International Science Journal of Engineering & Agriculture
2026; 5(3): 20-29
https://isg-journal.com/isjea
doi: 10.46299/j.isjea.20260503.03
ISSN: 2720-6319

INTERNATIONAL
SCIENCE GROUP

Модель делегування задач тестування між людиною та ШІ-агентами в SDLC

Сергій Коповський
Національний університет «Львівська політехніка», Львів, Україна
ORCID: 0009-0001-8969-2925

Анотація: У статті розглянуто проблему розподілу задач тестування програмного забезпечення між людиною та ШІ-агентами в межах життєвого циклу розроблення програмного забезпечення. Актуальність дослідження зумовлена швидким поширенням генеративного штучного інтелекту, інструментів інтелектуальної автоматизації, багатокрокових агентних сценаріїв і практик безперервного тестування, що підвищують продуктивність команди, але одночасно створюють ризики непрозорого делегування критичних рішень. Метою роботи є побудова формалізованої моделі, яка дає змогу визначати доцільний режим виконання тестової задачі: людиною, ШІ-агентом або у спільному режимі з контролем людини. Методичну основу становлять підходи до рівнів автоматизації, принципи взаємодії людини із ШІ, ідеї Human-in-the-Loop, ризик-орієнтоване мислення у тестуванні та формалізація багатокритеріального оцінювання. Для опису задачі використано нормовані параметри ризику, складності, неоднозначності, повторюваності та доменної глибини. Запропоновано функцію придатності задачі до делегування ШІ, шар обмежень, правило примусового повернення задачі людині за високого ризику, а також три рівні автономності з різним набором контрольних точок. На прикладі п’яти типових задач тестування показано, що ШІ найбільш ефективний у повторюваних і добре структурованих сценаріях, тоді як *exploratory testing*, високий ризик і глибока залежність від доменного контексту вимагають посиленої участі людини. Практична цінність роботи полягає в можливості використання моделі під час планування QA pipeline, налаштування агентних помічників, визначення зон автоматизації та проєктування політик Human-in-the-Loop у командах розроблення. Модель також може бути використана для уніфікації процедур контролю якості в міждисциплінарних командах розроблення.

Ключові слова: штучний інтелект; тестування програмного забезпечення; життєвий цикл розроблення програмного забезпечення; делегування задач; автономність; контроль людини.

1. Вступ

Інтеграція штучного інтелекту в інженерні процеси вже вийшла за межі допоміжного автодоповнення коду або окремих генераторів тестових даних. У сучасних QA-практиках ШІ-системи можуть пропонувати тест-кейси, створювати перевірки для API, готувати тестові дані, класифікувати дефекти, аналізувати журнали виконання, шукати першопричини падінь, формувати звіти та взаємодіяти з іншими інструментами через агентні ланцюжки. Це означає, що питання автоматизації зміщується від простого «що можна автоматизувати» до складнішого: «що саме, за яких умов і з яким рівнем довіри можна делегувати ШІ-системі без погіршення якості тестування».

На практиці команди розробки дедалі частіше працюють у змішаному режимі, де частина задач виконується фахівцем із тестування, частина — ШІ-помічником, а частина — у кооперації людини й інтелектуального агента. Однак відсутність єдиних формальних правил такого розподілу призводить до кількох типових проблем. По-перше, делегування часто відбувається ситуативно та залежить від індивідуальної довіри конкретного тестувальника до інструмента. По-друге, одна й та сама задача в різних командах може передаватися ШІ або

людині за різними неформальними критеріями, що знижує відтворюваність процесу. По-третє, неконтрольоване передавання критичних задач ШІ-агентам створює ризики пропуску дефектів, хибної інтерпретації вимог, втрати контексту й накопичення помилок у прийнятті рішень.

Особливо гостро ця проблема проявляється в межах повного SDLC (Software Development Life Cycle), де тестування виконує не лише функцію виявлення помилок, а й підтвердження відповідності продукту вимогам, бізнес-цілям, доменним обмеженням, очікуванням користувачів та регуляторним нормам. З цієї причини одна й та сама автоматизаційна логіка не може однаково добре працювати для regression testing, exploratory testing, проєктування тестів, аналізу дефектів або перевірки ризикових змін. Для прийняття обґрунтованого рішення необхідно одночасно враховувати кілька характеристик задачі: рівень ризику, складність, ступінь неоднозначності, повторюваність та глибину доменного знання.

У цій роботі пропонується формалізована модель делегування задач тестування між людиною та ШІ-агентами, яка поєднує кількісну оцінку придатності задачі до автоматизованого виконання з набором обов'язкових обмежень безпеки та механізмів людського контролю. На відміну від описових рекомендацій, запропонований підхід задає явні правила переходу між режимами «Людина», «ШІ» та «Людина + ШІ», а також дозволяє пов'язати рішення про делегування з рівнем автономності та потрібними контрольними точками. Це створює основу для узгодженішого планування тестування, пояснюваного використання ШІ в QA pipeline та побудови політик відповідального впровадження агентних систем у промисловій практиці.

2. Об'єкт і предмет дослідження

Об'єктом дослідження є процес делегування задач тестування програмного забезпечення між людиною та ШІ-агентами в межах життєвого циклу розроблення програмного забезпечення. Такий процес охоплює не лише фактичний розподіл виконавців, а й процедури прийняття рішень, встановлення рівня автономності, визначення меж відповідальності сторін та призначення контрольних перевірок для забезпечення якості результату.

Предметом дослідження є формальна модель прийняття рішення щодо делегування тестової задачі, яка враховує набір релевантних характеристик цієї задачі. У межах роботи розглядаються такі параметри:

1. Ризик помилкового виконання або неправильної інтерпретації результату.
2. Складність задачі як міра логічної, технічної або міжсистемної зв'язаності.
3. Неоднозначність вимог чи очікуваного результату.
4. Повторюваність, що відображає частоту та шаблонність виконання.
5. Доменна глибина, яка визначає ступінь необхідності в неформалізованому предметному знанні.

Сукупність цих характеристик визначає, наскільки безпечно, доцільно й економічно передавати виконання ШІ-агенту. У практичному вимірі предмет дослідження включає також три взаємопов'язані аспекти:

- Спосіб розрахунку придатності задачі до виконання ШІ.
- Шар обмежень, який не дозволяє моделі приймати формально оптимальне, але небезпечне рішення за умови високої невизначеності або критичності домену.
- Механізм Human-in-the-Loop, який визначає, на якому етапі людина має підтвердити вхідні дані чи результати або здійснити втручання у випадку збою.

Саме поєднання цих аспектів дає змогу розглядати делегування не як одноразовий вибір виконавця, а як керований і пояснюваний процес.

3. Мета та задачі дослідження

Метою дослідження є розроблення формалізованої моделі делегування задач тестування між людиною та ШІ-агентами, яка забезпечує узгоджене й відтворюване прийняття рішень щодо того, хто саме має виконувати конкретну задачу в межах SDLC: людина, ШІ-агент чи

людина спільно з ШІ. Досягнення цієї мети передбачає не тільки кількісне оцінювання придатності задачі до делегування, а й побудову системи запобіжників для сценаріїв, у яких повна автоматизація є небажаною або ризиковою.

Для досягнення поставленої мети в роботі сформульовано такі дослідницькі завдання:

- визначити ключові характеристики тестових задач, які впливають на можливість їх безпечного делегування;
- запропонувати функцію оцінювання придатності задачі до виконання ШІ;
- описати правила constraint layer (рівня обмежень), які мають пріоритет над числовою оцінкою у випадках високого ризику або високої неоднозначності;
- пов’язати рішення про делегування з відповідними рівнями автономності;
- визначити мінімально необхідні контрольні точки за методологією Human-in-the-Loop;
- провести пілотну перевірку моделі на типових задачах тестування, характерних для сучасного QA pipeline.

Практичним результатом реалізації цих завдань має стати модель, придатна для використання як під час ручного планування роботи тестувальників, так і під час конфігурації ШІ-помічників, сценаріїв orchestration та розроблення внутрішніх політик керування інтелектуальною автоматизацією в командах програмної інженерії.

4. Аналіз літератури

Проблема делегування задач між людиною й автоматизованими системами має міждисциплінарний характер, тому її аналіз доцільно починати з робіт, присвячених рівням автоматизації та поведінці людини в умовах часткової або повної передачі функцій системі. У роботі Parasuraman, Sheridan та Wickens сформульовано модель типів і рівнів взаємодії людини з автоматизацією, яка залишається базовою для оцінювання меж передачі рішень технічній системі [1]. Подальший розвиток цієї лінії досліджень пов’язаний із принципами взаємодії людини із ШІ, у яких акцент зміщено на пояснюваність, керованість та передбачувану поведінку ШІ-систем у реальних робочих сценаріях [2]. Водночас проблема належного рівня довіри до автоматизованих рішень була деталізована в дослідженнях довіри до automation, де показано, що надмірна довіра та недостатня довіра однаково шкідливі для якості виконання задач [3].

Для сфери програмної інженерії важливим є також те, що делегування в тестуванні не може розглядатися у відриві від стандартів самої тестової діяльності. Міжнародні стандарти серії ISO/IEC/IEEE 29119 задають загальну концептуальну рамку тестування та описують як базові поняття, так і типові артефакти тестової документації [4, 5]. Ці документи важливі тим, що вони дозволяють трактувати тестову задачу як структурований робочий об’єкт із чіткими входами, виходами, критеріями завершення та свідченнями виконання. Класичні роботи з теорії та практики тестування наголошують, що не всі види тестової активності однаково формалізовані: поруч із добре алгоритмізованими діями існують евристичні, інтуїтивні та дослідницькі практики, які значною мірою залежать від досвіду тестувальника [6, 7].

Окремий кластер літератури стосується безпосередньо застосування ШІ, машинного навчання й великих мовних моделей у тестуванні. Сучасні огляди показують, що ШІ вже використовується для генерації тест-кейсів, локалізації дефектів, побудови тестових оракулів, пріоритизації регресійних наборів, аналізу логів та ремонту програмних артефактів [8]. Систематичний огляд методів машинного навчання в software testing фіксує зростання різноманітності підходів і наголошує на нерівномірності якості емпіричних доказів, масштабів експериментів та критеріїв оцінювання [9]. Пізніші роботи, спрямовані вже конкретно на еволюцію штучного інтелекту в тестуванні, підкреслюють розширення спектра задач: від традиційного defect prediction до агентної автоматизації, тестування ШІ-систем і використання пояснюваних моделей у QA-процесах [10]. Водночас емпіричні оцінки великих мовних моделей у тестуванні демонструють неоднозначну картину: LLM можуть підсилювати ручне тестування та виявляти додаткові помилки, але також схильні до галюцинацій і контекстних помилок, тому потребують обов’язкового людського контролю [11].

Додаткову теоретичну основу для запропонованої моделі формує література з Human-in-the-Loop. У роботі Mosqueira-Rey та співавторів узагалено стан напряму HITL-ML і показано, що включення людини може відбуватися не лише на етапі корекції результату, а й на етапах постановки задачі, відбору даних, пояснення моделі та навчання [12]. Огляд Wu та співавторів розглядає HITL з позиції даних, тренування моделі та побудови систем, підкреслюючи, що людське втручання особливо важливе в умовах нестачі даних, високої невизначеності та потреби в доменному знанні [13]. Для нашого дослідження це означає, що людина має бути не резервним виконавцем на випадок відмови ШІ, а проєктованим елементом процесу, присутність якого змінюється залежно від класу задачі.

У контексті побудови формальної моделі вагомими є також методологічні праці з багатокритеріального прийняття рішень і практичної валідації концептуальних моделей. Підхід Saaty до ієрархічного аналізу рішень створює підстави для подальшого калібрування ваг критеріїв, якщо модель буде впроваджуватися в конкретному домені чи організації [14]. Праці Yin щодо case study як дослідницької стратегії є релевантними для пілотної перевірки формальної моделі на обмеженій, але типологічно репрезентативній множині задач [15].

Отже, аналіз літератури показує наявність значної кількості робіт про автоматизацію, взаємодію людини із ШІ, HITL та ШІ у тестуванні ПЗ, проте виявляє відсутність узагальненої моделі, яка б одночасно враховувала ризик, складність, неоднозначність, повторюваність, доменну глибину, рівень автономності та обов'язкові правила людського контролю під час делегування задач тестування.

5. Методи досліджень

У роботі використано концептуальне моделювання, елементи багатокритеріального оцінювання, ризик-орієнтований підхід до тестування та пілотну перевірку моделі на типовій множині QA-задач. Основне припущення полягає в тому, що кожну задачу тестування можна описати скінченним набором параметрів, нормованих на інтервалі від 0 до 1. Нульове значення відповідає мінімальному прояву характеристики, одиничне — максимальному.

Для формалізації рішення про делегування вводиться функція

$$D(t, C, R, K) -> \{H, A, H + A\}, \quad (1)$$

де t — тип задачі тестування, C — контекст виконання, R — ризик, K — складність, α — неоднозначність, ρ — повторюваність, δ — доменна глибина, H — виконання людиною, A — виконання ШІ-агентом, $H+A$ — спільне виконання або виконання ШІ з обов'язковим людським контролем. Формула (1) задає множину допустимих виконавців, а конкретне рішення уточняється через кількісну оцінку придатності до делегування.

Показник придатності задачі до виконання ШІ визначається за формулою

$$\text{AI\_suit} = 0.35(1 - \alpha) + 0.30\rho + 0.20(1 - \delta) + 0.15(1 - \text{K}), \quad (2)$$

де S_AI — показник придатності використання штучного інтелекту де більші значення відповідають більшій доцільності використання ШІ. Обрана структура формули відображає логіку, за якою найсильніше на користь ШІ працюють низька неоднозначність та висока повторюваність, тоді як велика доменна глибина й висока складність зменшують автономну придатність задачі. Ваги в поточній версії моделі задаються експертно, а в подальших дослідженнях можуть бути відкалібровані на основі ієрархічного аналізу рішень або емпіричних даних про успішність виконання задач [14].

Щоб зменшити ймовірність небезпечного делегування, поверх числової оцінки накладається шар обмежень. Перший тип обмеження — *override rule*: якщо ризик перевищує поріг $R > 0.8$, задача не делегується ШІ незалежно від значення S_AI.

$$\text{R} > 0.8 \Rightarrow \text{D} = \text{H}, \quad (3)$$

Другий тип обмеження — *constraint layer*: якщо неоднозначність перевищує поріг $\alpha > 0.4$ і водночас доменна глибина є суттєвою, тобто $\delta > 0.5$, мінімально допустимим рішенням стає режим *H+A*.

$$\alpha > 0.4 \wedge \delta > 0.5 \Rightarrow D \geq H + A, \tag{4}$$

Третє правило стосується exploratory testing: задачі цього типу вважаються людсько-центричними через високу залежність від гіпотез, інтуїції, контекстних сигналів і потреби оперативно змінювати напрямок дослідження системи.

З урахуванням пріоритету перелічених обмежень фінальне рішення щодо виконавця задається виразом

$$\begin{gathered} S\_AI > 0.7 \Rightarrow D = A, \\ 0.4 \leq S\_AI \leq 0.7 \Rightarrow D = H + A, \\ S\_AI < 0.4 \Rightarrow D = H. \end{gathered} \tag{5}$$

Формула (5) визначає вибір виконавця залежно від значення показника придатності S_AI з урахуванням пріоритету обмежень.

Така послідовність перевірок важлива: вона гарантує, що кількісна привабливість автоматизації не скасовує обмеження, пов’язані з ризиком або недостатньою визначеністю задачі.

Окремо визначається рівень автономності, який задає інтенсивність людського нагляду:

$$\begin{gathered} R > 0.6 \Rightarrow L1, \\ 0.3 < R \leq 0.6 \Rightarrow L2, \\ R \leq 0.3 \wedge S\_AI > 0.7 \Rightarrow L3, \end{gathered} \tag{6}$$

Рівень *L*1 означає домінування людини в прийнятті рішень, *L*2 – гібридний режим, *L*3 - низькоризикову ситуацію, у якій ШІ може виконувати операційну частину задачі з мінімально необхідним контролем. Відповідно до формули(6) автономність залежить не лише від ризику, а й від того, наскільки сама задача підходить для ШІ.

Для того щоб параметри задачі оцінювалися узгоджено, у роботі використано інтерпретаційні шкали. Ризик відображає потенційну ціну помилки тестування: вплив на безпеку, відповідність вимогам, витрати на виправлення, репутаційні втрати та ризик помилкового релізу. Складність описує кількість залежностей, кількість системних меж, частку неявної логіки, потребу в комбінуванні різнорідних артефактів і ймовірність каскадного впливу помилки. Неоднозначність фіксує нечіткість вимог, відсутність завершених acceptance criteria, конфлікт інтерпретацій або неможливість однозначно визначити очікуваний результат. Повторюваність характеризує частоту виконання задачі, стабільність її структури та можливість багаторазового застосування одного й того самого алгоритму виконання. Доменна глибина відображає обсяг предметного знання, яке неможливо легко витягнути з формалізованих вимог або тестової документації.

Перед використанням моделі доцільно узгодити в команді правила оцінювання параметрів, щоб знизити суб’єктивність. Для цього можуть бути використані експертні калібрувальні сесії, матриці прикладів, внутрішні глосарії та історичні дані про попередні тестові задачі. У межах цього дослідження параметри задавалися експертно на основі характеру репрезентативних задач. Пілотна валідація моделі виконувалася за логікою case study: рішення моделі порівнювалися з очікуваним експертним режимом виконання для вибраних задач із QA pipeline [15].

Перед обчисленням рішення також корисно зафіксувати базову шкалу інтерпретації параметрів. Вона потрібна не для математичного розрахунку як такого, а для того, щоб члени

команди однаково трактували значення 0.2, 0.5 або 0.8. У таблиці 1 наведено узагальнену схему такої інтерпретації, придатну для калібрування внутрішніх політик делегування.

Таблиця 1. Інтерпретація параметрів задач

Критерій	Низьке значення (0.0–0.3)	Середнє значення (0.4–0.6)	Високе значення (0.7–1.0)
Ризик R	Локальні наслідки, низька вартість помилки, відсутність блокування релізу	Відчутний вплив на продукт, потреба у додатковій перевірці	Критичний вплив на реліз, відповідність вимогам, безпеку або безперервність бізнесу
Складність K	Обмежений обсяг, мало залежностей, стабільні інтерфейси	Кілька залежностей, змішана логіка, помірна зв'язаність	Багато інтеграцій, непрозора логіка, каскадні наслідки збоїв
Неоднозначність α	Чіткі вимоги та критерії прийнятності	Частково неуточнені результати або граничні випадки	Конфліктні інтерпретації, exploratory-контекст, неповні критерії
Повторюваність ρ	Разова або рідкісна задача, низький потенціал повторного використання	Повторювана задача з частковою варіативністю	Часта, шаблонна задача зі стабільним сценарієм виконання
Доменна глибина δ	Достатньо загального інженерного знання	Потрібен певний доменний контекст	Сильна залежність від неявного бізнесового або регуляторного знання

Таким чином, метод дослідження поєднує формальну частину, де рішення задається формулами (1) – (6), і прикладну частину, де використовуються експертні оцінки, порогові значення та набір контрольних точок HITL. Саме таке поєднання робить модель достатньо простою для впровадження в командну практику і водночас придатною до подальшого розширення, статистичної перевірки та інтеграції з даними інженерних процесів.

6. Результати досліджень

Для демонстрації роботи моделі було відібрано п'ять типових задач тестування, які відрізняються за характером виконання, ступенем формалізації та вимогами до доменного знання: *test design*, *execution*, *exploratory testing*, *regression testing* та *bug triage*. У таблиці 2 зафіксовано експертно оцінені значення параметрів, розраховану придатність до делегування та фінальне рішення моделі.

Таблиця 2. Результати делегування задач

Задача	R	K	α	ρ	δ	S_AI	Рішення
Test design	0.7	0.8	0.7	0.3	0.8	0.265	H+A
Execution	0.2	0.3	0.1	0.9	0.2	0.850	A
Exploratory testing	0.7	0.9	0.9	0.1	0.8	0.120	H
Regression testing	0.5	0.4	0.2	0.9	0.3	0.780	A
Bug triage	0.6	0.6	0.5	0.2	0.7	0.355	H+A

Отримані результати показують, що модель не зводить делегування до механічного протиставлення *manual versus automated testing*. Вона, навпаки, розділяє задачі на три

принципово різні класи. Перший клас — задачі, де ШІ доцільно використовувати як основного виконавця. До нього належать *execution* та *regression testing*, для яких характерні висока повторюваність, низька неоднозначність і помірна або низька складність. У таких випадках автоматизований виконавець забезпечує виграш у швидкості, масштабованості та стабільності повторення дій. Другий клас — задачі гібридного характеру. *test design* та *bug triage* показують, що навіть за наявності окремих автоматизовуваних кроків рішення потребує участі людини, тому що задача або містить велику неоднозначність, або суттєво залежить від доменного контексту, або має підвищений ризик хибної класифікації. Третій клас — задачі, де домінувати повинна людина. *Exploratory testing* належить саме до таких: воно ґрунтується на формуванні гіпотез, динамічному навчанні в процесі виконання, відстеженні слабких сигналів і контекстній інтерпретації поведінки системи.

Розглянемо детальніше приклад для *regression testing*. Для цієї задачі задано $K = 0.4$, $\alpha = 0.2$, $\rho = 0.9$, $\delta = 0.3$ та $R = 0.5$. Підстановка у формулу (2) дає значення $S\_AI = 0.78$. Оскільки ризик не перевищує поріг примусового повернення людині, exploratory-характер відсутній, а *constraint layer* не активується, застосовується правило формули (3), за яким задача передається ШІ. При цьому відповідно до формули (6) призначається рівень автономності $L2$, оскільки ризик є середнім. Отже, мова не йде про неконтрольовану автоматизацію: ШІ може виконати основний операційний цикл, але людина повинна перевірити результат і взяти участь у випадку відхилення, збою або нетипового наслідку.

Практичне значення такого рішення полягає в тому, що воно дозволяє економити час команди на масових, повторюваних перевірках без втрати процесного контролю. Для regression testing ШІ може швидко формувати або запускати тести, перевіряти вихідні артефакти та готувати попередній звіт, тоді як фахівець зосереджується на інтерпретації аномалій, перевірці ризикових змін і підтвердженні релізної готовності. У такий спосіб модель не просто визначає виконавця, а й допомагає перерозподілити людську увагу на ті елементи, де вона приносить найбільшу цінність.

Показовим є також випадок *bug triage*. Тут підсумкове значення S_AI є низьким, але вирішальним фактором стає не тільки рахунок придатності, а й спрацювання *constraint layer*: неоднозначність досягає $\alpha = 0.5$, а доменна глибина — $\delta = 0.7$. Це означає, що навіть за умови використання ШІ для попереднього групування, пошуку дублікатів або узагальнення звітів остаточне рішення про класифікацію дефекту, бізнесовий пріоритет та відповідального виконавця повинно лишатися за людиною або прийматися в гібридному режимі. Саме на таких сценаріях видно, чому числова оцінка без шару обмежень була б недостатньою.

Окрім вибору виконавця, модель формалізує точки контролю Human-in-the-Loop. У таблиці 3 подано зв’язок між рівнем автономності та обов’язковими перевірками. Це важливо, бо одна й та сама задача може бути делегована ШІ, але з різною інтенсивністю людського нагляду залежно від ризику й контексту.

Таблиця 3. Контрольні точки Human-in-the-Loop

Рівень	Логіка керування	Обов’язкові контрольні точки
$L1$	Виконання під домінуючим контролем людини для високоризикових задач або задач із сильним потенційним впливом на якість релізу	$V1$ — pre-execution validation; $V2$ — post-generation validation; $V3$ — failure handling
$L2$	Гібридний режим для задач із середнім ризиком або для задач, що потребують обмеженої, але не нульової людської інтерпретації	$V2$ — post-generation validation; $V3$ — failure handling
$L3$	Режим із провідною роллю ШІ для низькоризикових, добре специфікованих і високо повторюваних задач	$V3$ — failure handling

Пілотне порівняння результатів моделі з експертною оцінкою на зазначених п’яти задачах показало збіг у чотирьох випадках. Єдине розходження стосувалося *bug triage*: модель визначила режим $H{+}A$, тоді як експерт був схильний повністю залишити задачу за людиною. Це розходження не є критичним, але воно виявляє чутливість моделі до порогових значень, насамперед до межі неоднозначності. З дослідницької точки зору це позитивний результат, оскільки він вказує на параметри, які потребують подальшого калібрування.

Узагальнюючи результати, можна зробити кілька практичних висновків. Поперше, ШІ доцільно використовувати там, де задача має стабільний шаблон виконання, чітко визначені входи та виходи і низький ризик помилкового рішення. По-друге, навіть високий показник придатності не повинен автоматично вести до повної автономії, якщо доменна глибина або неоднозначність залишаються суттєвими. По-третє, модель дозволяє командам не лише приймати одиничні рішення щодо делегування, а й формувати стабільну політику використання ШІ помічників у межах QA pipeline, у якій кожен тип задачі має прогнозований режим виконання, рівень автономності та мінімально необхідний набір перевірок.

7. Перспективи подальшого розвитку досліджень

Подальший розвиток запропонованої моделі пов’язаний передусім із переходом від експертно заданих ваг і порогів до емпірично відкаліброваних параметрів. Перспективним напрямом є накопичення корпусу реальних тестових задач із різних команд і доменів, для яких можна буде порівнювати рішення моделі, рішення експертів і фактичні результати виконання. Такий корпус дозволить оцінити не лише точність делегування, а й стійкість моделі до змін контексту, відмінностей між організаціями та впливу специфічних індустріальних вимог.

Другий напрям стосується деталізації самих критеріїв. У подальших роботах ризик може бути декомпозований на бізнесовий, технічний, регуляторний та репутаційний компоненти, а доменна глибина — на формалізоване і неформалізоване знання. Це дасть змогу зробити модель чутливішою до окремих класів систем, наприклад до медичних, фінансових або safety-critical продуктів. Доцільним є також дослідження міжекспертної узгодженості під час оцінювання параметрів, щоб відокремити недоліки самої моделі від варіативності людського судження.

Третій напрям пов’язаний із агентними системами нового покоління. У міру появи багатокрокових ШІ-агентів, здатних самостійно звертатися до репозиторіїв, систем відстеження дефектів, CI/CD-конвеєрів і тестових середовищ, модель делегування має враховувати не лише тип задачі, а й композицію агентних дій, тобто маршрут виконання, довжину ланцюжка, кількість зовнішніх інструментів та можливі точки втрати контексту. У цьому випадку Human-in-the-Loop має проектуватися не тільки на рівні фінального результату, а й на рівні окремих кроків оркестрації.

Нарешті, важливим є розроблення пояснювального інтерфейсу для моделі делегування. Користувач повинен бачити не тільки підсумкове рішення, а й аргументацію: які саме параметри обмежили автономію, чому спрацювало правило *constraint layer*, які контрольні точки були призначені та як можна змінити режим делегування через уточнення вимог, зниження неоднозначності або покращення контексту виконання. Така прозорість є необхідною умовою для реального впровадження моделі в інженерну практику.

8. Висновки

У роботі запропоновано формалізовану модель делегування задач тестування між людиною та ШІ-агентами в межах SDLC. Модель спирається на п’ять основних характеристик задачі — ризик, складність, неоднозначність, повторюваність і доменну глибину – та поєднує кількісну оцінку придатності до делегування з набором правил безпеки, які не дозволяють передавати критичні або слабо визначені задачі ШІ без належної участі людини. На відміну від суто евристичних рекомендацій, запропонований підхід задає чітку функцію делегування, явний порядок застосування обмежень, формальний зв’язок із рівнями автономності та набір контрольних точок Human-in-the-Loop.

Пілотне застосування моделі до типових задач тестування показало, що вона добре відокремлює повторювані та структуровані сценарії, де ШІ може бути основним виконавцем, від задач, які вимагають людської інтерпретації, творчого пошуку або глибокого доменного контексту. Практична користь моделі полягає в тому, що вона може бути використана як основа для внутрішніх політик використання ШІ в QA pipeline, для планування зон автоматизації та для підвищення пояснюваності рішень про делегування. Подальший розвиток моделі доцільно спрямувати на емпіричну калібровку ваг і порогів, накопичення галузевих даних, а також інтеграцію з багатокроковими агентними сценаріями та системами підтримки прийняття рішень.

Список літератури:

1) Parasuraman, R., Sheridan, T. B., & Wickens, C. D. (2000). A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans, 30*(3), 286–297. https://doi.org/10.1109/3468.844354

2) Amershi, S., Weld, D., Vorvoreanu, M., Fourney, A., Nushi, B., Collisson, P., Suh, J., Iqbal, S., Bennett, P. N., Inkpen, K., Teevan, J., Kikin-Gil, R., & Horvitz, E. (2019). Guidelines for Human-AI Interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (pp. 1–13). ACM. https://doi.org/10.1145/3290605.3300233

3) Lee, J. D., & See, K. A. (2004). Trust in automation: Designing for appropriate reliance. *Human Factors, 46*(1), 50–80. https://doi.org/10.1518/hfes.46.1.50_30392

4) ISO/IEC/IEEE. (2022). *ISO/IEC/IEEE 29119-1:2022. Software and systems engineering—Software testing—Part 1: General concepts*. International Organization for Standardization.

5) ISO/IEC/IEEE. (2021). *ISO/IEC/IEEE 29119-3:2021. Software and systems engineering—Software testing—Part 3: Test documentation*. International Organization for Standardization.

6) Myers, G. J., Sandler, C., & Badgett, T. (2011). *The art of software testing* (3rd ed.). Wiley.

7) Whittaker, J. A. (2009). *Exploratory software testing: Tips, tricks, tours, and techniques to guide test design*. Addison-Wesley Professional.

8) Wang, J., Huang, Y., Chen, C., Liu, Z., Wang, S., & Wang, Q. (2024). Software testing with large language models: Survey, landscape, and vision. *IEEE Transactions on Software Engineering*. https://doi.org/10.1109/TSE.2024.3368208

9) Ajorloo, S., Jamarani, A., Kashfi, M., Haghi Kashani, M., & Najafizadeh, A. (2024). A systematic review of machine learning methods in software testing. *Applied Soft Computing, 162*, 111805. https://doi.org/10.1016/j.asoc.2024.111805

10) Escalante-Viteri, A., & Mauricio, D. (2025). Artificial intelligence in software testing: A systematic review of a decade of evolution and taxonomy. *Algorithms, 18*(11), 717. https://doi.org/10.3390/a18110717

11) Li, Y., Liu, P., Wang, H., Chu, J., & Wong, W. E. (2025). Evaluating large language models for software testing. *Computer Standards & Interfaces, 93*, 103942. https://doi.org/10.1016/j.csi.2024.103942

12) Mosqueira-Rey, E., Hernandez-Pereira, E., Alonso-Rıos, D., Bobes-Bascaran, J., Fernandez-Leal, A., & Moret-Bonillo, V. (2023). Human-in-the-loop machine learning: A state of the art. *Artificial Intelligence Review, 56*, 3005–3054. https://doi.org/10.1007/s10462-022-10246-w

13) Wu, X., Xiao, L., Sun, Y., Zhang, J., Ma, T., & He, L. (2022). A survey of human-in-the-loop for machine learning. *Future Generation Computer Systems, 135*,

14) 364–381. https://doi.org/10.1016/j.future.2022.05.014

15) Saaty, T. L. (1980). *The analytic hierarchy process*. McGraw-Hill.

16) Yin, R. K. (2018). *Case study research and applications: Design and methods* (6th ed.). SAGE Publications.

A model for delegating testing tasks between humans and AI agents in the SDLC

Serhiy Kopovskyi
Lviv Polytechnic National University, Lviv, Ukraine
ORCID: 0009-0001-8969-2925

Abstract: This paper addresses the problem of distributing software testing tasks between human specialists and AI agents within the software development life cycle. The growing use of generative artificial intelligence, intelligent automation tools, and multi-step agent workflows increases the productivity of quality assurance teams, yet it also introduces the risk of opaque and weakly controlled delegation of critical testing decisions. The purpose of the study is to develop a formal model that determines the appropriate execution mode for a testing task: human-led, AI-led, or collaborative human plus AI execution. The proposed approach combines a quantitative suitability function with a constraint layer, three autonomy levels, and a Human-in-the-Loop mechanism. The model relies on normalized task characteristics, including risk, complexity, ambiguity, repeatability, and domain depth. A delegation function and a task suitability score are introduced to support explainable assignment decisions. In addition, explicit override rules prevent unsafe delegation in high risk and highly ambiguous contexts. The model is piloted on five representative testing tasks: test design, execution, exploratory testing, regression testing, and bug triage. The results show that AI is most effective in repetitive and well-structured scenarios, whereas exploratory work, high ambiguity, and deep domain dependency require stronger human participation. The practical value of the study lies in its applicability to QA pipeline planning, AI assistant governance, automation boundary definition, and the design of responsible Human-in-the-Loop policies for software engineering teams. The model also creates a foundation for future calibration with empirical data and for integration with multi-agent testing workflows.

Keywords: artificial intelligence; software testing; software development life cycle; task delegation; autonomy; human control.