ARTIFICIAL INTELLIGENCE IN THE CYBERSECURITY FIELD

DOI 10.20535/2411-1031.2024.12.2.315741
УДК 004(89+4’2)

ВОЛОДИМИР СОКОЛОВ,
ВЯЧЕСЛАВ РЯБЦЕВ,
ОЛЕКСАНДР УСПЕНСЬКИЙ,
ДАНИЛО КОПИЧ

НАПРЯМИ ЗАСТОСУВАННЯ ШТУЧНОГО ІНТЕЛЕКТУ В ТЕХНОЛОГІЯХ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В статті представлено результати систематизованого аналізу сучасного стану застосування штучного інтелекту (ШІ) в інженерії програмного забезпечення (ПЗ) на основі аналізу публікацій, опису можливостей ШІ, досвіду його застосування та проведених експериментів. Сформовано концептуальні засади дослідження, які визначають: сприйняття ШІ як інструменту, а не суб’єкту праці; основні напрями його застосування – це інженерія та менеджмент; предметом застосування ШІ є обробка артефактів (синтез та аналіз) та отримання консультацій; підкреслено необхідність оцінки якості продуктів, отриманих від ШІ, та аналізу ризиків його застосування. В якості напрямів застосування ШІ в менеджменті визначено: процеси угоди (розробка концепції продукту та створення контракту), організаційні процеси (формування проєктної групи та вибір технологій) та управління проєктом (планування, управління ризиками, контроль та аналіз виконання проєкту) В якості напрямів застосування ШІ в інженерії розглянуто: управління вимогами, проєктування, конструювання, тестування та документування. Для систематизації аналізу напрямів застосування ШІ було розроблено концептуальну модель, яка включає: напрям, предмет та режим застосування ШІ. Режим застосування ШІ включає: формат промпту (постановка задачі та набір вхідних даних), потрібний продукт (результат) та його тип (готовий продукт, прототип, шаблон, варіанти рішень, інформаційна підтримка), роль ШІ (виконавець, співавтор, консультант), форму взаємодії зі ШІ (зовнішній сервіс, інтеграція через API, інтегрована система або локальна автономна система). На основі концептуальної моделі сформовано структуру похідних моделей для аналізу застосування ШІ по конкретних напрямках з оглядом можливостей найбільш ефективних інструментів ШІ. В якості висновків визначено, що в напрямках менеджменту найбільш раціональною моделлю застосування ШІ є отримання консультацій та прототипів документації при звертанні до зовнішніх сервісів ШІ, в інженерії – створення прототипів проєктних рішень та прототипів документації на основі зовнішніх сервісів, застосування інтегрованих систем ШІ для конструювання та тестування в режимі співавторства. В якості ризиків застосування ШІ визначено можливість отримання недостатньо деталізованої документації, складних і заплутаних програмних артефактів та помилок в програмному коді. Для зниження ризиків і підвищення ефективності застосування ШІ визначено, що потрібен постійний контроль якості його продуктів та навчання на основі корпоративних вимог та стандартів.

Ключові слова: штучний інтелект, технології розробки програмного забезпечення, інженерія програмного забезпечення, менеджмент ІТ-проєктів, модель застосування штучного інтелекту.

Постановка проблеми. Незважаючи на наявність ефективних інструментів, розробка ПЗ залишається складним інтелектуальним процесом, що поєднує науку, техніку і мистецтво в єдиний творчий процес. Бурхливий розвиток інструментів ШІ та їх активне застосування в інженерії ПЗ призвело до технологічної революції в цій галузі, що суттєво вплинуло на ринок

праці. Поява інструментів ШІ відкриває нові можливості його застосування як для вирішення інтелектуальних, так і рутинних задач, а також, можливо, і для покращення бізнес-процесів під час розробки ПЗ. Різке підвищення продуктивності праці розробників ПЗ призвело до зменшення потреб у кількості працівників, і ця тенденція буде зберігатися допоки не відбудеться усвідомлення ролі ШІ та перерозподілу функцій між людиною та ШІ в технологіях розробки програмного забезпечення (ТРПЗ).

Тому усвідомлення феномену ШІ, його ролі та місця в ТРПЗ, визначення напрямів та моделей ефективного застосування інструментів ШІ є актуальною проблемою.

Аналіз останніх досліджень та публікацій. Сучасні дослідження із застосування ШІ в інженерії ПЗ спрямовані як на вирішення чисто технічних задач, таких як генерація коду, так і на побудову сценаріїв використання ШІ в усіх процесах життєвого циклу (ЖЦ) ПЗ.

В [1] розглядаються чотири сценарії взаємодії між ролями людей, інструментами, ШІ та процесами розробки ПЗ. *Перший сценарій* є традиційним, без застосування ШІ. *Другий сценарій*: люди домінують над ШІ, який використовується для автоматизації окремих частин ручних і повторюваних завдань, таких як написання коду, документування, тестування та розгортання. ШІ також використовується для допомоги людям у таких завданнях, як дизайн, виправлення помилок і прийняття рішень. *Третій сценарій*: ШІ починає виконувати окремі ролі, такі як управління процесами, проєктування, реалізація, тестування, встановлення та обслуговування, а люди зосереджуються на найбільш складних завданнях та виконують контроль за всіма операціями для отримання високоякісних результатів. *Четвертий сценарій*: ШІ керує операціями розробки в різних ролях, а люди спостерігають і контролюють процеси, зосереджуючись на оперативному контролі, вирішенні проблем, забезпечення якості та безпеки. Ці сценарії розглядаються як найближчі перспективні напрями застосування ШІ в розробці ПЗ.

В [2] розглядається застосування ШІ в традиційному циклі розробки ПЗ, що включає планування проєкту, аналіз завдань, проєктування, реалізацію, тестування, впровадження та супровід. Зроблено висновок про загальну ефективність застосування ШІ в розробці ПЗ, звільнення людей від рутинних задач для більш креативної праці та необхідність адаптації працівників галузі до активного застосування ШІ.

В [3] розглядається застосування ШІ за наступними напрямами: управління проєктом; управління вимогами; управління людськими ресурсами та розподіл завдань; генерація та оптимізація коду; пошук помилок в коді; тестування; впровадження. Зроблено висновки про те, що інтеграція ШІ в розробку ПЗ може революціонізувати та покращити якість та ефективність процесів. При цьому, впровадження ШІ в інженерію ПЗ створює проблеми, такі як складність інтеграції, етичні міркування та потреба в спеціальних навичках. Потрібні подальші дослідження для вивчення довгострокового впливу ШІ на методи розробки ПЗ, стандарти інтеграції ШІ та оцінка етичних та суспільних наслідків розробки ПЗ, керованого ШІ.

В [4] розглядається приклад інструменту AutoDev на основі OpenAI GPT-4, реалізованого з використанням технології кооперації агентів, інтегрованого в систему програмування Visual Studio Code. Він автоматизує процеси генерації коду та тестів та досягає високих результатів якості коду при тестуванні на наборі даних HumanEval (близько 90%).

В [5] міститься звіт про промисловий досвід кодування за допомогою ШІ в професійній розробці ПЗ, в якому зазначено, що найбільш популярними інструментами ШІ є: веб-ресурс ChatGPT Plus, OpenAI API як програмний інтерфейс до GPT-4, сервіс GitHub Copilot, реалізований як плагін в Visual Studio Code, JetBrain та інші середовища програмування, а також GitHub Copilot X як попередня закрита версія з можливостями не тільки доповнення коду, а й спілкування в режимі чату. Аналізувались наступні напрями застосування ШІ: написання нового коду, супровід існуючого коду, переклад коду на іншу мову, модульне тестування, написання документації, задачі підтримки. Зроблено наступні висновки: ретельна

розробка промптів є суттєвою для ефективної генерації коду, якість згенерованого коду залежить від мови програмування (для Go – краще, для Python – гірше), ШІ підвищує ефективність розробки (у 2 – 10 разів) за рахунок швидкої генерації тривіального коду і зменшення зусиль на застосування незнайомих технологій, а для складного коду підвищення ефективності складає 20-30%, ШІ загалом покращує якість коду, але за рахунок декількох ітерацій його генерації, а при втраті контексту може генеруватися код з помилками, тому код має ретельно контролюватись людиною.

Робота [6] містить глибокий огляд застосування 30 великих мовних моделей (LLM) в інженерії ПЗ включно з описом архітектури моделей, принципів їх навчання та детальним аналізом їх застосування по 43 задачах в чотирьох найважливіших фазах інженерії ПЗ (управління вимогами та проєктування, розробка, тестування, супровід). Зроблено висновки, що великі мовні моделі вносять значні зміни в численні практики та підходи інженерії ПЗ завдяки їх здатності обробляти складні завдання, пов’язані з програмним кодом.

В роботі [7] порівнюється ефективність інструментів ШІ для генерації коду GitHub Copilot, Amazon CodeWhisperer та OpenAI ChatGPT на основі показників якості коду, таких як коректність, надійність, безпека, та придатність до супроводу, а також визначення їх переваг та недоліків. Визначено, що ChatGPT при тестуванні на наборі даних HumanEval має найкращі показники коректності коду.

Робота [8] аналізує області, де ChatGPT може бути ефективно застосований, такі як визначення неякісного коду, рефакторинг, генерація, покрокове виконання вихідного коду, виправлення помилок, покращення розуміння коду та аналіз коду. Зроблено висновки, що ChatGPT має значний потенціал щодо використання в інженерії ПЗ. Разом з тим визначено обмеження ChatGPT, такі як проблеми з генерацією коду зі складною логікою, обмежені знання мов програмування, обмежені математичні та наукові знання, проблеми з обробкою кількісних величин (наприклад, в різних одиницях вимірювання).

Стаття [9] присвячена аналізу структур шаблонів промптів для покращення якості відповідей ChatGPT при його застосуванні в інженерії ПЗ. Розглядаються шаблони промптів для виявлення вимог, проектування системи та моделювання, забезпечення якості коду та рефакторингу. Зроблено висновок, що структуровані промпти покращують якість відповідей ChatGPT та знижують кількість помилок, але необхідна експертиза відповідей з боку людини.

В роботі [10] проведено порівняльний аналіз продуктивності ChatGPT 3.5 щодо генерації коду для мов програмування C, C++, Go, Javascript, Julia, Perl, Python, R, Ruby та Smalltalk. Визначено, що ChatGPT має здатність генерувати код для різних завдань, але ефективність генерації коду залежить від мови та рівня задачі, причому в деяких випадках може генеруватися код з помилками або взагалі не генеруватись.

В [11] проведено емпіричне дослідження для вивчення можливостей ChatGPT для аналізу коду, його здатності розуміти синтаксис, статичну та динамічну поведінку коду, написаного чотирма мовами програмування: Python, Java, C та Solidity. Результати дослідження показують, що ChatGPT здатний розуміти правила синтаксису коду та має певні здатності зрозуміти статичну структуру, але не так добре розуміє динамічну поведінку коду.

В [12] розглядаються можливості проєктування інтерфейсу користувача на основі ШІ. Зроблено висновок, що аналізуючи поведінку та вподобання користувачів web-ресурсів, ШІ може персоналізувати діалог для покращення взаємодії з користувачем з урахуванням його вподобань. Крім того, ШІ може оптимізувати процес проєктування, прискорити розробку і підвищити ефективність інтерфейсу.

В [13] вивчається використання OpenAI ChatGPT, GitHub Copilot, Google Bard/Gemini та Microsoft Bing Chat для допомоги у розробці ПЗ. Це включає аналіз вимог, створення документації і звітів, а також написання історій в системі управління проєктами Jira (наприклад, про помилки та проблеми). Визначено, що одними з важливих мотивів застосування ШІ є заощадження часу та коштів, підвищення продуктивності праці та безпеки

коду за рахунок заміни вартісних сканерів безпеки на ШІ, а також підвищення кваліфікації за рахунок навчання у ШІ новим технологіям.

Таким чином, аналіз останніх публікацій свідчить про актуальність досліджень можливостей ШІ та напрямів його застосування в інженерії ПЗ.

Метою статті є систематизація напрямів та визначення моделей ефективного застосування ШІ в ТРПЗ з урахуванням сучасного стану та перспектив розвитку інструментів ШІ.

Виклад основного матеріалу дослідження. Для проведення дослідження було сформовано методичний підхід з урахуванням процесів ЖЦ [14] та показників якості [15].

Концептуальні засади дослідження. В процесі дослідження було сформовано декілька постулатів щодо суті та ролі ШІ в ТРПЗ, які допомогають у визначенні напрямів та моделей його застосування. Нижче наведений зміст цих постулатів.

1. *ШІ – це інструмент, а не суб'єкт праці*. З появою інструментів ШІ не припиняються спроби його уособлення (персоніфікації, надання йому людських рис) та сприйняття його в якості суб'єкту, який є членом колективу людей. Автори вважають, що ШІ є саме інструментом і він не може вважатися членом команди розробників ПЗ як самостійний суб'єкт, хоча й може залучатися для виконання окремих завдань, притаманних певній ролі людини в проєктній групі. Загальний принцип відповідальності людини за прийняті рішення та отриманий результат виключає ШІ як самостійного виконавця, якому можна доручити цілісну задачу без контролю та затвердження людиною отриманого від ШІ результату. ШІ також не може вважатися автором програмних артефактів (включаючи принцип інтелектуальної власності), навіть якщо він використовується як кодогенератор, що є давно відомою функцією, притаманною таким інструментам, як компілятори, дизасемблери, декомпілятори, генератори коду класів з діаграм класів UML тощо.

2. *ШІ має обмежену креативність*. Враховуючи, що інструменти ШІ навчені на існуючій інформаційні базі, то і креативні здібності ШІ обмежуються пошуком рішень шляхом комбінації існуючих. Враховуючи неформальний закон збереження інформації та алгоритмічної складності, не варто доручати ШІ генерацію нових ідей або пошук інноваційних рішень складних проблем.

3. *Основні напрями застосування ШІ в ТРПЗ* – це інженерія та менеджмент. Всі процеси ЖЦ ПЗ можна розбити на дві категорії: інженерія (група процесів технічного характеру з безпосереднього створення ПЗ) та менеджмент (група процесів управління проєктом).

4. *Предметом застосування ШІ* є обробка артефактів (їх синтез та аналіз) та консультування (надання консультацій). Відповідно до напрямів застосування ШІ такими артефактами є артефакти інженерії (програмні артефакти, технічна документація) та артефакти менеджменту (договірна документація, організаційно-розпорядча документація, плани, управлінські рішення тощо). Консультації – це експертні поради та рекомендації стосовно виконання робіт з синтезу або аналізу артефактів без безпосереднього виконання цих робіт (передбачається, що їх буде виконувати людина). Відповідно, *продуктом ШІ* є синтезовані артефакти, результати аналізу артефактів та консультації.

5. *Якість продуктів ШІ потребує контролю*. В залежності від напрямів та предмету застосування, інструменти ШІ демонструють різну якість результатів, що потребує залучення експертів для їх оцінки. Якість результатів ШІ залежить від якості та кількості даних, на яких він був навчений, повноти вхідних даних промптів, а також від контексту застосування. Відомо, що ШІ добре справляється з рутинними та шаблонними завданнями, такими як автодоповнення коду, автоматизоване тестування та створення базової документації, а стосовно прийняття важливих рішень та виконання творчих завдань ШІ потребує ретельного контролю, оскільки може пропонувати помилкові або неповні рішення.

6. *Застосування ШІ несе ризики*. Переоцінка можливостей ШІ та безконтрольне прийняття отриманих продуктів несе безліч ризиків, пов'язаних з якістю його роботи,

використанням застарілих або неефективних рішень, прийняттям хибних управлінських рішень тощо. Тому при застосуванні ШІ потрібно усвідомлювати такі ризики та критично ставитися до його продуктів.

7. Одним зі *шляхів зниження ризиків* та підвищення ефективності застосування інструментів ШІ в ТРПЗ є використання його здатності до навчання в процесі застосування (притаманна не всім інструментам), що дозволяє адаптувати та підвищувати кваліфікацію ШІ шляхом його навчання (донавчання) на корпоративних стандартах та прикладах реалізації проєктів певної компанії-розробника ПЗ.

Моделі застосування ШІ в ТРПЗ. Для структуризації досліджень було розроблено концептуальну модель *CM* застосування ШІ в ТРПЗ, яка виглядає наступним чином

$$CM = \langle D, S, M \rangle, \quad (1)$$

де D – напрям;

S – предмет;

M – режим застосування ШІ в ТРПЗ.

Розглянемо більш детально концептуальну модель *CM* та її складові.

Напрям D застосування ШІ в ТРПЗ визначає процес або групу процесів ЖЦ ПЗ, об'єднаних спільним видом діяльності, в якій застосовується ШІ. Виділено наступні напрями.

1. *Менеджмент* – сукупність процесів ЖЦ з організації, планування, контролю та координації всіх етапів і ресурсів проєкту для досягнення його цілей у визначені терміни з необхідною якістю та в межах бюджету. Для аналізу було обрано як організаційні процеси, так і процеси технічного менеджменту, для автоматизації яких застосування ШІ є можливим та доцільним. Розглядаються наступні групи процесів та окремі процеси менеджменту:

1.1. *Процеси угоди*: розробка концепції продукту; створення контракту.

1.2. *Організаційні процеси*: формування складу виконавців проєктної групи; вибір технологій для реалізації проєкту.

1.3. *Управління проєктом*: планування проєкту; ризик-менеджмент; контроль та аналіз виконання проєкту.

2. *Інженерія* – сукупність процесів ЖЦ, що безпосередньо пов'язані зі створенням ПЗ. Було обрано наступні процеси для аналізу:

2.1. *Управління вимогами*: формування вимог; аналіз вимог; створення технічного завдання (ТЗ); створення специфікацій вимог (Software Requirements Specification, SRS).

2.2. *Проєктування*: проєктування архітектури; проєктування БД; розробка інтерфейсів.

2.3. *Конструювання*: написання коду; аудит, аналіз та рефакторинг коду; створення та аналіз запитів до БД; створення інсталяційного пакету.

2.4. *Тестування*: розробка тестів; автоматизоване тестування; аналіз результатів тестування.

2.5. *Документування* – створення проєктної документації.

Предмет S застосування ШІ в ТРПЗ описує завдання або мету застосування ШІ. Виділено наступні узагальнені завдання для ШІ:

– *генерація артефакту* – предметом застосування є синтез певного артефакту менеджменту (документу, плану, тощо) або інженерії (програмного коду, тестів, моделей тощо), коли результатом є новий артефакт;

– *аналіз артефакту* – предметом застосування ШІ є аналіз артефакту з певною метою (оцінка вартості та часу реалізації проєкту, аудит безпеки та рефакторинг коду тощо), коли результатом є певні висновки щодо якості, відповідності вимогам, оцінки, показники та рекомендації щодо покращення артефакту або суті того, що він представляє;

– *консалтинг* – предметом застосування ШІ є отримання консультацій (рекомендацій, порад, довідок, варіантів рішень, навчальних матеріалів) за певним напрямом.

Режим M застосування ШІ визначає характеристики конкретного звертання до ШІ:

– *формат промпту* – набір вхідних даних запиту, що може включати загальну постановку задачі та всі необхідні дані (інструкції для генерації коду, вимоги, артефакти для аналізу, шаблони документів для синтезу відповіді тощо); для інтегрованих доповнювачів коду промптом може вважатись поточний рядок коду, коментар, визначення методу або вихідний файл програми;

– *продукт* – результат застосування ШІ (відповідь на промпт), який може бути формальним (структурованим по заданій формі) або неформальним (у довільній формі, наприклад, консультація):

- згенерований артефакт;
- результати аналізу артефакту (похідний артефакт);
- консультація (поради щодо генерації та аналізу артефактів, варіанти рішень);

– *тип продукту*, який повертає ШІ:

- готовий продукт – повне рішення задачі або підзадачі, яке використовується “як є” без додаткового редагування;
- прототип – часткове або рамкове рішення для подальшого опрацювання;
- шаблон – продукт, який можна багаторазово застосовувати як основу для подальших подібних завдань чи проектів;
- варіанти рішень – результат включає декілька варіантів рішень для вибору;
- інформаційна підтримка – результат не містить конкретного рішення, але надає поради, рекомендації, довідкові та навчальні матеріали для отримання результату;

– *роль* – визначає, в якій якості застосовується ШІ:

- виконавець – повертає готовий продукт прийнятної якості, який може застосовуватись “як є” без додаткових корегувань;
- співавтор – повертає елементи рішення, які після доопрацювання можна використати як кінцевий продукт;
- консультант – не розв’язує задачу, а тільки надає консультації щодо шляхів та способів її розв’язку;

– *форма взаємодії* зі ШІ визначає технологічний аспект використання конкретного інструменту ШІ:

- ШІ як зовнішній сервіс – інструмент ШІ використовується як окрема система з web-інтерфейсом;
- інтеграція через API – ШІ є частиною автоматизованих робочих процесів, до яких можна звертатися через API (без необхідності прямого доступу до інтерфейсу системи) з іншого ПЗ, наприклад, для автоматизації запитів на обробку даних, генерації контенту чи коду;
- інтегрована система – ШІ є безпосередньо вбудованим у ПЗ, яким користується команда, та функціонує як складова частина робочого середовища (наприклад, інтелектуальний помічник, інтегрований в IDE для розробників);
- локальна автономна система – ШІ розгорнуто локально в межах інфраструктури проєкту або компанії, де він працює як автономний або у приватній мережі для забезпечення конфіденційності та швидкості доступу.

Концептуальна модель (1) використовується для формування та аналізу конкретних моделей застосування ШІ за кожним напрямом.

Модель M застосування ШІ певного напряму *D* має наступну форму

$$M_D = \langle S, P, T, R, F \rangle, \tag{2}$$

де S – предмет застосування ШІ;

P – формат промпту;

T – тип результату;

R – роль ШІ;

F – форма взаємодії зі ШІ.

Отримані результати. З огляду на виділені основні напрями застосування ШІ та концептуальну модель (1), отримані результати згруповані за напрямами та їх процесами і викладаються відповідно до окремих елементів загальної структури моделі (2).

1. Напрями та моделі застосування ШІ в процесах менеджменту

1.1. Процеси угоди. Розглядаються можливості ШІ для створення договірної документації.

Розробка концепції продукту. ШІ може допомогти у створенні концепції продукту як прототипу цього документу. Промпт має містити основні положення, такі як назву системи та її призначення, бізнес-ціль, основні задачі та функції системи, склад інформаційної бази, категорії користувачів та технічні вимоги.

На сьогодні найбільш ефективними при розробці концепції програмного продукту є такі інструменти ШІ (зовнішні сервіси).

ChatGPT: може допомогти з формулюванням концепції продукту, визначенням бізнес-цілей, основних функцій, користувачів, бізнес-вимог та згенерувати прототип концепції.

Jasper: підходить для формулювання ідей продукту, мети та ключових аспектів концепції.

Notion AI: підтримує генерацію ідей та описів для концепцій, а також допомагає структурувати їх в зручному форматі.

Copy.ai: добре підходить для створення базових концепцій продукту та бізнес-вимог.

Scalenut: може допомогти у визначенні ключових особливостей і функціональності продукту на основі аналізу потреб користувачів.

Створення контракту. ШІ можна застосовувати для генерування прототипу контракту на створення ПЗ або для аналізу контракту. Промпт для генерації контракту має містити:

- предмет договору;
- назви сторін;
- вартість та строки виконання договору;
- концепцію продукту.

Серед інструментів ШІ для створення контракту добре себе зарекомендували наступні.

Juro: спеціалізується на створенні юридичних документів, контрактів і має можливості налаштування шаблонів для контрактів, зокрема з додаванням спеціальних умов.

Ironclad: призначений для створення юридичних документів, автоматизації контрактів і є зручним для командної роботи.

LegalRobot: використовує ШІ для створення та перевірки контрактів, пропонує юридичні шаблони та полегшує узгодження контрактів.

Clause: інструмент, який дозволяє створювати та управляти контрактами, інтегруючи динамічні умови і положення (підходить для підтримки автоматизованих і адаптивних умов).

PactSafe: допомагає створювати прості контракти та автоматизувати їх підписання.

Microsoft Copilot (зовнішній сервіс або інтегрований у Word або Excel): здатен інтегруватися з документами, допомагаючи формувати вимоги і компоненти контракту, враховуючи стандарти й шаблони.

Kira Systems: орієнтований на аналіз і генерацію юридичних документів, а також підходить для розробки концептуальних документів на основі шаблонів.

1.2. Організаційні процеси. Організаційні процеси часто пов’язані з прийняттям управлінських рішень на основі множини вхідних даних та базуються на досвіді людей і важко піддаються автоматизації. Тому інструменти ШІ можуть застосовуватись в основному в режимі отримання консультацій або прототипів рішень.

Формування команди – це процес вибору членів команди з необхідними навичками для реалізації проєктних завдань. ШІ може аналізувати компетенції та навички учасників команди, їхню продуктивність і робочі потоки для підбору оптимального складу команди для проєкту.

Інструменти ШІ можуть здійснювати пошук потрібних кандидатів, використовуючи професійні соціальні мережі та відкриті бази даних пошуку роботи. До таких інструментів належать перелічені нижче.

HireEZ: інструмент для підбору персоналу, який аналізує професійні навички та досвід кандидатів, щоб визначити найкращих фахівців для роботи в команді.

LinkedIn Talent Insights: використовує ШІ для аналізу наявних фахівців і створення оптимального складу команди на основі їхнього досвіду та навичок.

Toggl Hire: платформа з вбудованим ШІ для оцінки технічних навичок кандидатів і створення персоналізованого профілю команди для проєкту.

Для підбору складу команди зазвичай потрібно ввести такі дані:

– *інформація про проект*: тип і масштаб проекту, бізнес-вимоги, часові рамки, бюджет, технологічні вимоги;

– *профілі необхідних ролей*: перелік ролей та компетенцій, рівень кваліфікації, ключові відповідальності;

– *методологія та інструменти розробки ПЗ*;

– *особисті характеристики та культурні особливості*: soft skills, культурна сумісність, географічні вимоги та часовий пояс.

Ці дані допоможуть інструментам ШІ створити оптимальну команду, яка відповідає як вимогам проекту, так і культурним та організаційним потребам.

Вибір технологій для реалізації проєкту як правило включає вибір власне методології (наприклад, Scrum або Kanban), інструментів та мов програмування, а також інфраструктури командної роботи над проєктом. Інструменти ШІ допомагають на основі даних про вимоги до проєкту, бюджету та часу реалізації підібрати найкращі технології для роботи. Найчастіше на сьогодні застосовуються наведені нижче інструменти.

Codex (*GitHub Copilot*): може надавати рекомендації щодо найкращих мов програмування та бібліотек для реалізації функцій або завдань на основі вимог проєкту.

Crystal: інструмент, що використовує ШІ для аналізу даних про бізнес та проєкти з метою підбору оптимальних технологій та рішень для досягнення бізнес-мети.

TARA AI: платформа для управління проєктами, що допомагає визначити технології, оцінити ресурси та терміни на основі аналізу минулих проєктів і технологічних трендів. Використовує машинне навчання для аналізу попередніх проєктів і вимог до нового проєкту, пропонуючи відповідні технології та інструменти для його реалізації.

AWS Well-Architected Tool: платформа від Amazon для створення оптимальних архітектур рішень у хмарі. Вона аналізує вимоги до проєкту та пропонує найкращі технології для використання у хмарі (мікросервіси, безсерверні обчислення, контейнеризація тощо).

Формат промпту при виборі технології має включати:

– функціональні вимоги до проєкту;

– вимоги до масштабованості, надійності та продуктивності;

– наявність та досвід команди у певних технологіях;

– тип середовища розгортання (локальний сервер, хмара, контейнеризація тощо).

1.3. Управління проєктом. ШІ можна використовувати для оцінки вартості та термінів виконання проєкту, планування ресурсів, розробки плану та контролю його виконання, управління ризиками тощо. Інструменти управління проєктом зі ШІ поєднують функції підбору команди, оцінки вартості та термінів, а також вибору технологій. На даний час накопичений позитивний досвід застосування наведених нижче засобів ШІ.

Jira з використанням ШІ: інструменти Jira для управління проєктами інтегруються з ШІ, що допомагає прогнозувати терміни, визначати пріоритети завдань та автоматизувати процеси.

Trello AI Integration: використовує ШІ для автоматизації задач і оптимізації робіт.

Monday.com: інструмент для управління проєктами, який надає командам можливості для планування, організації та відстеження робочих процесів у проєктах розробки ПЗ.

Wrike: платформа для управління проєктами, що пропонує широкі можливості для організації та відстеження проєктів, зокрема в сфері розробки ПЗ.

Smartsheet зі ШІ: платформа для управління проєктами, яка використовує ШІ для оцінки ресурсів, планування командної роботи, прогнозування ризиків та вибору технологій.

Планування проєкту. Інструменти ШІ з планування проєкту можуть використовувати історичні дані, аналітику продуктивності та оцінки завдань для прогнозування вартості проєкту, ресурсів та часу його реалізації.

Forecast: платформа з підтримкою ШІ для планування ресурсів і проєктного менеджменту, яка використовує машинне навчання для оцінки термінів виконання та ресурсів для проєкту на основі попередніх даних.

Pricefx: інструмент ШІ для оцінки вартості проєкту, що аналізує ринкові умови, витрати на ресурси та персонал.

LiquidPlanner: прогнозує час виконання проєктів, використовуючи історичні дані про попередні проєкти, також оптимізує робочі процеси на основі реальних даних, допомагаючи визначати найкращі підходи та технології.

Ризик-менеджмент. Інструменти ШІ здатні оцінювати ризики, базуючись на історичних даних попередніх проєктів і допомагають спрогнозувати успішність реалізації проєкту.

Keen.io: використовує ШІ для аналітики продуктивності команди, передбачає потенційні ризики та підказує, як мінімізувати можливі проблеми в ході реалізації проєкту.

Proggio: платформа, що використовує ШІ для прогнозування результатів проєктів, допомагає оцінювати ризики та визначати ключові області для зосередження зусиль.

IBM OpenPages with Watson: дозволяє автоматизувати управління ризиками через аналіз великих масивів даних.

Формат промпту для ризик менеджменту має містити:

– дані про проєкт;
– поточні ризики;
– дані про залежності та обмеження;
– показники продуктивності та поточного стану робіт.

Продуктом ШІ є звіти для керівництва, які містять найважливіші ризики, запропоновані стратегії для їх уникнення, поточний прогрес по відношенню до термінів та бюджету, а також динаміку ризиків у часі.

Контроль та аналіз виконання проєкту. Як правило інструменти ШІ для управління проєктом містять відповідний функціонал щодо контролю виконання проєкту. Додатково можна вказати такі інструменти.

Targetprocess: використовує ШІ для управління в рамках Agile-методології, допомагаючи відслідковувати прогрес виконання проєктів, задач та етапів.

Trello + Butler: Trello інтегрується з Butler для автоматизації повторюваних процесів і застосування інтелектуальних алгоритмів для управління задачами.

Clarizen: використовує ШІ для надання аналітики в реальному часі про виконання проекту, управління ресурсами та прогнозування можливих ризиків.

Smartsheet: використовує інтелектуальні алгоритми для планування, відслідковування та коригування термінів і бюджету.

Продуктом є звіти та пропозиції щодо корегування планів, бюджетів та термінів.

2. Напрями та моделі застосування ШІ в процесах інженерії. Застосування ШІ для створення ПЗ – це основний напрям, який активно досліджується в галузі розробки ПЗ з моменту появи GPT-моделей. Основним предметом застосування ШІ в інженерії є генерація та аналіз програмних артефактів.

2.1. Управління вимогами. ШІ може допомогти у створенні початкового набору вимог та згенерувати такі документи, як технічне завдання та специфікації вимог за заданою формою або у відповідності до заданого стандарту. Найбільш популярні рішення для цього наведені нижче.

ChatGPT: може генерувати SRS на основі детальних вхідних даних, включаючи мету проєкту, функціональні вимоги, прецеденти використання та технічні обмеження.

Jira з доповненням Confluence та підтримкою ШІ: Confluence дозволяє створювати й структурувати SRS-документацію на основі шаблонів і співпраці команди, може автоматизувати процес збору вимог із Jira та інших систем, створюючи звіти та документацію.

IBM Watson: пропонує рішення на основі аналізу вимог і специфікацій до ПЗ, допомагає в аналізі бізнес-вимог і рішень.

TARA AI: допомагає автоматично збирати вимоги до проєкту, аналізує їх і може генерувати документ SRS, використовуючи історичні дані з попередніх проєктів, шаблони та введені дані про функціональність.

IBM Engineering Requirements Management DOORS Next: інструмент для збору й управління вимогами, який може автоматизувати процес генерації SRS. Може структурувати вимоги, створюючи звіти та документацію.

SpecFlow: інструмент для створення специфікацій на основі поведінкових сценаріїв. Може використовуватися для генерації вимог і документів SRS на основі поведінкових тестів і сценаріїв використання системи.

Формат промпту має включати:

- опис мети проєкту;
- основні функціональні вимоги;
- вимоги до інтерфейсів, інтеграцій та архітектури системи;
- нефункціональні вимоги (швидкість, надійність, безпека тощо);
- цільову аудиторію або користувачів;
- будь-які технічні обмеження (інтеграції, технології, платформи);
- шаблон (структура) вихідного документу.

Для генерації ТЗ відповідно до ДСТУ 34.602 потрібно налаштувати процес генерації документації вручну або через спеціалізовані інструменти з підтримкою шаблонів. ChatGPT, Jira з Confluence, IBM DOORS та TARA AI можуть допомогти автоматизувати значну частину процесу створення ТЗ, але доведеться додатково адаптувати шаблони для забезпечення повної відповідності національному стандарту.

2.2. Проєктування. Цей напрям пов'язаний з виконанням інтелектуальних складних завдань, які є критичними для подальшої реалізації системи і часто базуються на досвіді проєктувальників, але частина задач може бути розв’язана з використанням ШІ.

Проєктування архітектури. Інструменти ШІ можуть використовуватись для вирішення наступних задач проєктування:

- розбиття системи на модулі та компоненти;
- вибір патернів проєктування (MVC, Singleton, Factory тощо);
- розробка концептуальної архітектури або дизайну інтерфейсу;
- макетування користувацького інтерфейсу (UI/UX);
- створення прототипів функцій або системних компонентів.

Наведемо деяки інструменти-сервіси для створення архітектурних діаграм.

Lucidchart (*AI-based*): допомагає у проєктування системи у формі діаграм.

CloudSkew: використовується для створення архітектурних діаграм хмарних рішень. Рекомендує інструменти і технології для хмарних архітектур (наприклад, AWS, Azure, GCP);

EdrawMax AI: містить безліч шаблонів діаграм для моделювання архітектур IT-систем різного рівня, забезпечуючи командну роботу та можливість експорту проєктів в різні формати.

Під час вибору та створенні архітектурних рішень використовуються таки допоміжні інструменти ШІ.

ChatGPT: може допомагати у створенні та коригуванні проєктної документації, архітектурних рішень, а також пропонувати рішення для розробки коду;

DeepAI Architect: цей інструмент використовує машинне навчання для створення архітектурних моделей на основі вимог до системи, аналізує вимоги та пропонує архітектурні рішення, включаючи варіанти мікросервісів, монолітів або гібридних підходів.

Формат промпту, як правило, потребує визначити:

– вимоги до системи (функціональні, нефункціональні);

– вимоги до масштабованості, надійності та продуктивності;

– інтеграційні вимоги (API, сервіси).

Проєктування БД. Задачі проєктування БД, створення запитів до БД та трансформації даних достатньо ефективно розв’язуються за допомогою інструментів ШІ. Серед них ті, що наведені нижче.

ChatGPT та Codex від OpenAI: може автоматично генерувати схеми бази даних на основі описів таблиць і зв'язків між ними, виконувати їх нормалізацію та оптимізацію.

ER/Studio Data Architect: інструмент для створення ER-діаграм з підтримкою автоматичного генерування структури бази даних та аналізу моделей даних з використанням ШІ. Генерує пропозицій щодо покращення структури БД.

Інтерфейс з відповідним інструментом може бути достатньо специфічним, промпт може містити опис основних сутностей і їх атрибутів, вимоги до продуктивності та масштабованості БД, дані про взаємозв'язки між об'єктами в системі.

Розробка інтерфейсів. ШІ допомагає в дизайні користувацьких інтерфейсів. Найбільш популярні таки інструменти.

Figma з AI-плагінами: допомагає генерувати дизайн елементів інтерфейсу або оптимізувати існуючий дизайн на основі аналізу користувацького досвіду.

Uizard: платформа ШІ для швидкого створення прототипів інтерфейсів на основі текстових описів, дозволяє перетворити опис вимог до інтерфейсу у візуальні макети.

Sketch2Code (*Microsoft AI*): перетворює ескізи або прототипи інтерфейсу, зроблені вручну, в готовий HTML код, що прискорює створення користувацького інтерфейсу на ранніх етапах розробки.

Screenshot to Code: нейронна мережа, що перетворює графічні зображення інтерфейсу (знімки екранів) в програмний код.

2.3. Конструювання. Цей напрям включає процеси реалізації ПЗ і на сьогодні активно застосовується безліч подібних інструментів ШІ. Більше 75% розробників використовують штучний інтелект хоча б для одного професійного завдання щодня. В компанії Google більше 25% нового коду генерується за допомогою ШІ, а компанія Amazon заощадила 4500 людино-років завдяки своїй нейронній мережі для кодингу Amazon Q (79% коду було прийнято без корегування).

Написання коду включає створення функціональних модулів або компонентів ПЗ, таких як:

– модулі, класи, функції або закінчені компоненти програми;

– конфігураційні файли (наприклад, для налаштування інфраструктури або середовища).

Найбільш ефективними на сьогодні є наступні інструменти кодінгу.

ChatGPT (*OpenAI*): допомагає у вирішенні питань, написанні функцій та алгоритмів для різних мов програмування.

Codex (*OpenAI*): інструмент для генерації фрагментів коду за текстовим описом.

GitHub Copilot: інструмент на основі GPT, що допомагає написати код, пропонуючи фрагменти або блоки коду на основі контексту.

Tabnine: асистент, який використовує автодоповнення для прискорення написання коду.

Amazon Q: чат-бот, побудований з використанням кількох моделей ШІ, який може інтегруватися з інструментами розробки ПЗ, має розвинені функції щодо генерації та аналізу коду, тестування та виконання інших функцій.

CodeWhisperer (*AWS*): інтегрується з IDE і надає рекомендації в реальному часі.

Code GPT: плагін-генератор коду для VSCode, Cursor, та Jet Brains, побудований на технології агентів.

Cursor AI: повноцінне середовище розробки, оснащене безліччю корисних інструментів ШІ, створених спеціально для програмістів, одним натисканням клавіші може дописати цілий блок коду, враховуючи завдання, або відповісти на будь-яке запитання щодо проекту.

IntelliCode: інструмент від Microsoft, що надає рекомендації та автодоповнення коду на основі ШІ.

Kite: автодоповнення коду для Python, JavaScript, Go та інших мов.

Аналіз коду – це перевірка коду на якість та відповідність стандартам якості, безпеки та покращення коду. Додатково до генераторів коду можна розглядати наступні інструменти.

DeepCode ma Snyk: виконують статичне тестування безпеки коду (SAST).

CodePal: нейромережа, яка може автоматично виправляти помилки та генерувати код.

Adrenaline: інноваційна платформа, яка не лише покращує якість написаного коду, а й виправляє помилки за допомогою ШІ.

Запити до БД. Інструменти ШІ допомагають створювати складні запити до БД, а також скрипти для міграції баз даних або налаштування реплікації.

SQLizer: інструмент, який спрощує створення SQL-запитів до БД. Дозволяє генерувати SQL-код із текстових описів або перетворювати дані (наприклад, з файлів CSV, JSON) у SQL-таблиці або запити.

AI Query: інструмент ШІ, створений для спрощення роботи з БД за допомогою генерації SQL-запитів із природної мови.

Створення інсталяційного пакету. Цей процес, як правило, потребує розробки певного проєкту з використанням скриптової мови для отримання інсталяційного варіанту постачання системи для різних середовищ, що потребує значних зусиль. Тому для цього також можна використовувати інструменти ШІ. Найпоширеніші засоби ШІ для цієї задачі перелічені нижче.

GitHub Actions із Copilot: автоматизує процес складання та пакування програмного забезпечення. Copilot може допомогти писати сценарії для автоматизації, тоді як GitHub Actions виконує CI/CD процеси для складання, тестування і створення інсталяційного пакета.

AWS CodeBuild з функціями ШІ від AWS CodeGuru: забезпечує автоматизацію складання і створення інсталяційного пакету. CodeGuru може додатково здійснювати аналіз коду для оптимізації і виявлення потенційних проблем в пакеті, покращуючи загальну якість.

Microsoft Azure DevOps з Copilot для Azure Pipelines: підходить для автоматизації CI/CD процесів з інтеграцією Copilot, який допомагає створювати сценарії для Azure Pipelines. Може бути використаний для побудови, пакування та розгортання інсталяційних пакетів.

Ansible з OpenAI GPT (для написання сценаріїв автоматизації): підходить для автоматизації створення і налаштування інсталяційних пакетів, а GPT може допомогти написати складні сценарії автоматизації.

NSIS (Nullsoft Scriptable Install System) з інтеграцією Copilot або ChatGPT: є популярним інструментом для створення інсталяторів для Windows. Використання Copilot або ChatGPT допомагає з написанням скриптів інсталятора, спрощуючи створення установчих файлів.

2.4. Тестування. Тестування ПЗ може бути полегшено за рахунок використання інструментів ШІ, які можуть надавати рекомендації щодо тестування, генерувати

функціональні тести з урахуванням вимог та тестового покриття, тести продуктивності, безпеки, інтерфейсу, аналізувати результати тестів на основі журналів роботи програми, та виконувати end-to-end (E2E) тестування (тип тестування, який перевіряє повну функціональність системи від початку до кінця, щоби переконатися, що система працює як єдине ціле та відповідає бізнес-вимогам).

Приклади інструментів ШІ для тестування ПЗ наведені нижче.

Testim.io: платформа для автоматизації тестування, що використовує ШІ для створення, виконання та управління тестами. Підтримує автоматизацію end-to-end тестів та автоматично оновлює тести відповідно до змін в коді.

Katalon Studio: універсальна платформа для автоматизації тестування веб-застосунків, мобільних застосунків, API та десктопних програм. Забезпечує інтеграцію з системами управління тестуванням, такими як Jira, та підтримує різні платформи (Windows, macOS, Linux).

Applitools: платформа для візуального тестування, яка використовує ШІ для перевірки візуальної коректності інтерфейсів. Її унікальна функція Visual AI дозволяє тестувати дизайн і функціональність застосунків на різних пристроях і браузерах, забезпечує автоматичне порівняння знімків екрану, що допомагає швидко виявляти зміни в інтерфейсі. Інтегрується з більшістю популярних фреймворків для автоматизації тестування та DevOps-платформ.

Mabl: інтелектуальний інструмент автоматизації тестування, який забезпечує тестування end-to-end із використанням машинного навчання. Дозволяє виконувати регресійне тестування, тестування продуктивності та UI-тестування з автоматичним оновленням тестів у разі змін у програмі. Інтегрується з CI/CD конвеєрами для безперервної доставки та виявлення дефектів.

GitLab CI/CD: інтегрує ШІ для автоматичного тестування коду, моніторингу прогресу та автоматичної перевірки якості.

Sonar AI CodeFix: аналізує код на предмет дотримання стандартів, безпеки та продуктивності.

DeepCode: інструмент для аналізу та перевірки коду, який виявляє проблеми з безпекою та допомагає уникнути помилок.

Checkmarx: використовується для пошуку вразливостей у коді та аналізу його безпеки.

2.5. Документування. ШІ можна використовувати для створення проєктної документації. Для створення документу типу “Технічний проєкт” (також відомого як SDD – Software Design Document), який включає архітектурні рішення, технічні вимоги, діаграми, опис структур бази даних, API і т.п., інструменти ШІ можуть використовувати в якості вхідних даних код програми з коментарями, описи окремих проєктних рішень та моделей, схеми БД тощо. Для цього можна використати таки інструменти ШІ.

ChatGPT: підходить для генерації текстової документації, опису архітектури, структур даних, а також надає рекомендації щодо технологій.

Jasper AI: забезпечує структурування документів і текстове наповнення на основі введених інструкцій, корисний для формування введення та опису модулів у технічному проєкті.

Notion AI: підходить для створення структурованих документів, зокрема технічних проєктів, допомагаючи організувати великий обсяг технічної інформації та підтримуючи гнучкий формат редагування.

Microsoft Word з функцією Copilot (для Microsoft 365): дозволяє автоматизувати структурування і написання технічного проєкту, включає функції для редагування тексту, вставки діаграм та схем, а також допомагає в управлінні версіями документів.

Ці інструменти можна комбінувати для генерування тексту, організації розділів документації, а також для створення схем і діаграм, що дозволяє створити комплексний

технічний проєкт, який охоплює всі необхідні елементи архітектури та технічних специфікацій.

Висновки. Проведений аналіз напрямів застосування ШІ в ТРПЗ дозволяє зробити наступні висновки:

– ефективність застосування ШІ в усіх процесах ЖЦ ПЗ зростає і ця тенденція, вочевидь, буде продовжена за рахунок випуску нових інструментів та покращення якості базових моделей;

– ШІ покращує продуктивність та ефективність розробників, але потребує залучення людей для перевірки, уточнення та вдосконалення результатів;

– якість результатів ШІ залежить від якості та кількості даних, на яких він був навчений, а також від якості промптів та контексту застосування;

– більшу ефективність ШІ демонструє при виконанні задач інженерії, де вхідні дані є достатньо чіткими, а результат відносно просто перевіряється (для деяких задач можна отримувати готові рішення без додаткових змін, як-от генерація коду рішення рутинних задач);

– в задачах менеджменту ПЗ ШІ ефективніше вирішує прості завдання, ніж складні, тому його доцільно використовувати для отримання прототипів та варіантів рішень;

– для критичних рішень і творчих завдань ШІ ще потребує ретельного контролю з боку людей, оскільки він може пропонувати помилкові або неповні рішення;

– для максимально якісного застосування ШІ в розробці ПЗ необхідно збалансувати його можливості з експертним втручанням людей, щоб мінімізувати ризики і підвищити якість кінцевого продукту;

– одним з напрямів підвищення ефективності ШІ є його навчання в інтересах певної організації на закінчених проєктах та корпоративних стандартах.

В якості перспектив застосування ШІ в ТРПЗ можна очікувати наступне:

– комплексне застосування декількох моделей ШІ по всіх напрямках ТРПЗ з синхронізацією всіх процесів та артефактів проєкту (проєктних моделей, коду, тестів, планів, документації та стандартів);

– спеціалізація інструментів ШІ шляхом навчання на корпоративних стандартах.

Перспективами подальших досліджень є детальний аналіз та формування ефективних моделей застосування ШІ по окремих напрямках ТРПЗ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] J. Sauvola, S. Tarkoma, M. Klemettinen, J. Riekki, and D. Doermann, “Future of software development with generative AI”, *Autom Softw Eng*, vol.31, no.1, 2024. doi: https://doi.org/10.1007/s10515-024-00426-z.

[2] M. Barenkamp, J. Rebstadt, and O. Thomas, “Applications of AI in classical software engineering”, *AI Perspect*, vol.2, no.1, 2020. doi: https://doi.org/10.1186/s42467-020-00005-4.

[3] S. Kumar, “Artificial Intelligence in Software Engineering: A Systematic Exploration of AI-Driven Development”, *IJIRSET*, vol.13, no.6, 2024. doi: https://doi.org/10.15680/ijirset.2024.1306220.

[4] M. Tufano, A. Agarwal, J. Jang, R.Z. Moghaddam, and N. Sundaresan, “Automated AI-Driven Development”, A*rxiv*, 2024. [Online]. Available: https://arxiv.org/abs/2403.08299. Accessed on: 15 Nov. 2024. doi: https://doi.org/10.48550/arXiv.2403.08299.

[5] R. Ramler, M. Moser, L. Fischer, M. Nissl, and R. Heinzl, “Industrial Experience Report on AI-Assisted Coding in Professional Software Development”, i*n Proc. LLM4Code'24*, NY, USA, 2024. doi: https://doi.org/10.1145/3643795.3648377.

[6] Q. Zhang et al., “A survey on large language models for software engineering”, *ArXiv*, 2023. [Online]. Available: https://arxiv.org/abs/2312.15223. Accessed on: 15 Nov. 2024. doi:

https://doi.org/10.48550/arXiv.2312.15223.

[7] B. Yetiştiren, I. Özsoy, M. Ayerdem, and E. Tüzün, “Evaluating the Code Quality of AI-Assisted Code Generation Tools: An Empirical Study on GitHub Copilot, Amazon CodeWhisperer, and ChatGPT”, *ArXiv*, 2023. [Online]. Available: https://arxiv.org/abs/2304.10778. Accessed on: 15 Nov. 2024. doi: https://doi.org/10.48550/arXiv.2304.10778.

[8] A.R. Sadik, A. Ceravola, F. Joublin, and J. Patra, “Analysis of ChatGPT on Source Code”, *ArXiv*, 2023. [Online]. Available: https://arxiv.org/abs/2306.00597. Accessed on: 15 Nov. 2024. doi: https://doi.org/10.48550/arXiv.2306.00597.

[9] J. White, S. Hays, Q. Fu, J. Spencer-Smith, and D.C. Schmidt, “ChatGPT prompt patterns for improving code quality, refactoring, requirements elicitation, and software design”, *ArXiv*, 2023. [Online]. Available: https://arxiv.org/abs/2303.07839. Accessed on: 15 Nov. 2024. doi: https://doi.org/10.48550/arXiv.2303.07839.

[10] B. Alessio, “A Comparative Study of Code Generation using ChatGPT 3.5 across 10 Programming Languages”, *ArXiv*, 2023. [Online]. Available: https://arxiv.org/abs/2308.04477. Accessed on: 15 Nov. 2024. doi: https://doi.org/10.48550/arXiv.2308.04477.

[11] M. Wei et al., “LMs: Understanding Code Syntax and Semantics for Code Analysis”, *ArXiv*, 2023. [Online]. Available: https://arxiv.org/abs/2305.12138. Accessed on: 15 Nov. 2024. doi: https://doi.org/10.48550/arXiv.2305.12138.

[12] X. Li, H. Zheng, J. Chen, Y. Zong, and L. Yu, “User Interaction Interface Design and Innovation Based on Artificial Intelligence Technology”, *JTPES*, vol. 4, no. 03, pp. 1–8, Mar. 2024. doi: https://doi.org/10.53469/jtpes.2024.04(03).01.

[13] J.H. Klemmer et al., “Using AI Assistants in Software Development: A Qualitative Study on Security Practices and Concerns”, *ArXiv*, 2024. [Online]. Available: https://arxiv.org/abs/2405.06371. Accessed on: 15 Nov. 2024.

[14] ДСТУ ISO/IEC/IEEE 12207:2018 Інженерія систем і програмних засобів. Процеси життєвого циклу програмних засобів (ISO/IEC/IEEE 12207:2017, IDT).

[15] ДСТУ ISO/IEC 25010:2016 Інженерія систем і програмних засобів. Вимоги до якості систем і програмних засобів та її оцінювання (SQuaRE). Моделі якості системи та програмних засобів (ISO/IEC 25010:2011, IDT).

Стаття надійшла 20.10.2024.

REFERENCE

[1] J. Sauvola, S. Tarkoma, M. Klemettinen, J. Riekki, and D. Doermann, “Future of software development with generative AI”, *Autom Softw Eng*, vol.31, no.1, 2024. doi: https://doi.org/10.1007/s10515-024-00426-z.

[2] M. Barenkamp, J. Rebstadt, and O. Thomas, “Applications of AI in classical software engineering”, *AI Perspect*, vol.2, no.1, 2020. doi: https://doi.org/10.1186/s42467-020-00005-4.

[3] S. Kumar, “Artificial Intelligence in Software Engineering: A Systematic Exploration of AI-Driven Development”, *IJIRSET*, vol.13, no.6, 2024. doi: https://doi.org/10.15680/ijirset.2024.1306220.

[4] M. Tufano, A. Agarwal, J. Jang, R.Z. Moghaddam, and N. Sundaresan, “Automated AI-Driven Development”, A*rxiv*, 2024. [Online]. Available: https://arxiv.org/abs/2403.08299. Accessed on: 15 Nov. 2024. doi: https://doi.org/10.48550/arXiv.2403.08299.

[5] R. Ramler, M. Moser, L. Fischer, M. Nissl, and R. Heinzl, “Industrial Experience Report on AI-Assisted Coding in Professional Software Development”, i*n Proc. LLM4Code'24*, NY, USA, 2024. doi: https://doi.org/10.1145/3643795.3648377.

[6] Q. Zhang et al., “A survey on large language models for software engineering”, *ArXiv*, 2023. [Online]. Available: https://arxiv.org/abs/2312.15223. Accessed on: 15 Nov. 2024. doi:

https://doi.org/10.48550/arXiv.2312.15223.
[7] B. Yetiştiren, I. Özsoy, M. Ayerdem, and E. Tüzün, “Evaluating the Code Quality of AI-Assisted Code Generation Tools: An Empirical Study on GitHub Copilot, Amazon CodeWhisperer, and ChatGPT”, *ArXiv*, 2023. [Online]. Available: https://arxiv.org/abs/2304.10778. Accessed on: 15 Nov. 2024. doi: https://doi.org/10.48550/arXiv.2304.10778.
[8] A.R. Sadik, A. Ceravola, F. Joublin, and J. Patra, “Analysis of ChatGPT on Source Code”, *ArXiv*, 2023. [Online]. Available: https://arxiv.org/abs/2306.00597. Accessed on: 15 Nov. 2024. doi: https://doi.org/10.48550/arXiv.2306.00597.
[9] J. White, S. Hays, Q. Fu, J. Spencer-Smith, and D.C. Schmidt, “ChatGPT prompt patterns for improving code quality, refactoring, requirements elicitation, and software design”, *ArXiv*, 2023. [Online]. Available: https://arxiv.org/abs/2303.07839. Accessed on: 15 Nov. 2024. doi: https://doi.org/10.48550/arXiv.2303.07839.
[10] B. Alessio, “A Comparative Study of Code Generation using ChatGPT 3.5 across 10 Programming Languages”, *ArXiv*, 2023. [Online]. Available: https://arxiv.org/abs/2308.04477. Accessed on: 15 Nov. 2024. doi: https://doi.org/10.48550/arXiv.2308.04477.
[11] M. Wei et al., “LMs: Understanding Code Syntax and Semantics for Code Analysis”, *ArXiv*, 2023. [Online]. Available: https://arxiv.org/abs/2305.12138. Accessed on: 15 Nov. 2024. doi: https://doi.org/10.48550/arXiv.2305.12138.
[12] X. Li, H. Zheng, J. Chen, Y. Zong, and L. Yu, “User Interaction Interface Design and Innovation Based on Artificial Intelligence Technology”, *JTPES*, vol. 4, no. 03, pp. 1–8, Mar. 2024. doi: https://doi.org/10.53469/jtpes.2024.04(03).01.
[13] J.H. Klemmer et al., “Using AI Assistants in Software Development: A Qualitative Study on Security Practices and Concerns”, *ArXiv*, 2024. [Online]. Available: https://arxiv.org/abs/2405.06371. Accessed on: 15 Nov. 2024.
[14] SSU ISO/IEC/IEEE 12207:2018 Systems and software engineering. Software life cycle processes (ISO/IEC/IEEE 12207:2017, IDT).
[15] SSU ISO/IEC 25010:2016 Systems and software engineering. Requirements for the quality of systems and software tools and its evaluation (SQuaRE). Models of system and software quality (ISO/IEC 25010:2011, IDT).

VOLODYMYR SOKOLOV
VIACHESLAV RIABTSEV
OLEKSANR USPENSKYI
DANYLO KOPYCH

APPLICATION DIRECTIONS OF ARTIFICIAL INTELLIGENCE IN SOFTWARE DEVELOPMENT TECHNOLOGIES

The article presents the results of a systematic analysis of the current state of application of artificial intelligence (AI) in software engineering (SW) based on the analysis of publications, assessment of AI capabilities, experience in its application, and conducted experiments. The conceptual foundations of the research were formed, which determine: perception of AI as a tool, not an individual of work; the main directions of its application are engineering and management; the subject of AI application is the processing of artifacts (synthesis and analysis) and obtaining consultations; the need to assess the quality of AI-derived products and analyze the risks of its use is emphasized. Directions of application of AI in management: agreement processes (development of product concept and contract), organizational processes (project group formation and selection of technologies) and project management (planning, risk management, control and analysis of project implementation) Directions of application of AI in engineering: requirements management, design,

construction, testing and documenting. To systematize the analysis of AI application directions, a conceptual model was developed, which includes: the direction, subject, and mode of application of AI. The mode of application of AI: the format of the prompt (problem statement and set of input data), the required product and its type (finished product, prototype, template, solution options, information support), the role of AI (executor, co-author, consultant), form of AI interaction (external service, integration via API, integrated system or local autonomous system). A structure of derivative models was formed for the analysis of the application of AI in specific directions with an overview of the capabilities of the most effective AI tools. As conclusions, it was determined that in management, the most rational model of using AI is to receive consultations and prototypes of documentation when contacting external AI services, in engineering – creating prototypes of project solutions and documentation based on external services, using integrated AI systems for design and testing in co-authorship mode. The risks of using AI include the possibility of obtaining insufficiently detailed documentation, complex and confusing software artifacts, and errors in the software code. To reduce risks and increase the effectiveness of AI application, it is determined that constant quality control of its products and training based on corporate requirements and standards is required.

Keywords: artificial intelligence, software development technologies.

Соколов Володимир Володимирович, кандидат технічних наук, доцент, доцент кафедри кібербезпекиі застосування інформаційних системі технологій, Інститут спеціального зв'язку та захисту інформації Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського", Київ, Україна. ORCID 0000-0002-5779-7167, v.sokolov@kpi.ua.

Рябцев Вячеслав Віталійович, кандидат технічних наук, доцент, доцент кафедри кібербезпекиі застосування інформаційних системі технологій, Інститут спеціального зв'язку та захисту інформації Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського", Київ, Україна. ORCID 0000-0001-8331-0132, viacheslav.riabtsev@gmail.com.

Успенський Олександр Анатолійович, кандидат технічних наук, доцент, доцент кафедри кібербезпекиі застосування інформаційних системі технологій, Інститут спеціального зв'язку та захисту інформації Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського", Київ, Україна. ORCID 0000-0001-6953-421X, uspensky@ukr.net.

Копич Данило Олексійович, курсант, Інститут спеціального зв'язку та захисту інформації Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського", Київ, Україна. ORCID 0009-0005-9809-546X, danyla.kopych@gmail.com.

Sokolov Volodymyr, candidate of technical sciences, associate professor, associate professor at the cybersecurity and application of information systems and technologies academic department, Institute of special communication and information protection of National technical university of Ukraine "Igor Sikorsky Kyiv polytechnic institute", Kyiv, Ukraine.

Riabtsev Viacheslav, candidate of technical sciences, associate professor, associate professor at the cybersecurity and application of information systems and technologies academic department, Institute of special communications and information protection of National technical university of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine.

Uspenskyi Oleksandr, candidate of technical sciences, associate professor, associate professor at the cybersecurity and application of information systems and technologies academic department, Institute of special communications and information protection of National technical university of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine.

Kopych Danylo, cadet, Institute of special communications and information protection of National technical university of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine.