0001-5352-077X, Scopus Author ID: 22034616000.